# PEOPLE COUNTING USING OVERHEAD PANORAMIC CAMERA

*Yujia Xue and Celalettin Yurdakul*


Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

Aug. 30, 2018

Technical Report No. ECE-2018-03

# Contents

# List of Figures

# 1 Introduction

The main purpose of this project is to develop a comprehensive system that combines an overhead fisheye camera with deep learning algorithms in order to estimate the number of people inside an indoor scene [1]. Crowd estimation from single image or video stream arises in various applications including crowd control, public safety and reducing energy consumption of buildings [2]. Particularly, we are interested in accurate estimation of crowd in indoor scenes such as office rooms and coference halls so that it can adaptively adjust facilities such as lighting, heating and cooling [3][4].

# 2 Literature review

## 2.1 People Counting

The most extensively studied approach to people counting are feature based methods. Feature based methods involve certain steps: first subtract background (this involves recording a blank frame of the scene), second extract feature map with hand-crafted features (edge detection, SIFT, HOG [5], etc), finally perform a regression step to estimate the number of people [6][3]. Other approaches have also been developed for specific applications. In people counting from video stream, KLT tracker and clustering are used to detect and track moving people in a sequence of frames [7]. However, such methods do not work well when the scene contains multiple still people.

## 2.2 Convolutional Neural Network

Recently deep learning has been very popular and proved efficient in multiple vision tasks including: image classification, target detection and tracking, image reconstruction, image segmentation, etc [8][9]. Zhang [10] proposed the first CNN architecture to solve people counting. Other CNN architectures have been developed later to further improve accuracy and robustness [11]. There are mainly three types of approaches: directly estimating number of people, estimating density map of the scene and accurately labelling coordinates of individual people. In this project, we pursue an image-to-image architecture [8] that predict a density map of the scene. Another linear step (summation) is followed to further estimate the exact number of people in the image.

# 3 Implementation

In this section, we will divide the entire system into two parts: imaging part and algorithm part. On the imaging side, we will elaborate how we calibrate and correct lens distortion in fisheye camera. On the algorithm side, we will focus on the CNN based approach. We will discuss the detailed design of the network, preparation of dataset and training process.

## 3.1 RGB + Infrared Fisheye Imaging System

The conventional surveillance cameras which employ normal lenses can image the indoor/outdoor environments with minor lens distortion which is an aberration due to spherical shape of the lens. In an ideal system, the pinhole camera provides zero distortion. However, the field of view of the conventional lenses or pin-holes are limited to certain degrees, i.e., Field-of-View (FOV) $< 60°$ [12]. On the other hand, the fisheye cameras provide wide-angle FOV($> 120°$) with the expense of high lens distortion which requires image correction depending on the lens design parameters. The main challenge of correcting lens distortion is the varying magnification from image center to edges. The model based fisheye correction algorithms and methods will be applied to reduce the high-lens distortion in the fisheye cameras [13]. Also, it is important to note that the overhead fisheye cameras less prone to occlusion compare to the conventional cameras, that eliminates the necessity for multiple low-FOV cameras at different corners of a room.

The images will be acquired in both the infrared and optical spectra at different day times to account for different illumination levels. In IR regime, we expect that the acquisitions at low-light scheme will be superior to RGB regime. Thus, the IR capability of the overhead camera is advantageous in the low-light indoor areas. The IR images from the *VRCAM* can be easily obtained through the on/off switch on web interface of the camera settings. For the sake of convenience, we opt for the RGB/indoor setting in the preliminary results. We will further investigate the integration of IR capability of the *VRCAM* into our detection method.

### 3.1.1 Fisheye-lens Calibration and Correction

This section explains the calibration of the *VRCAM* and the correction of the distorted images using the Matlab vision toolbox [14]. The toolbox implements the generalized omni-directional camera calibration method introduced by Scaramuzza *et. al.* for [15]. The paper demonstrated the automatic and robust calibration which does not require any priori information on the camera model, thus, can be applicable to many fish-eye camera in the market. Once the calibration is implemented, the fish-eye camera images can be easily corrected. In a nutshell, the pipeline of the proposed calibration method is the following:

- Calibration image acquisition using a known planar checkerboard pattern
- Checkerboard feature extractions, i.e., corners, in terms of sensor plane coordinates $(u", v")$
- Least square minimization problem of system linear equation for extrinsic parameters
- Extrinsic parameter substitution to solve for intrinsic parameters
- Iteratively detection of the object center

