Sangogboye and Kjærgaard *Energy Informatics* (2018) 1:12
https://doi.org/10.1186/s42162-018-0016-4

Energy Informatics

RESEARCH Open Access

PreCount: a predictive model for correcting real-time occupancy count data

Fisayo Caleb Sangogboye* and Mikkel Baun Kjærgaard

*Correspondence:
fsan@mmmi.sdu.dk
SDU Center for Energy Informatics
Mærsk McKinney Møller Institute
University of Southern Denmark,
Odense, Denmark

Abstract

Sensing the number of people occupying a building in real-time facilitates a number of pervasive applications within the area of building energy optimization and adaptive control. To ascertain occupant counts, the adoption of camera-based sensors i.e. 3D stereo-vision and thermal cameras have grown significantly. However, camera-based sensors can only produce occupant counts with accumulating errors. Existing methods for correcting such errors can only correct erroneous count data at the end of the day and not in real-time. However, many applications depend on real-time corrected counts. In this paper, we present an algorithm named PreCount for accurately correcting raw counts in real-time. The core idea of PreCount is to learn error estimates from the past. We evaluated the accuracy of the PreCount algorithm using datasets from four buildings. Also, the Normalized Root Mean Squared Error was used to evaluate the performance of PreCount. Our evaluation results show that in real-time PreCount achieved a significantly lower Normalized Root Mean Squared Error compared to raw counts and other correction approach with a maximum error reduction of 68% when benchmarked with ground truth data. By presenting a more accurate algorithm for estimating occupant counts in real-time, we hope to enable buildings to better serve the actual number of people to improve both occupant comfort and energy efficiency.

Keywords: Occupant count, Real-time system, Machine learning, Energy optimization

Introduction

Estimating the number of people in commercial and public buildings with the aid of pervasive sensors is receiving increasing attention. This is because the permeation of pervasive computing have prospects in facilitating several building applications. One application area customary to commercial outlets is in the area of disaster prevention and management. Most commercial outlets are mandated by government laws to provide accurate estimates of occupant counts in real-time. Lastly, occupant presence is the major driving factor for energy consumption in buildings (Chang and Hong 2013; Caucheteux et al. 2013). Consequently, several approaches for simulating, monitoring and optimizing energy consumption in buildings including the model based approaches have become prominent and have received significant research attention (Arendt et al. 2016). A similar research attention is currently given to the concept of demand response (DR) in commercial buildings for facilitating the control of energy demand (Kjærgaard et al. 2016). DR leverages on forecasts and real-time data that comprises majorly of occupant counts to schedule DR events without impeding occupant comfort. Recently, the United States

Department of Energy (US Department of Energy 2017) highlighted that, obtaining very accurate occupant counts can facilitate a 30% energy savings in buildings and it can enable building management systems to achieve occupant comfort in-line with the ASHRAE standard.

A range of sensor methods has been applied for estimating occupant counts. One line of research has studied reusing common building sensors for occupant counting. These sensors include CO_2 sensors, PIR sensors, energy metering, sensors of HVAC systems or WiFi access points (Christensen et al. 2014). However, these sensors are very inaccurate in their estimates. Sangogboye et al. in Sangoboye and Kjærgaard (2016) highlighted that camera-based counting technologies amongst other sensors are the state-of-art method for obtaining high quality count estimates in commercial buildings. Kjærgaard et al. (2016) corroborated this proposition by benchmarking the raw counts retrieved from both 3D stereo-vision cameras and PIR sensors with manual ground truth count estimates. The obtained count estimates from both the 3D stereo-vision cameras and PIR sensors achieved Root Mean Square Errors (RMSE) of 3.3 and 21.7, respectively.

However, while camera-based count estimates are more accurate than count estimates from other sensors, these sensor suffers from errors that are accumulated and propagated over time. For instance, when a camera sensor misses an occupant, this error is propagated until another offsetting error occurs or a correction approach is adopted to remove the error (Beltran et al. 2013). These errors are a result of several causes such as poor lighting condition and occlusion. The state-of-art methods for correcting such erroneous count estimates are only capable of correcting erroneous counts in the past and not in real-time. This is because, these past correction methods are governed by constraints that require data until the end of the day. Subsequently, statistics about this data are used to formulate an appropriate correction model for a day (Sangoboye and Kjærgaard 2016; Hutchins et al. 2007; Kuutti et al. 2014).

The topic of correcting erroneous count estimates in real-time is to the best of our knowledge uncharted and it differs from the previously stated approaches for correcting counts estimates in the past. Figure 1 illustrates the difference between past correction and real-time correction. Here, past corrections are computed at end of the day $t_{24:00}$ and it covers count correction from $t_{24:00}$ back to the beginning of the day $t_{00:00}$. Conversely, real-time corrections are computed at time t_{now} and it only covers the current time of the day. For instance, the past correction algorithm presented in Sangoboye and Kjærgaard (2016) cannot suffice for real-time correction because they require count data for the whole day. This count data is used to instantiate the upper bound of the probability and propagation matrices that are used in estimating count errors. Table 1 categorizes these past correction methods into two approaches according to their correction methodologies and inherent constraints.

From the categorization in Table 1, it can be observed that the naive approach is insensitive to overestimation in buildings. This problem was highlighted in one of the building cases presented in Sangoboye and Kjærgaard (2016), where the past count estimates obtained through the naive approach is the same as the erroneous raw count. The probabilistic approach, on the other hand, achieved high fidelity count estimates compared to ground truth data for all building cases presented in Sangoboye and Kjærgaard (2016). Secondly, unlike the probabilistic approach which is non-adaptable for real-time

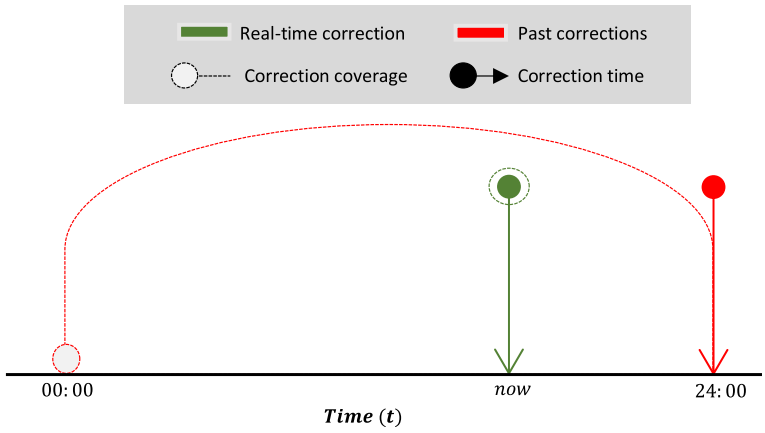

Fig. 1 Real-time correction versus past correction

correction, the naive approach can be easily adapted for correcting count estimates in real-time, but it does not provide high accuracy.

In this paper, we present PreCount - an algorithm for estimating count errors in real-time via a predictive model. This algorithm leverages the accuracy of the probabilistic correction methods to model erroneous counts in a dataset. Subsequently, it utilizes the derived model to accurately determine the error estimates in real-time. We make the following contributions:

1. Formulate the real-time occupant count estimation problem using past corrections and their corresponding erroneous raw counts.
2. Present the PreCount algorithm, how it trains several correction models and utilizes the best correction model to estimate count error in time.
3. Present a feature analysis and preparation process that robustly identify count errors.
4. Present an extensive evaluation result that highlights the overall performance of PreCount in four building cases. Datasets from the first three building cases were used to concretize and evaluate our design assumptions. The dataset from the last building case was used to test and validate these design assumptions. Also, we evaluate PreCount for different sizes of training data.

Table 1 Approaches for past correction

	Methodology	Constraints
Naive approach	1. It subdivides past raw counts into daily profiles with timestamps $\{t_0, \ldots, t_n\}$, where n is the end of the day.	It assumes that most buildings have periods during night time where the number of occupants go to zero.
	2. It initializes the first transition of each day i.e. transition at time t_0 to zero.	
	3. It assigns zero to all negative counts.	
Probabilistic approach	1. It subdivides past raw counts into daily profiles with timestamps $\{t_0, \ldots, t_n\}$, where n is the end of the day.	1. It requires the observed maximum number of occupants to formulate and compute a transition matrix.
	2. It corrects each daily profile at time t_n using specific methods proposed in either Sangoboye and Kjærgaard (2016) and Kuutti et al. (2014).	2. It requires a complete estimation of the transition matrix propagation for each day to accurately correct obtained counts.

The paper is structured as follows. In the "Related work" section, we present the state-of-art that highlights the methods for correcting erroneous occupant counts in the past. In the "PreCount" section, we introduce the PreCount algorithm and its elements. This "Evaluation" section includes a detailed description of the feature preparation stage and the correction methods used for correcting erroneous occupant counts in real-time. In the same section, we present the building cases and datasets used in this paper, the error patterns in our datasets and we justify the relevant features for identifying count errors in real-time. In the "Evaluation" section, we evaluate the performance of PreCount with three evaluation cases and with the datasets from the four building cases. In the "Discussion" section, we discuss the limitations of this study and propose relevant improvement opportunities. Finally, we conclude in the "Conclusions" section.

Related work

In this section, we present the State of the art for correcting and estimating occupancy count in buildings. In the first subsection, we present the state-of-art for correcting count errors in camera technologies while in the second subsection, we present other related works for estimating occupant count in buildings.

Occupant count correction methods

Ihler et al. (Ihler et al. 2006) proposes a probabilistic method for correcting past raw counts obtained from a single sensor system. This method models count data from a single sensor by formulating a probabilistic model for each sensor, and the model differentiates between usual activity and an unusual burst of occupant count. The models are trained with six weeks of data. An inhomogeneous Poisson process is used to represent usual human activity while a hidden Markov process is used to model bursts of unusual behavior. Hutchins et al. (Hutchins et al. 2007) extends this method to a multi-sensor environment by linking individual sensor streams to form a multiple-sensor probabilistic model. This multiple-sensor probabilistic model is represented using a directed graphical model and it is used for estimating occupant counts in buildings.

On the other hand, Sangogboye et al. (Sangoboye and Kjærgaard 2016) introduces a training free probabilistic approach for correcting past counts named PLCOUNT. PLCOUNT takes both occupant transitions and cumulative counts as input to formulate an occupant count problem in the form of both a probability and propagation matrix respectively. Given this formulation, PLCOUNT initializes a probability matrix by estimating the likelihood that a space is occupied by a specific number of people. Subsequently, PLCOUNT computes the remaining time steps until the end of the day in the probability matrix by estimating the probabilities of measured count transitions. Lastly, PLCOUNT performs a backtracking operation on the propagation matrix to compute a new count estimate for each time step. PLCOUNT reported an 86% error reduction when it is compared to raw counts and benchmarked with ground truth count data.

Occupant count estimation methods

Erickson et al. (Erickson et al. 2011) presented OBSERVE - an occupant count estimation model that utilizes a Markov chain to model inter-room relationships and to estimate occupant count in real-time. In this work occupant count data were collected from 16

node sensor network of cameras (Kamthe et al. 2009). This work proposed two models - the closed distance Markov chain and the blended Markov chain for model occupant transitions between states. Erickson et al. evaluated the proposed models using three evaluation metrics - occupant variation, Jensen Shannon divergence and occupant arrival and departure rates. The obtained evaluation result indicates that both the blended Markov chain model and the closest distance Markov chain achieved similar performance with the Jensen Shannon metric while the blended Markov chain model achieved better performance for other metrics.

Beltran et al. proposed ThermoSense (Beltran et al. 2013) for estimating occupant counts with PIR sensors and thermal based sensing. The PIR sensor is used to determine the presence of occupants in rooms while the thermal based sensing is used to detect the thermal footprint of occupants for deriving the count of occupants in a room. The PIR sensor also helps to reduce the amount of energy used by the thermal-based sensor and other components of the sensing system. To facilitate the count of people in a room, ThermoSense firstly creates a thermal map when a room is unoccupied. This map is continually updated using the lowest temperature during times when the room is occupied or every 15 min when no movement is discovered in a room. Secondly, ThermoSense converts a measured 8X8 grid temperature values to create three feature vectors for training and comparing three different prediction methods – Artificial Neural Network (ANN), K-Nearest Neighbor (KNN) and Linear regression. The three feature vectors include the total active points, number of connected components and size of the largest component. Subsequently, an average filter is applied to the obtained raw occupancy estimate. The comparative result obtained shows that the KNN classifier achieved the least NRMSE value of 25%.

Erickson et al. (2009) deployed a wireless camera sensor network (Kamthe et al. 2009) for facilitating the estimation of occupant mobility patterns in a large multi-function university building. This work uses two predictive models for learning and estimating occupant counts. The first model - a multivariate-gaussian method learns and predict the count estimate of an occupant in a building. While the second model - an agent-based model simulates the mobility patterns of occupants. The multivariate model partitions occupant count data from several locations in a building into hourly slots and compute the mean and standard deviate for each slot. These estimates with probability distribution function and some expert assumptions from ground truth observations are used for randomly and collectively draw occupant distribution for each location. The agent-based model, on the other hand, is used to simulate each occupant's movement by modeling itineraries, path choice, and walking behavior. The evaluation result of both models shows similar performance and indicates that the multivariate model achieved an average NRMSE of 42.6% while the agent-based model achieved an average NRMSE of 43.3% for two rooms respectively.

Ekwevugbe et al. (2013) proposed an occupant estimation method that utilizes multi-sensory data from CO2, sound level, device case temperature and motion sensors to estimate occupant number in an open-plan office. This method utilizes a symmetrical uncertainty analysis for feature selection and a genetic based search to evaluate the optimal sensor combination for estimating occupant counts. Also, ground truth data are obtained using a thermal camera system to train and test an ANN estimation model. The evaluation results from the proposed model indicate

a minimum and maximum accuracy and RMSE of 67% and 1.01 and 75% and 0.77 respectively.

Ken et al. (Christensen et al. 2014) investigated three tiers of implicit occupant estimation that reuses existing infrastructure in buildings. The first tier involves the sole use of existing infrastructure such as Dynamic Host Control Protocol (DHCP) and Address Resolution Protocol (ARP) of network infrastructures to determine occupant counts. The second tier involves the augmentation of existing building infrastructure with an additional software component. For example, the Simple Network Management Protocol (SNMP) can be used to extract occupant activities such as typing. Also, additional software can be installed on host devices to obtain occupant location by laterating the signal strength of multiple Access points (AP). The third tier involves both the addition of dedicated sensors and software to existing building infrastructure. Ken et al. compared the accuracies of two implicit sensing - PC activity and DHCP with occupant estimates from PIR sensors to estimate occupant vacancy and presence. Overall, the PIR sensor, PC activity, and DHCP achieved a 91, 89 and 59% accuracy respectively.

PreCount

In this section, we propose the PreCount algorithm for accurately correcting raw counts in real-time. One approach to achieving real-time error correction from raw counts is to continuously learn the patterns and trends of count errors using machine learning approaches. We introduce the following definitions and notations to distinguish the types of datasets used in this work:

Transitions (ΔC) represents the difference between the entries and exits for each time-step in a dataset.

Occupant Count (CC) is the cumulative sum of ΔC from the first time-step of a given day to the last time-step of that day.