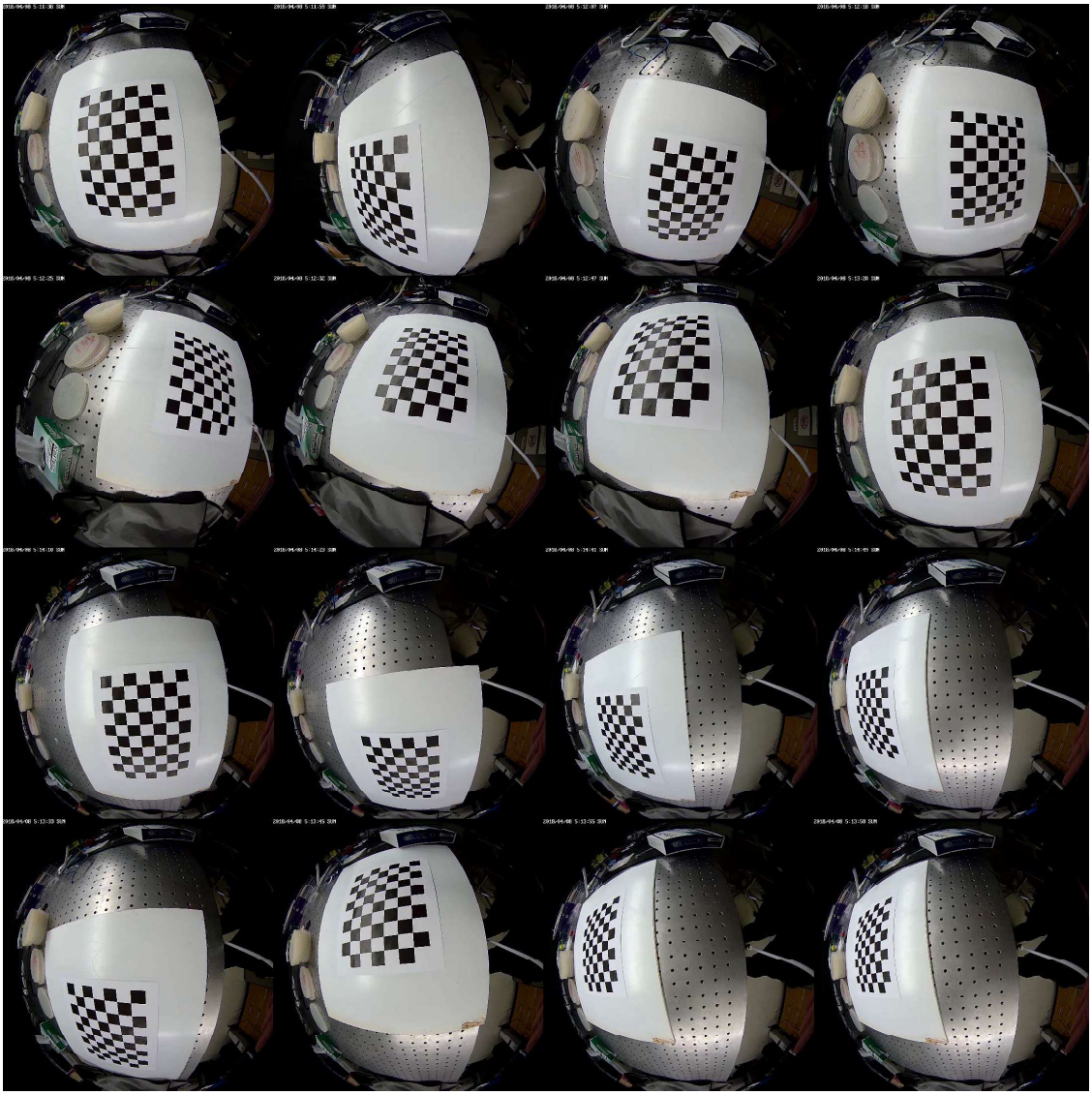

Figure 1: Calibration data set obtained with *VRCAM*

- Maximum likelihood estimate of extrinsic and intrinsic parameters

The image coordinates $(u", v")$ at the sensor plane is related to the object coordinates $(u', v')$ with the affine transformation matrices as following;

$$\begin{bmatrix} u'' \\ v'' \\ w'' \end{bmatrix} = R\lambda \begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} + T \tag{1}$$

where R and T denote rotation and translation matrices, $\lambda$ denotes depth factor, and