We differentiate ΔC and CC for raw counts in the past, raw count corrections in the past and raw counts in real-time with subscripts r, p and rt respectively. Hence, we compute the errors of the transitions and occupant counts in our past datasets as follows:

$$\Delta C_e = \Delta C_p - \Delta C_r \tag{1}$$

$$CC_e = CC_p - CC_r \tag{2}$$

PreCount employs a supervised machine learning approach to correct counts in real-time with the assumption that CC_e and error rates are repeatable over time. We validate this assumption in later subsections. To achieve this aim, PreCount applies a number of steps that are highlighted in Fig. 2, and we refer to the elements in the figure using their respective numbers. The rounded rectangles in the figure are the elements applied to the presented datasets, and the squared rectangles represent data. PreCount is comprised of both a training and a correction phase, and it assumes that counting cameras are available to provide real-time count data (CC_{rt}) for the correction phase. In the training phase, PreCount assumes that a training dataset comprising of both CC_r and their corresponding CC_e are available in a repository. In the case where there are no CC_e datasets, the past

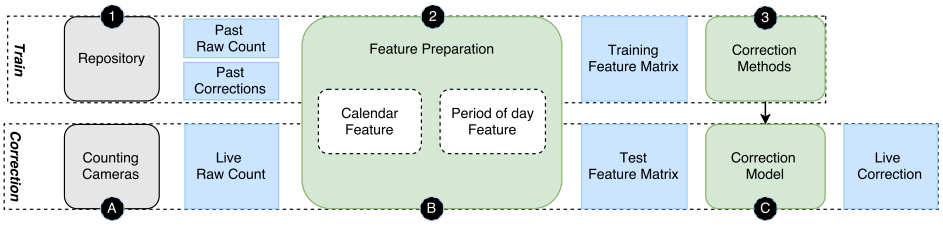

Fig. 2 Overview of the PreCount algorithm

correction method introduced in Sangoboye and Kjærgaard (2016) is used to generate CC_p dataset, and it is subsequently used to compute CC_e as specified in Eq. 2 (1). Subsequently, the feature preparation stage transforms the calendar and weather datasets into feature sets. This transformation is done alongside the CC_r dataset and the CC_e dataset to formulate a training feature matrix (2). We provide a definition and justification of the calendar and weather feature sets in later subsections. After the feature preparation stage, PreCount deploys a number of regression algorithms for correcting CC_{rt}. These regression algorithms are trained and evaluated with the training feature matrix from the feature preparation stage, and the best performing regression model is adopted as the correction model (3). In the correction phase, PreCount retrieves CC_{rt} from the installed counting cameras (A) and it applies the same feature preparation step in the training phase to obtain the test feature matrix (B). The selected correction model derived from (3) is used to estimate the inherent count errors in CC_{rt}. Lastly, we use the resulting error estimates to correct CC_{rt} (C).

In the following subsections, we present the details of the individual elements in the proposed algorithm. Here, we proceed with a particular focus on the feature preparation stage and the description of the regression algorithms deployed in PreCount. In subsequent subsections, we firstly introduce the datasets used in this work. Secondly, we provide the justification for employing a machine learning approach to correct count errors in real-time. Lastly, we highlight the rationale for including both the calendar and weather features to formulate a feature matrix.

Feature preparation

The feature preparation stage entails the representation of all features sets analyzed in the feature analysis stage alongside CC_r, CC, and CC_{rt}. This representation is used to obtain a feature matrix comprising of both an input feature sets and a target feature sets. The feature matrix is formulated as a multi-label and multi-output problem where each label in the input feature set and the target feature set can assume a real value.

The feature preparation stage assumes that all count datasets i.e. the CC_r, CC, and the CC_{rt} datasets have the same temporal resolution. Given that this requirement is met, these datasets are subdivided into daily profiles d_j such that they are comprised of $\{d_0, \ldots, d_m\}$ days and each day d_j is comprised of time slots $\{s_0, \ldots, s_n\}$. Given this formulation, the CC_r, CC_e and the CC_{rt} datasets are transformed into matrices with axes (d, s). Algorithm 1 differentiate the feature preparation process for the training phase, and the correction phase and in the following, we present the input feature sets and target feature sets for both the training and correction phase.

Algorithm 1 PreCount Feature Preparation

Input: Cumulative count dataset CC_r and CC_p or CC_{rt}

Output: Feature matrix

Determine calendar features for CC_r or CC_{rt}

Obtain all daily profile d_j in CC_r or CC_{rt} and features

day name $dn_j = \begin{cases} 0, & \text{if} d_j == Sunday \\ 1, & \text{if} d_j == Monday \\ \ldots, & \\ 6, & \text{if} d_j == Saturday \end{cases}$

day type $dt_j = \begin{cases} 0, & \text{if} d_j == Weekday \\ 1, & \text{if} d_j == Weekend \end{cases}$

holiday $h_j = \begin{cases} 0, & \text{if} d_j == Holiday \\ 1, & \text{if} d_j == Nonholiday \end{cases}$

seasons $we_j = \begin{cases} 0, & \text{if} d_j == Winter \\ 1, & \text{if} d_j == Spring \\ 2, & \text{if} d_j == Summer \\ 3, & \text{if} d_j == Autumn \end{cases}$

Determine period-of-day features for CC_r or CC_{rt}

Obtain the timeslots s_t in each d_j and feature

period-of-day $pd_{(d_j,s_t)} = \begin{cases} 0, & \text{if} s_t == Dawn \\ 1, & \text{if} s_t == Sunrise \\ 2, & \text{if} s_t == Noon \\ 3, & \text{if} s_t == Sunset \\ 4, & \text{if} s_t == Night \end{cases}$

Formulate feature matrix

Determine the current time slot s_k

if Training phase **then**

    Determine count error $CC_e = CC_p - CC_r$

    Derive input feature set $= \{dn_j, dt_j, h_j, we_j\} \cup \{pd_{(d_j,s_0)}, \ldots, pd_{(d_j,s_k)}\} \cup$

    $\{CC_{r_{d_j,s_0}}, \ldots, CC_{r_{d_j,s_k}}\}$

else if Correction Phase **then**

    $CC_e = unknown$

    Derive input feature set $= \{dn_j, dt_j, h_j, we_j\} \cup \{pd_{(d_j,s_0)}, \ldots, pd_{(d_j,s_k)}\} \cup$

    $\{CC_{rt_{d_j,s_0}}, \ldots, CC_{rt_{d_j,s_k}}\}$

end if

Derive target feature set $= \{CC_{e_{d_j,s_0}}, \ldots, CC_{e_{d_j,s_k}}\}$

Return feature matrix = {Input feature set} ∪ {Target feature set}

Input feature set

The input feature set of the feature matrix is comprised of the calendar feature, the period of feature and all the raw count data i.e. CC_r in the training phase or CC_{rt} in the correction phase. The time slots of the obtained raw count data range from the first time

slot of the current day till the current time slot. The input feature set is formulated as follows:

1. determine the values of all calendar feature such as day name (dn) - {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}, day type (dt) - {Weekday, Weekend}, holiday (h_j) - {holiday, non-holiday} for each d_j in CC_r or CC_{rt}. These features are represented as follows: {0, 1, 2, 3, 4, 5, 6 }, {0, 1} and {0, 1} respectively.
2. determine the period of day features (pd) of each time slot (d_j, s_t) in CC_r or CC_{rt} to obtain the values $\{pd_{d_0,s_0}, \ldots, pd_{d_m,s_s}\}$ where each pd_{d_j,s_t} can be one of these five values {Dawn, Sunrise, Noon, Sunset and Night}. To further differentiate the period of day features according to seasons, we include the seasonal values (we) of each day d_j in CC_r or CC_{rt}. We differentiate for all the seasons of the year {Winter, Spring, Summer, Autumn} and we represent these features as follows: {0, 1, 2, 3, 4} and {0, 1, 2, 3} respectively.

Given this formulation, the input feature set is comprised of all the aforementioned features and, raw count data $\left(\left\{CC_{r_{d_j,s_0}}, \ldots, CC_{r_{d_j,s_k}}\right\}\right.$ or $\left.\left\{CC_{rt_{d_j,s_0}}, \ldots, CC_{rt_{d_j,s_k}}\right\}\right)$. Where s_k is the current time slot, and s_0 is the time slot at the beginning of the day.

Target feature set

The target feature set in the training phase is comprised of the CC_e dataset $\left\{CC_{e_{d_j,s_{k-q}}}, \ldots, CC_{e_{d_j,s_k}}\right\}$ where k is the current time slot and q is the number of time-steps to the past. Correcting CC_{rt} from $k-q$ time slot to k is based on the rationale that the occupant count at each time slot is partially dependent on the occupant counts from previous time slots (Sangogboye and Kjærgaard 2017).

In this paper, we have varied the value of q based on the temporal granularity of a dataset. The value of q is computed as specified in Eq. 3. In this equation, the parameter *horizon* is the recommended estimation look-ahead for an occupant model with a default value of 180 min as specified in Sangogboye and Kjærgaard (2017). In the correction phase, the values of CC_e is unknown (?), and it is the value we seek to determine using the selected correction model in the training phase.

$$q = \frac{horizon}{temporal\ granularity}(in\ minutes) \tag{3}$$

Table 2 presents an example feature matrix that is comprised of both an input feature set and a target feature set. In the example feature matrix, the CC_{rt} dataset is represented in feature vector $d_j = Today$. The remaining prior feature vectors are the training feature vectors.

Table 2 Example of a PreCount feature matrix divided into input and target feature sets

Days	Input feature set											Target feature set			
	Calendar				Period of day				Raw count			Count error			
d_j	dn_j	dt_j	we_j	h_j	pd_{d_j,s_0}	pd_{d_j,s_1}	…	pd_{d_j,s_k}	$CC_{r_{d_j,s_0}}$	…	$CC_{r_{d_j,s_1}}$	$CC_{r_{d_j,s_k}}$	$CC_{e_{d_j,s_{k-10}}}$	$CC_{e_{d_j,s_{k-9}}}$	…	$CC_{e_{d_j,s_k}}$
2016-08-28	0	0	3	0	0	0	…	3	0	…	57	12	0	-12	…	11
2016-09-01	1	1	3	0	0	4	…	3	-2	…	65	32	-7	0	…	2
2016-10-18	2	1	0	1	0	4	…	4	5	…	88	25	4	2	…	23
Today	3	1	0	1	0	4	…	3	-3	…	53	27	?	?	…	?

Correction methods

In this section, we present two regression algorithms for estimating count errors in CC_{rt}. These regression algorithms include Random Forest (RFR) and AdaBoost Decision Tree (ADR). Both regression algorithms are trained using the derived feature matrix comprising of both the input and target feature sets. Subsequently, a regression model for predicting count error CC_e in CC_{rt} is derived from the training process. In the following, we provide the rationale for selecting these methods.

RFR

RFR is an ensemble method that utilizes decision trees as its primary estimators. This ensemble method sub-samples the presented training feature matrix into equal partitions, and for each partition, it fits an estimator. Given a test sample, each of the trained models, is used to produce some target estimates. Subsequently, all target estimates are averaged to produce the final target estimate. One essential tuning parameter for RFR is the number of estimators it utilizes to fit the training feature matrix. In this paper, we have utilized ten estimators to fit the training feature matrix. This is the same number of estimators specified in Pedregosa et al. (2011a). This number of estimators is assumed to avoid overfitting issues and to achieve model generalization.

ADR

An AdaBoost regressor is a meta-estimator (Pedregosa et al. 2011b) that fits multiple instances of the same regression algorithm to a training dataset. This is achieved by first fitting a single regression method to the entire dataset, and subsequently, it fits additional regression methods on the same dataset where the previous regression method are less accurate. Similar to RFR, AdaBoost ensemble enables the declaration of the number of estimators to be used in the learning procedure. However, unlike the RFR where all of the declared estimators are fitted to a subset of the training dataset, ADR performs incremental training that terminates when there is a perfect fit. Hence, the specified number of estimators only provides an upper limit of the number of estimators that can be used before the learning procedure is terminated. In this paper, we have chosen the same decision tree method as the base estimator for the AdaBoost meta-estimator hence the acronym ADR. Also, we have utilized the default number of estimators of 50 presented in Pedregosa et al. (2011b) to fit the training feature matrix.

Lastly, the rationale for choosing the decision tree regression method as the base estimator is based on its ability to discover complex dependencies between features and its strength in modeling both linear and non-linear relationships in a feature matrix. Also given that the problem of correcting the count errors CC_e in the obtained real-time count CC_{rt} is formulated as a multi-label regression problem where each target label can assume a real number, these correction models are utilized in this form to predict and correct the values of these labels. These labels, will enable a varying number of CC_{rt} tuples to be corrected as against the case of a single-label regression problem, where users are constrained to a single output.

Building cases and datasets

In this study, we have obtained datasets from four commercial building cases. We utilize the datasets from three of the building cases to concretize PreCount's design assumptions,

training, and evaluation. The dataset from the last building case is used to test and validate PreCount. The four building cases are a university teaching building (UNIVERSITY), a public library (LIBRARY), a shopping mall (MALL) and a research building (OFFICE). The datasets obtained from these buildings cover different ways commercial buildings are occupied and used by people. Each dataset contains both ΔC and CC of both the raw counts and their corresponding past corrections respectively. The raw count correction data in the past are obtained through the probabilistic method in Sangoboye and Kjærgaard (2016).

We have obtained 1-year worth of dataset from the first three building cases and 6-month worth of dataset from the OFFICE building. The evaluation of PreCount is done with this 1-year datasets. The evaluation uses the past correction data from the probabilistic approach as ground truth data. This is because of the cumbersome process involved in manually collecting ground truth data, especially in large commercial buildings. More so, the potency of obtaining high fidelity count estimates through the probabilistic approach provides a viable alternative for benchmarking subsequent methods such as a method for correcting erroneous count estimates in real-time (Sangoboye and Kjærgaard 2016). However, it should be noted that these datasets are only used for analysis, training and evaluation purposes and not for testing and validation. Additional, we have obtained 1-day of manually collected ground truth data from the OFFICE building. The 6-month past count data from this building and the manually collected ground truth data are used to validate and test PreCount. In the following, we provide a detailed description of the four building cases.

The UNIVERSITY is an $8000m^2$ building that records an average of 800 to 900 occupants on normal weekdays. The building is primarily a teaching building with some office spaces. The types of room in this building are comprised mainly of classrooms, study zones, offices, and restrooms. To obtain the raw count of occupants in this building, 17 Stereo-vision cameras are installed to cover all transitions between all entrances and exits. The cameras installed in this building are manufactured by Xovis. All the datasets from the UNIVERSITY are obtained at a temporal granularity of 1 min.

The MALL is a $36{,}000m^2$ building containing 80 commercial spaces or shops. To obtain the raw counts of occupants in this building, 22 Xovis Stereo-vision cameras are installed to cover all transitions between the entrances and exits of the building. This building records an average of 4500 to 5000 occupants on weekends and 3500 to 4500 occupants on weekdays. All the datasets from the MALL are obtained at a temporal granularity of 15 min.

The LIBRARY is a $60{,}000m^2$ building that accommodates a library and business spaces. It records an average of 800 to 1000 occupants on weekdays and 400 to 600 occupants on weekends. To obtain the raw count of occupants in this building, 7 network cameras are installed to cover all transitions between the entrances and exits of the building. The cameras installed in this building are manufactured by AXIS communications and are coupled with a software module namely Cognimatics TrueView People Counter to dynamically recognize people entering and exiting the building. All raw counts are publicly available in Jensen (2016), and the datasets from this building are obtained at a temporal granularity of 1 h.

The OFFICE is a $2500m^2$ building that records an average of 70 to 80 occupants on normal weekdays. The building is primarily an office building that hosts a number of

university researchers. Thus it is mainly comprised of offices, laboratories, meeting rooms and restrooms. To obtain the raw count of occupants in this building, 4 Stereo-vision cameras are installed to cover all transitions between all entrances and exits in the building. Also, the cameras installed in this building are manufactured by Xovis. All the datasets from OFFICE are obtained at a temporal granularity of 1 min.

Error patterns in raw occupant counts

In this section, we utilize datasets from the first three building cases to present our investigation of the error pattern in raw counts. As stated earlier, if there exists a consistent and repeatable error pattern, this will provide the basis for adopting a machine learning approach to accurately estimate count errors in real-time. To investigate the assumption that an error pattern exists, we reiterate the following proposition from Ihler et al. (2006), Hutchins et al. (2007), and Sangoboye and Kjærgaard (2016).

Proposition 1 *The rate of count errors in counting sensors is directly proportional to the rate of transitions in the spaces they are deployed.*

One imminent effect when the transition rate in a space increases is occupant occlusion. More specifically, it is well established that counting cameras generally have significant challenges in separating occupants that are occluded and this usually results in undercount errors. Sangogboye et al. (Sangoboye and Kjærgaard 2016) investigated this proposition by correlating ΔC_r and ΔC_e. The resulting correlation indicates a linear relationship between ΔC_r and ΔC_e. This linear relationship implies that a general increase in ΔC_r will usually result in an increase in ΔC_e. Similarly, in this paper, we correlate the rate of ΔC_r with ΔC_e for the first three building cases, and the obtained results show similar patterns as specified in Sangoboye and Kjærgaard (2016).