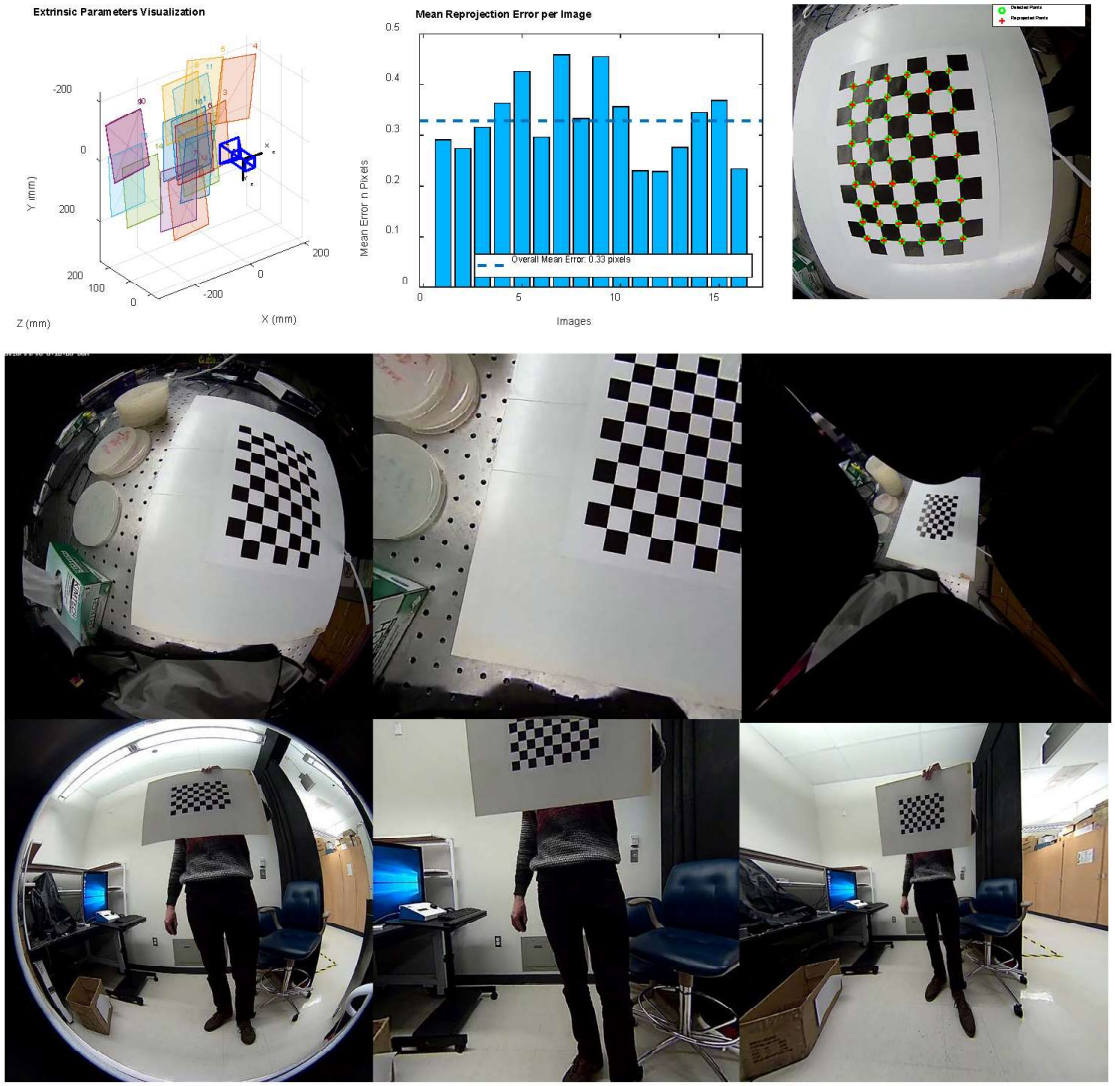


Figure 2: (Upper-left)Extrinsic parameters visualization extracted from calibration set, (Upper-center) mean reprojection error per image in calibration error, (Upper-right) Generated and projected corners on checkerboard , (Middle-left) distorted image taken by *VRCAM*, (Middle-center) undistorted image using same option, (Lower-left) distorted image taken by *VRCAM*,(Lower-center) undistorted image using same option, (Lower-Right) undistorted image using scale factor 0.4,

$w'$ denotes the non-linear function in a polynomial form as following

$$w' = f(u'', v'') = \sum_{n=0}^{N} \alpha_n \rho''^n \tag{2}$$

$$\rho''^n = \sqrt{u''^2, v''^2}$$

the function f is a Taylor expansion that depicts the lens distortion. The first derivative of the f is zero which implies $\alpha_1 = 0$. The extrinsic parameters are the matrices R and T, and intrinsic parameters are the coefficients of the polynomial function f. To solve extrinsic and intrinsic parameters separately, the cross product of the linear-equation 1 with left-hand side of the equation is calculated. The cross products results in two linear-equation, first as a function of intrinsic parameters and the second the extrinsic parameters [15]. First, the linear-equation for the extrinsic parameters is solved by minimazing least-square problem. Then, the calculated extrinsic parameters is substituted into the first linear-equation. Then, the equation is solved for intrinsic parameters. This linear problem is overdetermined, so that, the number of N can be minimized to reduce the computational cost. The minimization of N is determined by calculating the reprojection error which is the distance between the projected and the correct object points. Also, algorithm implements iterative calibration method to eliminate the image center mismatch with the sensor. Moreover, since the previous methods only minimize the geometric distance which does not account for noise, the non-linear maximum likehood estimate is implemented. The results in [15] shows that the reprojection error is less compare to the linear minimization.

We implement the camera calibration by following the Matlab toolbox example for GoPro cameras. First, we take a series of calibration images at different position relative to the camera (Fig. 1). The checkerboard image with known dimensions is used during the calibration. Then, we generate the checkerboard points using Matlab function *generateCheckerboardPoints*. The extrinsic and intrinsic calibration parameters are estimated using the function *estimateFisheyeParameters*. The reprojection and generated corners and the generated error of the calibration are in Fig. 2. The function *undistordfisheye* corrects the distorted fish-eye camera. The*undistordfisheye* has options to generate corrected image such as scale factor stretch edge of the image points to fit into same image size, and the full view which leave the edge of the image points as it is. As seen from the 2, the undistortion stretches a lot the edges of the images. This stems from the high distortion of the *VRCAM*, thus, scale factor options gives more meaningful.

## 3.2 CNN Model Design and Training

In this part, we will elaborate the architecture of our convolutional neural network, preparation and pre-processing of dataset, and training phase.

### 3.2.1 Image-to-image CNN architecture

We choose U-Net type of encoder-decoder architecture to perform image-to-image estimation of density map. There are two main concerns of using U-Net: previous feature layers can be easily accessed via skip connection; the depth of network is tunable. The multi-scale pyramid architecture with skip connection can fully utilize different feature levels. U-Net architecture can easily be modified with advanced building blocks (DenseBlock) and its depth can also been easily tuned. In this project,

we use a 5-layer U-Net with DenseBlock. In this project, two models are trained: one takes RGB image as input and the other takes in grayscale image. We want to find out how color information improves counting accuracy.