Figure 3 highlights the correlation results for the UNIVERSITY building case. From this figure, it can be observed that in the x-axis, we correlated transition bins instead of the ordinary ΔC_r against ΔC_e. We borrowed this concept of binning from histogram charts, and this enables us to group the sparse range of transition events in buildings. Here, we apply a bin size of 10 which implies that bin 0 contains all absolute transitions ranging from 0 to 9 while bin 1 contains all absolute transition events ranging from 10 to 19 and so on. The Pearson correlation coefficient for both variables is 0.9, and this corroborates the linear relationship observed in Sangoboye and Kjærgaard (2016).

The result from Fig. 3 corroborates the finding in Sangoboye and Kjærgaard (2016) that count errors follow a linear pattern. The repeatable patterns. Provide the basis for training a machine learning task.

Feature analysis

The adoption of a machine learning methodology for any given task requires the identification of well-defined feature sets that can characterize both the explicit and implicit patterns within a problem domain. Within the contest of real-time error correction, such feature sets should represent the factors that influence the propagation of count errors. In the following, we present our investigation of two categories of feature sets namely the calendar and weather feature sets that influences the propagation of CC_e in buildings. We justify the inclusion of these feature sets because of the high variability they display with

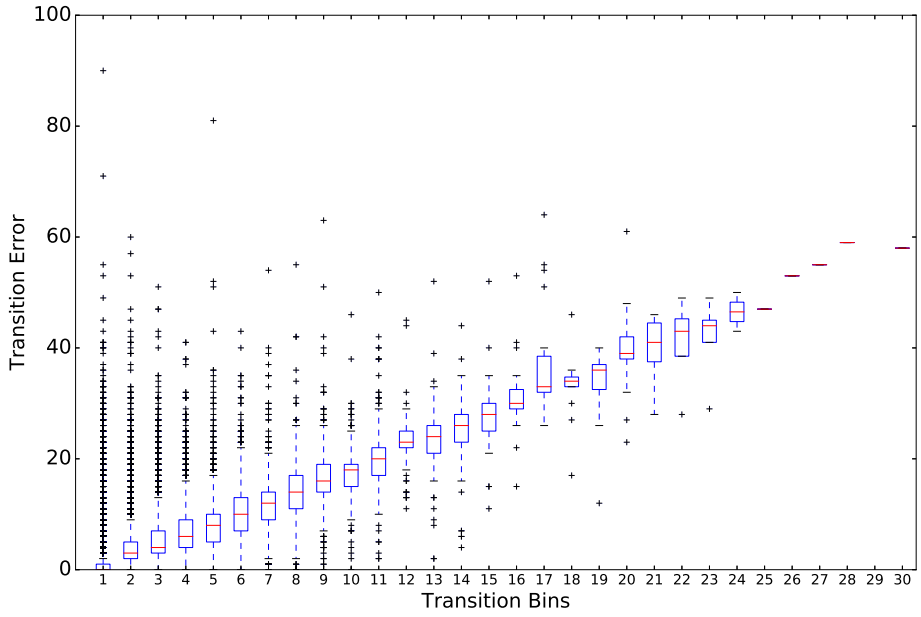

Fig. 3 Correlation between transition bins (bin width = 10 counts) and absolute transition errors

regards to CC_e Sangoboye and Kjærgaard (2016) and Shan et al. (2003). These variabilities were investigated by correlating count datasets with the values of these feature sets.

Calendar features

Occupant patterns in buildings vary across different types of days. Figure 4 highlights the occupant pattern on weekends and weekdays for the UNIVERSITY dataset. These patterns were derived by computing the mean of all daily profiles in CC_p with a 95% confidence interval. Following the proposition in 1, the magnitude difference in both occupant patterns indicates that the error rates associated with these patterns will vary significantly. Also, we hypothesize that since each day of a week may be characterized by different occupant schedules and that more patterns may exist for other day types such as holidays, we

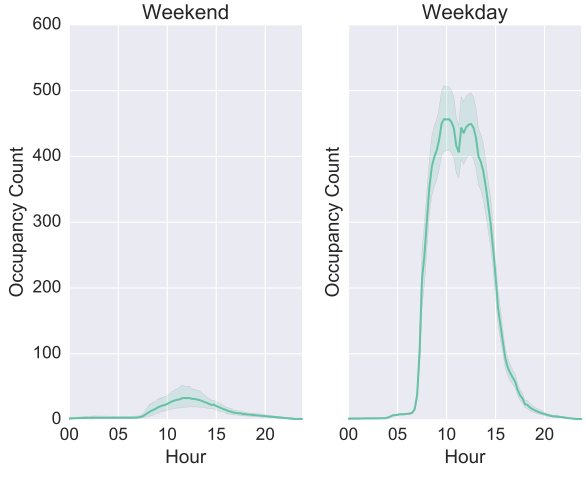

Fig. 4 Occupant pattern on weekends vs weekdays

argue that the features set for correcting CC_e in real-time should include such calendar features to navigate the different occupant patterns that may arise thereof.

Weather features

Stereo-vision, thermal and other camera technologies are significantly susceptible to poor lighting conditions (Shan et al. 2003). For instance, (Shan et al. 2003) indicated that this may be as a result of the predominant high luminance during summer and spring seasons, and the predominant poor luminance during autumn and winter seasons for the standard European weather condition. We utilize two independent datasets namely the period of a day and seasonal variation to investigate how the varying luminance conditions affect counting cameras.

To quantify the influence of the period of the day on count errors, we extract ΔC_p and ΔC_e datasets associated with the five (5) periods of the day namely:

Sunset periods between the time in the evening when the sun is about to disappear below the horizon, and the time it disappears

Noon periods between the time when the sun is at its highest point and the beginning of sunset

Sunrise periods between the time in the morning when the top of the sun breaks the horizon and the beginning of noon

Dawn periods between the time in the morning when the sun is at a specific number of degrees below the horizon and the beginning of sunrise

Night periods between the astronomical dusk of one day and the astronomical dawn of the next

The period of day dataset was obtained using a Python library given in Kennedy (2010), and these datasets are obtained at the same resolution as the count datasets and for the same coverage period.

To obtain a unified view of how both the ΔC_p and ΔC_e datasets that are associated with each period of the day, we compute the following normalization.

$$\Delta\hat{C}_e(a) = \frac{\sum_{j=1}^{m} |x_{aj}|}{\sum_{i=1}^{n}\sum_{j=1}^{m} |x_{ij}|} \tag{4}$$

where x_{ij} denotes the j-th element of i-th period of the day, x_{aj} represents the j-th element of a period of the day and $\Delta\hat{C}_e(a)$ is the normalized dataset of ΔC_e for the a-th period of the day. The value of m may differ depending on the number of elements in $\Delta C_e(a)$. The same normalization is performed on the ΔC_p dataset. Subsequently, we computed the error rate for each period of the day from the normalized datasets as follows:

$$ErrorRate = \frac{\Delta\hat{C}_e}{\Delta\hat{C}_p} * 100\% \tag{5}$$

Figure 5 highlights this correlation for the dataset obtained from the UNIVERSITY building case. From Fig. 5, it can be observed that both the autumn and winter seasons have similar patterns for the same periods, while for both the spring and summer seasons, only the sunrise and noon periods are similar. Similarly, the error rates for the winter, summer, and autumn seasons are at the highest during the night periods. However, during spring,