Fig. 3 illustrates the architecture of our CNN model. Blue blocks are input &

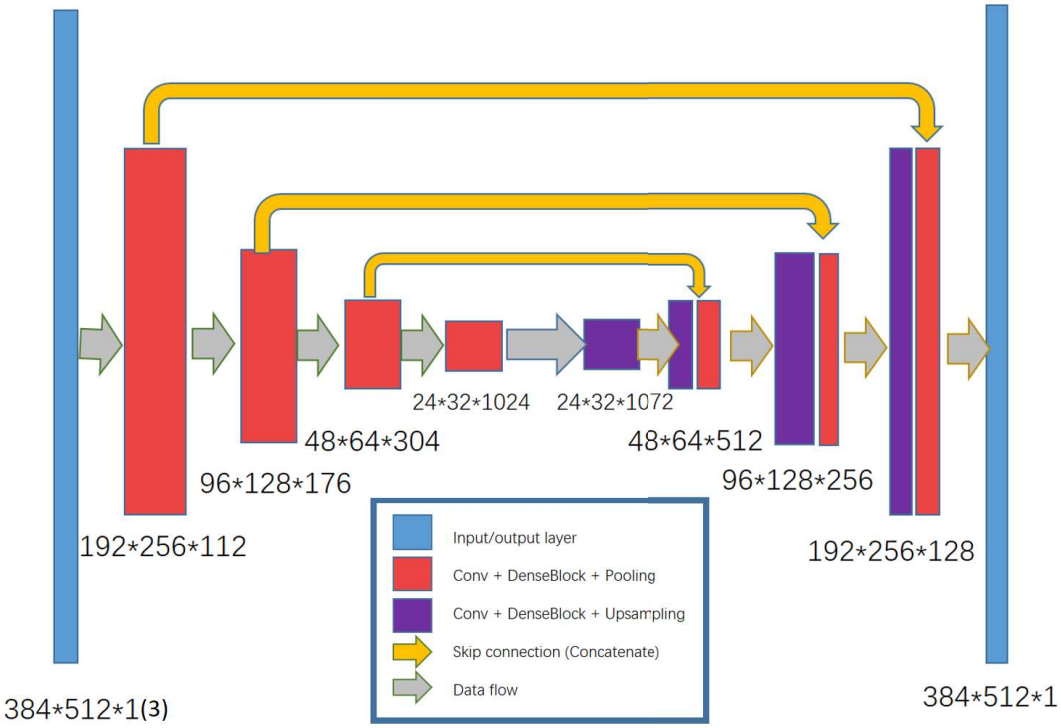


Figure 3: Our CNN architecture is a 5-layer U-Net modified with DenseBlock. Each red block consists of a convolution layer, a DenseBlock and a $2 \times 2$ max-pooling layer. Each magenta block consists of a convolution layer, a DenseBlock and a $2 \times 2$ upsampling layer. Yellow arrows stand for skip connection meaning we concatenate two layers together.

output layer with spatial dimension of $384 \times 512$. Red blocks consist of a convolution layer (64 kernels, kernel size $3 \times 3$, ReLU activation), a DenseBlock (3 layers, each layer has 16 filters) and a max-pooling layer (pooling size $2 \times 2$) [9]. 4 illustrates the inner structure of a DenseBlock. Similarly magenta blocks consist of a convolution layer, a dense block and an upsampling layer. Yellow arrows are skip connection between layers which have the same spatial dimensions. Grey arrows simply illustrates data flow. Pooling layer and upsampling layer squeeze and expand spatial dimension by a factor of two respectively. Notice the architecture has a pyramid-like structure so that spatial dimensional is compressed at the beginning and later interpolated for better feature learning. The idea of using skip connection is that in order to detect people, both high-level features (eyes, nose, etc.) and low-level features (edges, etc.)

must be used. Skip connection concatenates feature maps from different levels.

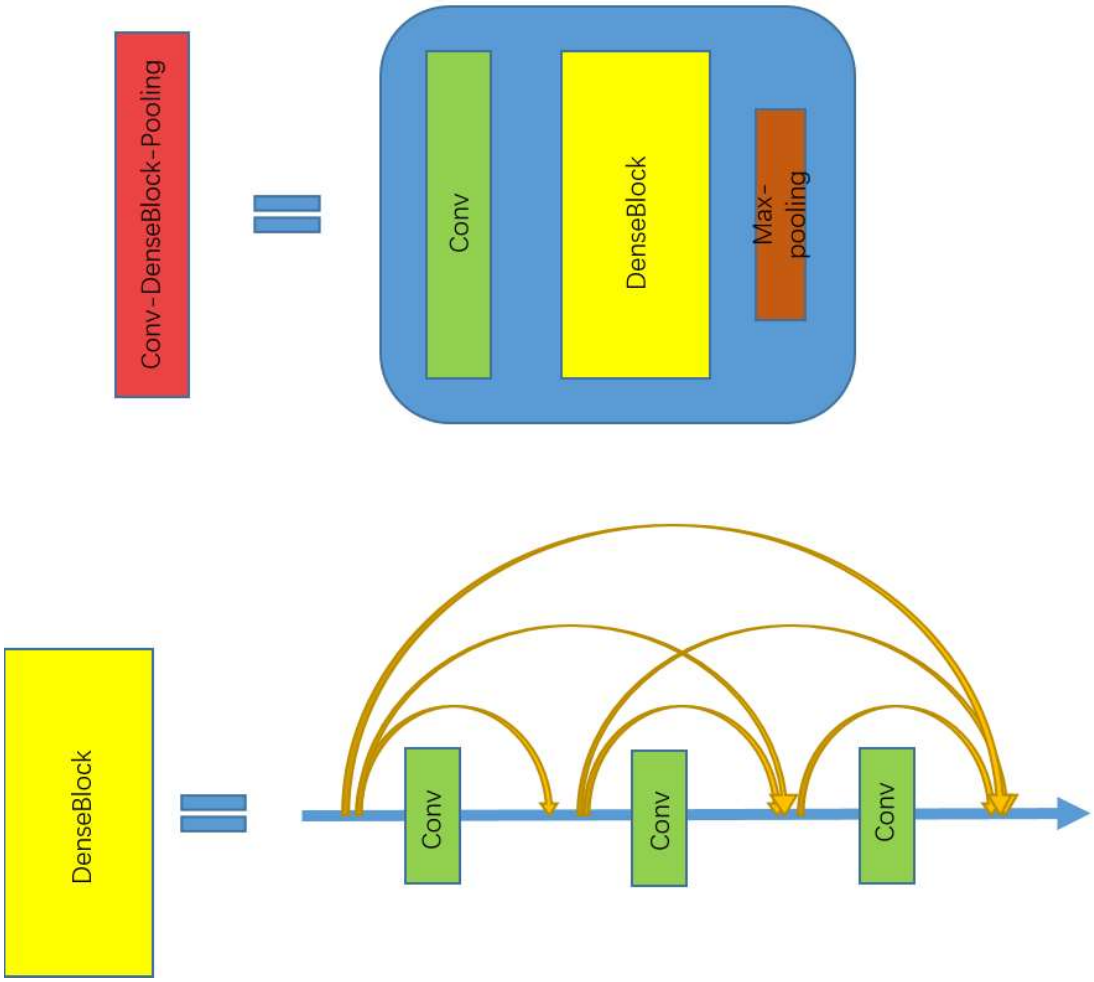


Figure 4: Illustration of a Conv+DenseBlock+Pooling block. Each DenseBlock contains three inter-connected convolution layers.

### 3.2.2 Data preparation

The neural network is trained on a public people counting dataset from ShanghaiTech University [2]. The dataset contains more than 700 outdoor crowd images, along with manually labelled coordinates of individual persons. To perform an image-to-image training, we need to first generate synthetic density maps from the coordinates. Here we use Gaussian kernels to emulate the density map. The density map for each image is synthesized by the summation of Gaussian kernels with varying size centred

at individual locations.

$$D_{map}(x,y) = \sum_{i=1}^{N} exp(-\frac{(x-l_x^i)^2 + (y-l_y^i)^2}{(\sigma_{min} + \alpha y)^2}) \tag{3}$$

where $N$ is the total number of people in the image, $l_x^i, l_y^i$ are the image coordinates of $i^{th}$ person in the image, parameter $\alpha$ controls how kernel size changes vertically. The reason we use Gaussian kernels with varying sizes stems from the observation that images are taken with a perspective angle so that people in the front will appear larger in size while people in the background will be smaller. To account for this magnification change, we make the kernel size adaptive to vertical location. We also make an assumption here that all images are taken with the same perspective angle therefore $\alpha$ is a constant which simplifies the model.

Fig. 5 shows 6 samples from the training dataset. The ground truth density map is synthesized using the above procedure.

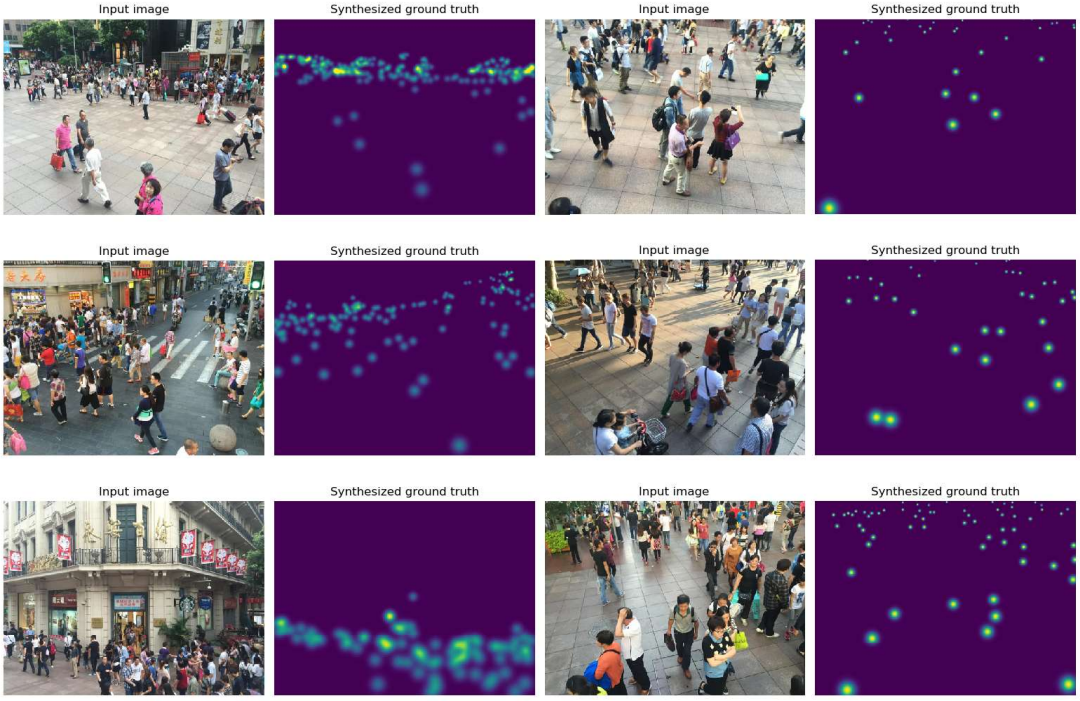


Figure 5: Sample image pairs from the training dataset.

### 3.2.3 Model implementation and training

In this section, we will discuss the implementation and training phase. The model is implemented in Keras with Tensorflow as the backend. We use binary cross-entropy

as the loss function in training, which is defined as:

$$loss_{ENTP} = -\sum_{i,j} y_{true}(i,j) log(y_{pred}(i,j)) \tag{4}$$

The reason we used cross-entropy as loss function instead of other spatial loss functions like mean-squared-error or mean-absolute-error is that we found for sparse output (like density maps), the network intends to predict all zeros when using spatial loss functions. Spatial loss functions evaluate the network by how accurate the prediction is compared to ground truth. Because most of the pixels are zeros in the ground truth density map, zero-prediction achieves high accuracy. However, when using cross-entropy zero output has a much larger penalty in the loss function and provides larger gradient for training.