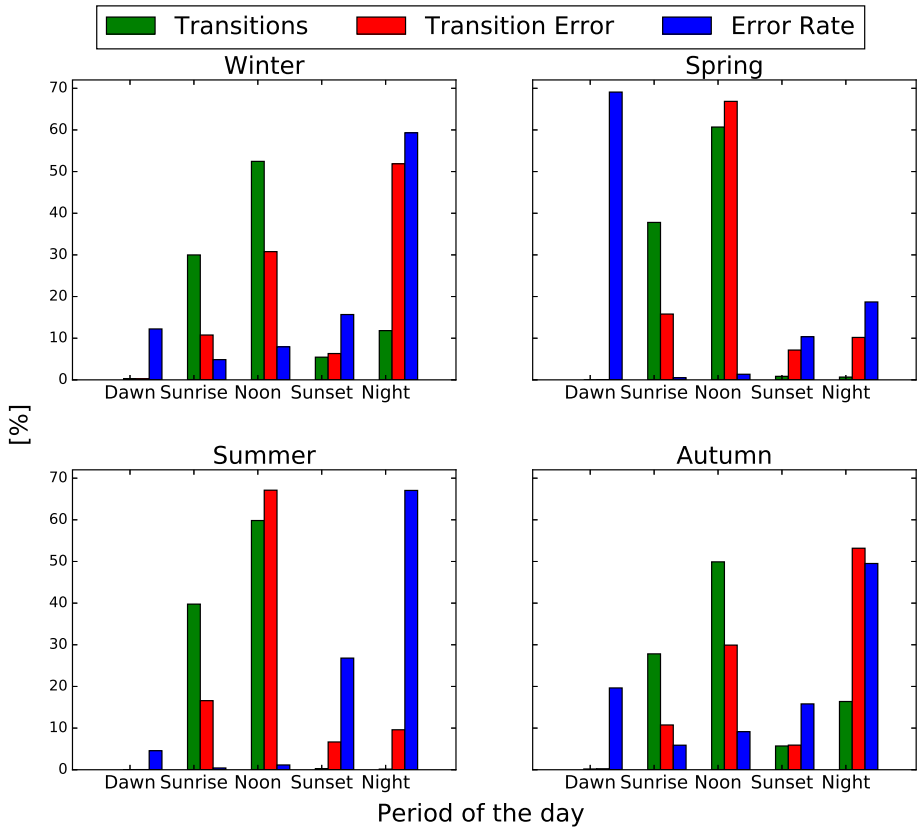

Fig. 5 Correlation between periods of the day, normalized transition and transition errors

the error rates are at the highest at dawn. Figure 5 also shows some distinctiveness compared to Fig. 3. The noon period of the day has the highest proportion of all transitions for all seasons. However, the error rates during this period are low compared generally to dawn, sunset and night periods of the day. This indicates that while more accumulated ΔC_e are propagated during periods with more ΔC_r, the rate at which these ΔC_e are propagated are entirely independent of the rate of ΔC_r. We argue that this variation in error rates are well captured with the period of the day and seasonal variation feature sets.

Evaluation

In this section, we present an evaluation of the correction methods deployed in PreCount. The correction methods are evaluated using the Normalized Root Mean Square Error (NRMSE) metric, and a Leave-One-Out cross-validation method was adopted to ensure a comprehensive evaluation process. This evaluation method exhaustively evaluates the performance of each correction method such that for each evaluation step, it uses $n - 1$ samples in a feature matrix for training and 1 sample for testing. Thus the number of evaluation cases equals the number of feature vectors (samples) in the feature matrix. We favor NRMSE over other standard metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE) because RMSE can only be interpreted with prior knowledge of the duration of occupant detection while NRMSE is a non-dimensional form of RMSE. In the case of MAPE, we favor NRMSE because the occupant counts in a building

at a time stamp can be zero, especially at night times and we want to avoid division by zero. The NRMSE is derived from the root mean squared error (RMSE) as follows:

$$RMSE = \sqrt{\frac{\sum_{i=0}^{o}\left(CC_{g(i)} - CC_{rtc(i)}\right)}{o}} \tag{6}$$

$$NRMSE = \frac{RMSE}{\overline{CC_g}} \tag{7}$$

where $CC_{g(i)}$ and $CC_{rtc(i)}$ are ground truth occupant counts and real-time corrected counts respectively.

For each presented building cases and their datasets, we have utilized the entire training data to benchmark the performance of the presented correction methods. Additionally, we utilized the 1-day ground truth data from OFFICE to validate PreCount's correction methods. We also present an investigation into correction cases where few datasets are available for training and we highlight how the correction results vary given these variations in the size of training datasets.

PreCount and its elements are implemented in Python using some Python data processing (Jones et al. 2001) and machine learning libraries (Pedregosa et al. 2011c).

In the following we present the evaluation cases used to evaluate the performance of PreCount:

1. We compare the overall performance of PreCount's correction methods for each building case and present the most accurate correction method as PreCount. In this evaluation case, we benchmark all methods with CC_p and we compare the correction methods with both the CC_{rt} and the naive approach.
2. We compare the performance of PreCount's correction methods in the OFFICE building case for the 1-day ground truth data. In this evaluation case, we benchmark all the methods for correction with the ground truth data, and we compare the performance of PreCount's correction methods with the count estimates from the naive approach, CC_p, and CC_{rt}.
3. Lastly, we evaluate the performance of PreCount with different sizes of training datasets. For this evaluation, we varied the sizes of training datasets from 30 to 120 days. This evaluation is especially crucial in-order to investigate how the PreCount algorithm will perform during periods of early deployment where limited training datasets are available.

Results

Overall performance

Figures 6, 7, and 8 highlights the overall performance of PreCount's correction methods (ADR and RFR) compared to the naive approach and CC_{rt} in the UNIVERSITY, MALL and LIBRARY building cases respectively. From these figures, the RFR correction achieved better accuracy than the ADR correction method, while both methods achieved better accuracy than the naive approach. For the UNIVERSITY building case, the ADR, RFR and naive correction methods achieved an NRMSE of 0.85, 0.62 and 1.53 while CC_{rt} achieved an NRMSE of 1.60 for all test days in the count dataset. This indicates that PreCount's correction methods achieved a 60% error reduction for the UNIVERSITY building case compared to the naive approach and CC_{rt}. For the MALL building case, the ADR, RFR and

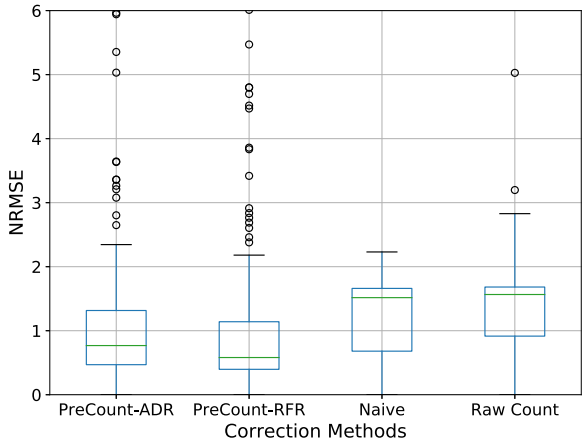

Fig. 6 Boxplot showing the performance of each correction method for UNIVERSITY building case

naive correction method achieved an NRMSE of 0.09, 0.05 and 0.12 respectively while the CC_{rt} achieved an NRMSE of 0.15 for all test days in the count dataset. This indicates that PreCount's correction methods achieved a 58% error reduction for the MALL building case. Lastly, for the LIBRARY building case, the ADR, RFR and the naive correction methods achieved an NRMSE of 0.15, 0.14 and 0.17 respectively while CC_{rt} achieved an NRMSE of 0.22 for all test days in the count dataset. This indicates that both regression methods achieved a 36% error reduction for the LIBRARY building case.

In all the building cases, the RFR model achieved better error reduction than the RFR model. This is because, the perfect fit of the Adaboost meta-estimator may have overfitted the correction model such that it is less generalization to new correction cases.

Ground truth result

Figure 9 highlights the performance of PreCount's correction methods (ADR and RFR), compared to the naive approach, CC_{rt}, and CC_p for the 1-day ground data in the OFFICE building case. For this evaluation case, the ADR, RFR, CC_p, and the naive approach achieved an NRMSE of 0.11, 0.10, 0.12 and 0.31 while the CC_{rt} achieved an NRMSE of 0.31.