We used ADAM optimizer [16] to minimize the above loss function. The training dataset contains 400 image pairs from the dataset (the other image pairs are used to test the network after training). We initialized the learning rate to be $10^{-4}$ and trained the network for 200 epochs with mini-batch size of 3. The model was fine-tuned with learning rate $10^{-6}$ for another 300 epochs. The whole training process took 14 hours on a single Nvidia Tesla P100 GPU. Fig. 6 shows training and testing loss of our models. It is observed that both RGB and grayscale model converges to the same amount of training error. The RGB model performs little bit better than the grayscale model on test dataset.

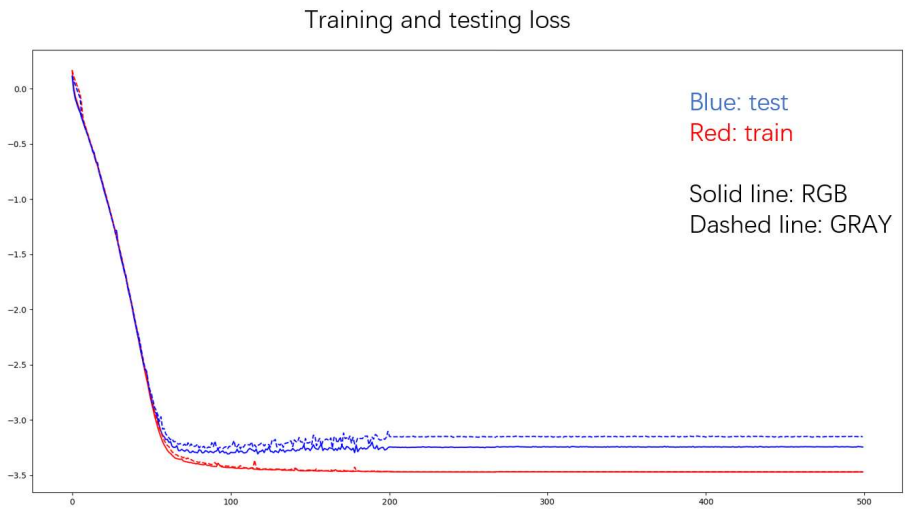


Figure 6: Training and testing loss of two models(RGB model and grayscale model)

### 3.3 Crowd Count

After the CNN predicts a density map of the image, we further predict the exact crowd count by a weighted summation.

$$\hat{N} = \sum_{x,y} D_{map}^{pred}(x,y) \times W(y) \tag{5}$$

$W(y)$ is weight function depending on vertical coordinates. Previously we synthesize density maps by Gaussian kernels with varying sizes. Therefore, each Gaussian kernel will contribute differently to the summation. Weight function here is used to normalize the summation of each Gaussian kernel. $W(y)$ is precomputed as:

$$W(y) = \frac{1}{\sum_{i,j} exp(-\frac{i^2+j^2}{(\sigma_{min}+\alpha y)^2})} \tag{6}$$

## 4 Experimental results

In this section, we show the results of our CNN with different kinds of test images. We test the network on test dataset from ShanghaiTech University [2], crowd images from Google, corrected images from fisheye camera.

### 4.1 Results on test dataset

Fig. 7 shows the results on test dataset: input image, the predicted density map and the ground truth density map(synthesized using the same method in previous section). The crowd count is on the title of each sub-plot.

Since we have the ground truth of test dataset, we further compute the relative error of the predicted output and its confusion matrix. Relative error rate is defined as:

$$\epsilon_{rel} = mean(\frac{N - \hat{N}}{N}) \tag{7}$$

The relative error on test dataset $\epsilon_{rel} = 9.27\%$. Fig. 8 shows the confusion matrices of two models. Ideally, all samples should be on the main diagonal. Here we observed deviation which means the proposed approach cannot perfectly predict the crowd count.

### 4.2 Results on crowd images from Google

We also tested the network crowd images downloaded from Google. Fig. 9 shows the predicted density maps and its crowd count in title. For better visualization, density


Figure 7: Results on test dataset.Left: input image (RGB and grayscale), middle: predicted density map, right: ground truth

map is overlapped with the input image in order to see how accurate the neural network can detect people.

## 4.3 Results on corrected fisheye image

When we test the neural network with corrected images from fisheye camera, the neural network failed to predict locations of people (Fig. 10). There are two main reasons: first the perspective of fisheye camera is very different from the perspective

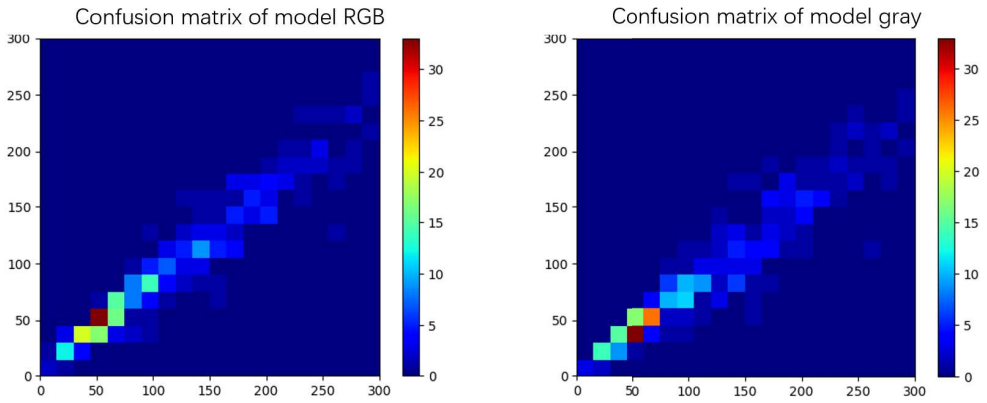


Figure 8: Confusion matrices of model RGB and model grayscale. It's observed that the predicted crowd count is always less than the true count.

in the training dataset; second compared to the dataset, the crowd are more sparse in corrected fisheye image.