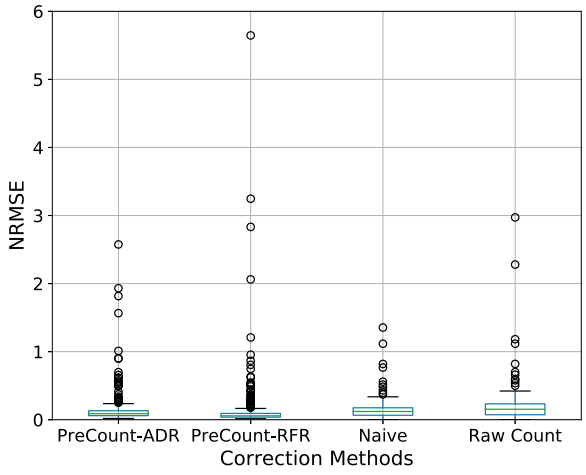

Fig. 7 Boxplot showing the performance of each correction method for MALL building case

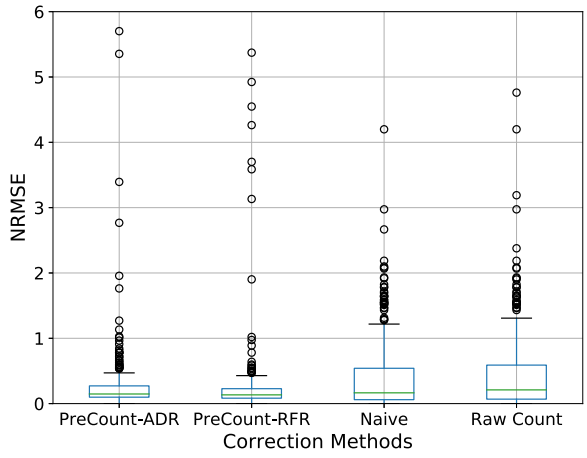

Fig. 8 Boxplot showing the performance of each correction method for LIBRARY building case

This indicates that the PreCount correction method achieved a 68% error reduction compared to the naive approach and CC_{rt} and, a slight improvement over CC_p. Also from this evaluation, it can be observed that none of PreCount's correction method violates the requirements stipulated by the United States Department of Energy (US Department of Energy 2017).

Comparison between building cases

In this section, we discuss the variation in the results obtained from all building cases. In Table 3, we summarize the performance of all correction methods for datasets obtained from all building cases. From this table, it can be observed that the error reduction

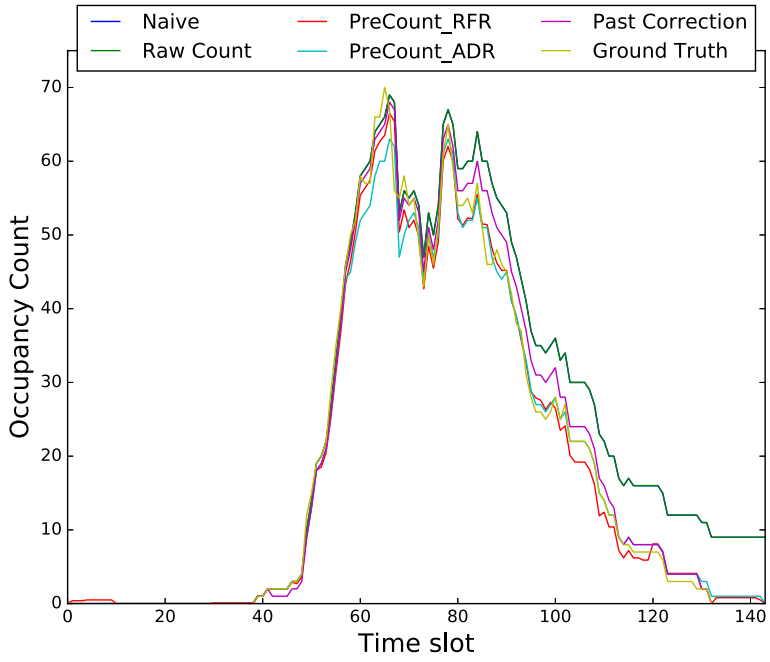

Fig. 9 Performance of each correction method in the OFFICE building case for 1-day ground truth data

Table 3 A table comparing the evaluation results for all building cases

Correction methods	Building cases			
	UNIVERSITY	MALL	LIBRARY	OFFICE
Raw count	1.60	0.15	0.22	0.31
Naive	1.53	0.12	0.17	0.12
PreCount_RFR	0.62	0.05	0.14	0.10
PreCount_ADR	0.85	0.09	0.15	0.11
Sampling rate (Minutes)	1	15	60	1
Error reduction(%)	60%	58%	36%	68%

from the UNIVERSITY, MALL, and OFFICE buildings are more significant compared to the LIBRARY building case. The primary cause of this disparity is the rate at which the count datasets are a sample from the camera sensors. For instance in LIBRARY building case, PreCount has fewer points to corrects over the course of the evaluation period because of the high sampling rate compared to other building cases. Also, the aggregation over a longer sampling period reduces the variability in the dataset, and such variability will usually facilitate the training of a more robust regression model.

Training dataset size

The results in the previous section have used the entire training dataset in the training phase of PreCount. However, there are instances where there are limited training data such as cases where counting devices have recently been deployed. For these cases, we investigate the accuracy of PreCounts, and we hypothesize that increasing the size of the training dataset should increase the accuracy of PreCount.

In Fig. 10, we varied the size of the training dataset for the MALL building case from 30 days to 120 days with a unit step of 30 days, and we employ a 95% confidence interval for the NRMSE. Also, we compared the NRMSE of CC_{rt} with the NRMSE of varying size

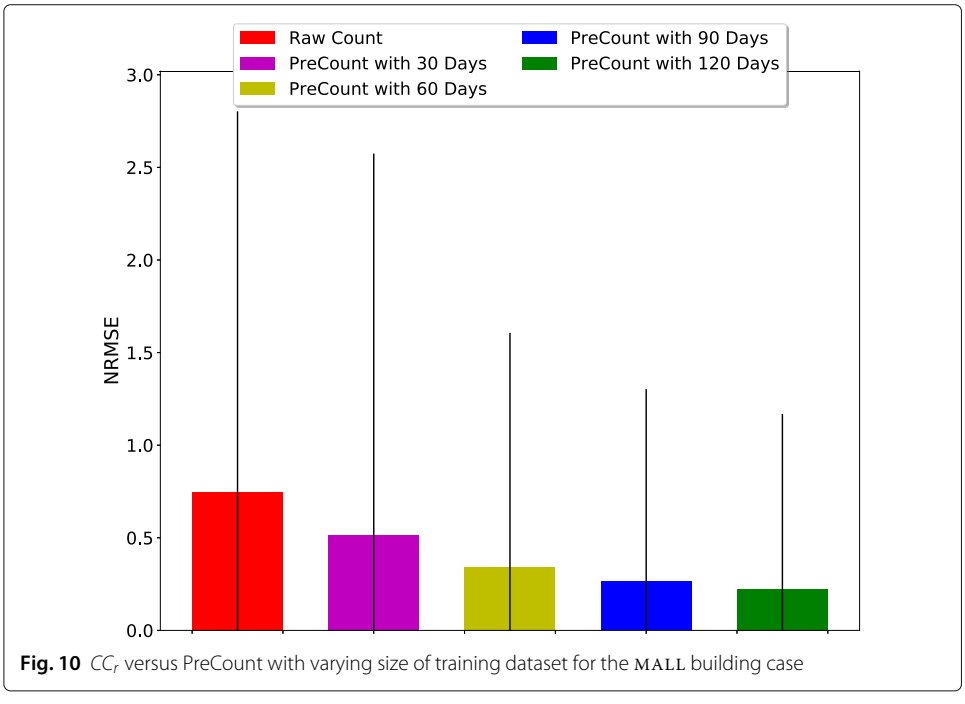

Fig. 10 CC_r versus PreCount with varying size of training dataset for the MALL building case

of the dataset. From Fig. 10, training datasets with sizes 30, 60, 90 and 120 days achieved an NRMSE of 0.52, 0.34, 0.26 and 0.22 respectively compared to CC_{rt} that has an NRMSE of 0.74. It can be observed that as the size of the training dataset increased, the NRMSE reduced i.e. the accuracy of PreCount increased over time. Also, with just 30 days of training data, PreCount achieved over 30% increase in accuracy compared to CC_{rt}.

Discussion

The evaluation results presented in the four evaluation cases documents the merit of PreCount for performing count correction in real-time. From the overall performance in "Overall performance" section, PreCount achieved a 60%, 58% and 36% error reduction in the UNIVERSITY, MALL and LIBRARY building respectively. The ground truth result in "Ground truth result" section shows a 68% count error reduction in an OFFICE building. With the results obtained from the varying training sizes of the dataset in the "Training dataset size" section, PreCount could achieve as high as 30% error reduction with just 30 days of the training dataset. While the accuracies obtained from PreCount are significant, these accuracies are achieved within the range of some of the highlighted feature sets in the "Feature analysis" section. Some of these feature sets were motivated from previous literature and are known contributors to erroneous counts, especially in camera technology. However, no previous literature enabled real-time count correction as enabled by PreCount. In this section, we discuss some of the limitations in this work, and we highlight some improvement opportunities to achieve better error detection and correction accuracy.

Firstly, PreCount assumes that there are available datasets for formulating the feature sets described in the "Feature analysis" section. However, in some locations, such datasets are not readily available. To address this concern, it could be relevant to perform a feature ranking analysis alongside the feature correlation for all presented feature sets. This can be done in the training phase of PreCount with methods such as principal component analysis (PCA) or single value decomposition (SVD) to determine what correction accuracy is expected with the inclusion or exclusion of each feature group.

Also in Fig. 10, we presented that increasing the size of the training dataset increases correction accuracy. However, it should be noted that as the size of training datasets increases, the time complexity for generating PreCount correction models also increases linearly. Therefore, a trade-off that maximizes both the accuracy for estimating count errors and estimation speed should be investigated to ensure that the latency for correcting real-time counts are minimal for each domain application.

Conclusions

In this paper, we present PreCount - a predictive model for correcting real-time occupant count data. We highlighted the non-adaptability of methods for correcting erroneous occupant counts in the past to correct occupant counts in real-time. We reviewed the accuracy of the methods for correcting occupant counts in the past and how PreCount leverages the accuracy of the probabilistic approach to accurately produce a real-time correction. Secondly, we present PreCount's elements, and we deployed two correction methods namely RFR and ADR in PreCount. Also, we present a "Feature analysis" section that highlights the features that can represent the factors influencing erroneous occupant counts in our real-time correction model. These factors were used for formulating

a real-time count correction problem. Thirdly, we evaluate the performance PreCount's correction methods using three evaluation cases and with datasets from four building cases. The first two evaluation cases benchmark the overall performance of both correction methods with CC_p dataset from the first three building and ground truth data from the last building respectively. The third evaluation case evaluates the performance of PreCount with varying sizes of training data. Lastly, we present the results from all the evaluation cases. The results from the first evaluation case indicate that RFR outperforms ADR in all building cases. In the second evaluation case, both correction methods achieved a maximum error reduction of 68% compared to CC_{rt} and the naive approach and, a slight improvement over CC_p. And in the third evaluation case, PreCount achieved an increasing accuracy as the training dataset increases and with just 30 days of training data, PreCount achieved an error reduction of over 30% when compared to raw counts. From the foregoing, PreCount can reliably produce high fidelity correction of occupant counts in real-time. Also, PreCount can achieve significant error reduction when limited training data are available for training.

In conclusion, given that PreCount achieves high fidelity correction of CC_{rt}, PreCount can facilitate a number of pervasive and real-time applications that spans several domains such as disaster prevention and management, building energy management, and queue management in retail stores.

Funding
This work is supported by Innovation Fund Denmark for COORDICY (4106-00003B).

Availability of data and material
Please contact author for data requests.

Authors' contributions
FCS carried out the implementation and design of the occupant estimation algorithm and drafted the manuscript. MBK assisted in formulating the idea of the study, participated in the design of the algorithm and helped to draft the manuscript. All authors read and approved the final manuscript.

Competing Interest
The authors declare that they have no competing interests.

Publisher's Note

Received: 26 March 2018 Accepted: 2 August 2018
Published online: 23 August 2018

References

Arendt K, Ionesi A, Jradi M, Singh AK, Kjærgaard MB, Veje C, Jørgensen BN (2016) A building model framework for a genetic algorithm multi-objective model predictive control. In: 12th REHVA World Congress CLIMA 2016. Aalborg University, Department of Civil Engineering, Aalborg

Beltran A, Erickson VL, Cerpa AE (2013) Thermosense: Occupancy thermal based sensing for hvac control. In: ACM, BuildSys'13. ACM, NY, USA. pp 11:1–11:8

Caucheteux A, Sabar AE, Boucher V (2013) Occupancy measurement in building: A litterature review, application on an energy efficiency research demonstrated building. Int J Metrol Qual Eng 42:135–144

Chang W-K, Hong T (2013) Statistical analysis and modeling of occupancy patterns in open-plan offices using measured lighting-switch data. Build Simul 6:23–32

Christensen K, Melfi R, Nordman B, Rosenblum B, Viera R (2014) Using existing network infrastructure to estimate building occupancy and control plugged-in devices in user workspaces. Int J Commun Netw Distrib Syst 12(1):4–29

Ekwevugbe T, Brown N, Pakka V, Fan D (2013) Real-time building occupancy sensing using neural-network based sensor network. In: Digital Ecosystems and Technologies (DEST), 2013 7th IEEE International Conference on. IEEE, Menlo Park. pp 114–119

Erickson VL, Carreira-Perpiñán MÁ, Cerpa AE (2011) Observe: Occupancy-based system for efficient reduction of hvac energy. In: Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on. IEEE, Chicago. pp 258–269

Erickson VL, Lin Y, Kamthe A, Brahme R, Surana A, Cerpa AE, Sohn MD, Narayanan S (2009) Energy efficient building environment control strategies using real-time occupancy measurements. In: Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings. ACM, New York. pp 19–24

Hutchins J, Ihler A, Smyth P (2007) Modeling count data from multiple sensors: A building occupancy model. IEEE, St. Thomas
Ihler A, Hutchins J, Smyth P (2006) Adaptive event detection with time-varying poisson processes. In: ACM, KDD '06. ACM, New York, NY, USA. pp 207–216
Jensen T (2016) Open Data aarhus. https://www.odaa.dk/dataset/taellekamera-pa-dokk1. Accessed 3 Apr 2017
Jones E, Oliphant T, Peterson P, et al. (2001) SciPy: Open source scientific tools for Python. http://www.scipy.org/. Accessed 24 Nov 2015
Kamthe A, Jiang L, Dudys M, Cerpa A (2009) Scopes: Smart cameras object position estimation system. In: EWSN '09. Springer-Verlag. pp 279–295
Kennedy S (2010) Astral python library. https://pypi.python.org/pypi/astral/0.8.1. Accessed 3 Apr 2017
Kjærgaard MB, Johansen A, Sangogboye F, Holmegaard E (2016) Occure: An occupancy reasoning platform for occupancy-driven applications. In: 2016 19th International ACM SIGSOFT Symposium on Component-Based Software Engineering (CBSE) IEEE, Venice. pp 39–48
Kjærgaard MB, Arendt K, Clausen A, Johansen A, Jradi M, Jørgensen BN, Nelleman P, Sangogboye FC, Veje C, Wollsen MG (2016) Demand response in commercial buildings with an assessable impact on occupant comfort. In: Smart Grid Communications (SmartGridComm), 2016 IEEE International Conference on. IEEE, Sydney. pp 447–452
Kuutti J, Saarikko P, Sepponen RE (2014) Real time building zone occupancy detection and activity visualization utilizing a visitor counting sensor network. In: IEEE. IEEE, Porto. pp 219–224
Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: A random forest regressor. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. Accessed 3 Apr 2017
Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: An adaboost regressor. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html. Accessed 3 Apr 2017
Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine learning in Python. JMLR 12:2825–2830
Sangoboye FC, Kjærgaard MB (2016) Plcount: A probabilistic fusion algorithm for accurately estimating occupancy from 3d camera counts. In: Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments, BuildSys '16. ACM, New York, NY, USA. pp 147–156
Sangogboye FC, Kjærgaard MB (2017) Promt: predicting occupancy presence in multiple resolution with time-shift agnostic classification. Comput Sci Res Dev:1–11
Shan S, Gao W, Cao B, Zhao D (2003) Illumination normalization for robust face recognition against varying lighting conditions. In: 2003 IEEE International SOI Conference. Proceedings (Cat. No.03CH37443) IEEE, Nice. pp 157–164
US Department of Energy (2017) Advanced Research Projects Agency Energy. Saving energy nationwide in structures with occupancy recognition (sensor). https://arpa-e-foa.energy.gov/FileContent.aspx?FileID=709732a8-c6f2-410f-95a8-21de1752e259. Accessed 13 Sept 2017