### 4.4 Re-train CNN on rotated images

As mentioned above, we observe that our neural network fails to detect people in fisheye images due to perspective angle change. Some faces appear horizontal after correcting distortion. To account of the perspective change, we augment the training dataset by rotating both input images and density maps. With the augmented dataset, we re-trained the network and tested on images in which faces are not vertically oriented. Fig. 11 shows the predicted density maps from: first column (RGB model trained on non-rotated dataset), second column (grayscale model trained on non-rotated dataset), third colum (RGB model trained on rotated dataset), fourth column (grayscale model trained on rotated dataset). The results demonstrate that with augmented dataset, the neural network does a better job at detecting people whose faces are not vertically oriented.

## 5 Conclusions

- The training dataset for CNN has small field-of-view and minor lens distortion compared to fisheye camera. Although the image correction method mentioned earlier is working well in the center of the lens, it fails far away from the lens center due to the high-distortion. This will results in false-positives/negatives at the exteriors of the FOV. We might crop the undistorted images in the center to eliminate the highly-distorted edges.
- The training data set is mostly taken at far from the humans. We are expecting

to detect the people on their desk, however, the detection of standing people close to camera center is still challenging (the network failed to detect person that is too close to the camera). To overcome this problem we could combine the feature-based method with CNN.

- The CNN data set labels are the coordinates of the heads in the images. The ground-truth is obtained by convolution of head coordinates with a Gaussian disk. However, this approach assumes the same magnification for all head locations which is not physically true. The further away from camera causes the further demagnification of the object. Thus, we will implement the shift-variant gaussian filter along the vertical axis.

- We still do not know the low-light scene performance of the proposed method. The IR capability of the camera might be sufficient to get enough light, however, it provides grayscale-like single channel image as opposed the three-channel (RGB) CNN data set. We will map the CNN data set to grayscale image to mimic the IR images.

# References

[1] Lingke Zeng, Xiangmin Xu, Bolun Cai, Suo Qiu, and Tong Zhang. Multi-scale convolutional neural networks for crowd counting. *arXiv preprint arXiv:1702.02359*, 2017.

[2] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 589–597, 2016.

[3] Bernt Schiele, Mykhaylo Andriluka, Nikodem Majer, Stefan Roth, and Christian Wojek. Visual people detection: Different models, comparison and discussion. In *Proceedings of the IEEE ICRA Workshop on People Detection and Tracking*, 2009.

[4] Muhammad Aftab, Chien Chen, Chi-Kin Chau, and Talal Rahwan. Automatic hvac control with real-time occupancy recognition and simulation-guided model predictive control in low-cost embedded system. *Energy and Buildings*, 154:141–156, 2017.

[5] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[6] Nil Goyette, Pierre-Marc Jodoin, Fatih Porikli, Janusz Konrad, and Prakash Ishwar. A novel video dataset for change detection benchmarking. *IEEE Transactions on Image Processing*, 23(11):4663–4679, 2014.

[7] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3169–3176. IEEE, 2011.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[9] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017.

[10] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 833–841. IEEE, 2015.

[11] Lokesh Boominathan, Srinivas SS Kruthiventi, and R Venkatesh Babu. Crowdnet: A deep convolutional network for dense crowd counting. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 640–644. ACM, 2016.

[12] Zhong Zhang, Peter L. Venetianer, and Alan J. Lipton. A Robust Human Detection and Tracking System Using a Human-Model-Based Camera Calibration. In *The Eighth International Workshop on Visual Surveillance - VS2008*, Marseille, France, October 2008. Graeme Jones and Tieniu Tan and Steve Maybank and Dimitrios Makris.

[13] Ciaran Hughes, Martin Glavin, Edward Jones, and Patrick Denny. Review of geometric distortion compensation in fish-eye cameras.

[14] Computer vision system toolbox, MATLAB Version: 9.3.0.713579 (R2017b). The MathWorks, Natick, MA, USA.

[15] D. Scaramuzza, A. Martinelli, and R. Siegwart. A toolbox for easily calibrating omnidirectional cameras. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5695–5701, Oct 2006.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.


Figure 9: Results tested on crowd images from Google. The crowd count of predicted density map is titled. The right column is for better visualization.

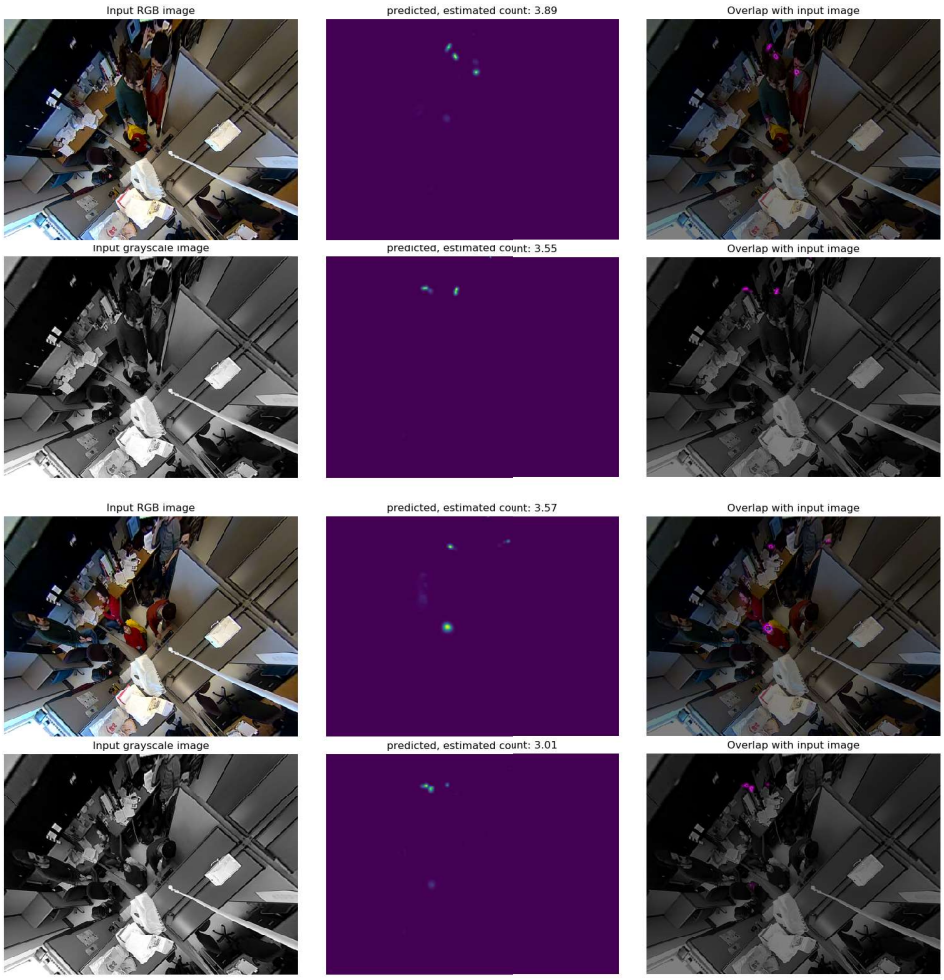


Figure 10: Results tested on corrected fisheye image. Due to the change of perspective, the neural network failed to detect people location.

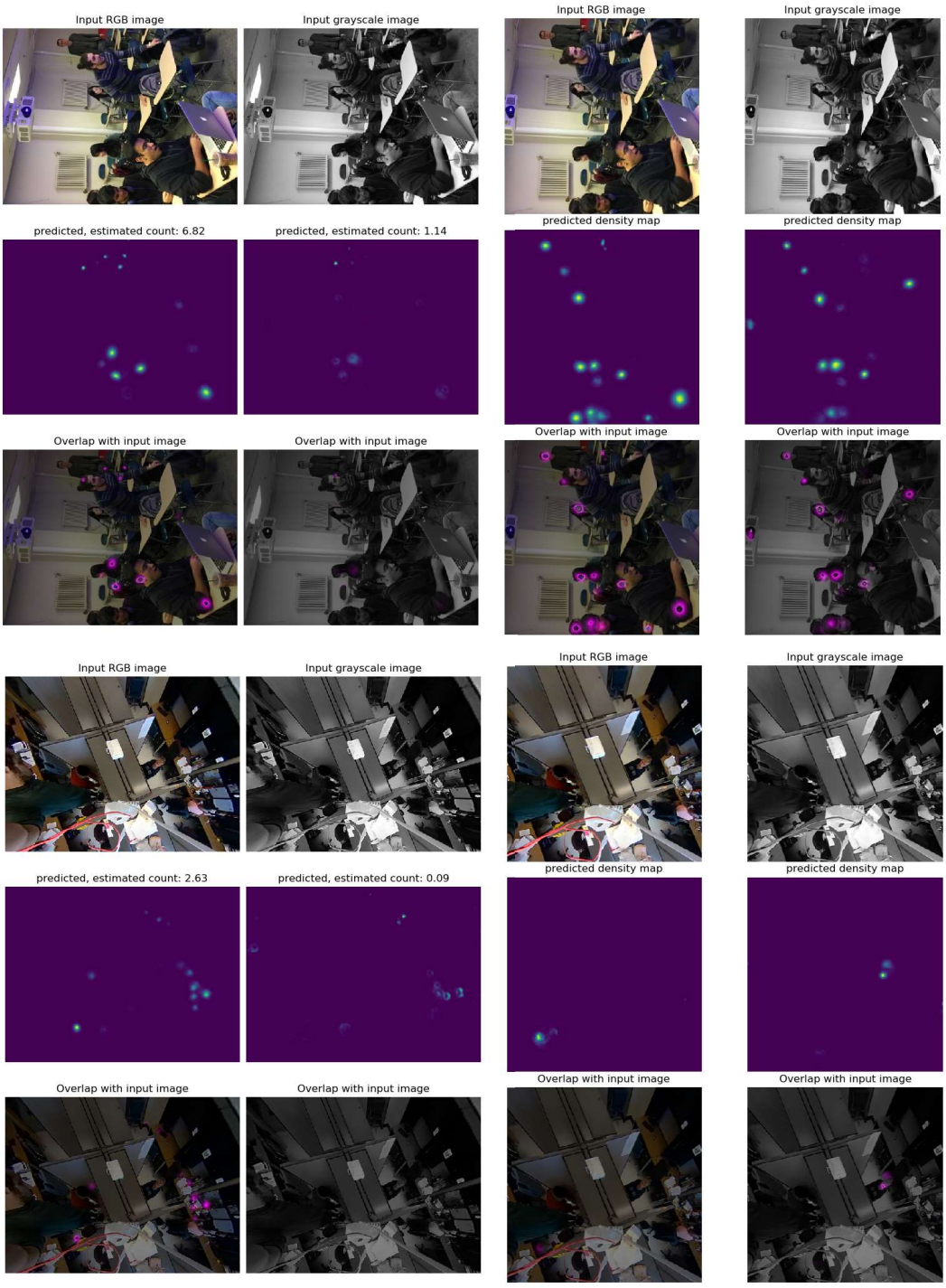


Figure 11: From left to right: RGB model trained on non-rotated dataset, grayscale model trained on non-rotated dataset, RGB model trained on rotated dataset; grayscale model trained on rotated dataset.