ROOM OCCUPANCY SENSING USING A THERMAL TRIPWIRE

Janis Intoy and Emily Lam

April 10, 2017

Boston University

Department of Electrical and Computer Engineering

Technical Report No. ECE-2017-01

BOSTON
UNIVERSITY

ROOM OCCUPANCY SENSING USING A THERMAL TRIPWIRE

Janis Intoy and Emily Lam

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

April 10, 2017

Technical Report No. ECE-2017-01

Here comes a special preface page. For example, if the report is restricted, then a suitable note can be included. This page can also be used to indicate to whom the document is dedicated, etc.

Summary

The efficiency of HVAC (Heating Ventilation & Air Conditioning) systems can be improved by making them adaptive to the number of people in a room. Automatic adjustments to room ventilation and temperature based on room occupancy reduces energy use and is more cost efficient. In order to a estimate a room's occupancy level, the Occusense Senior Design team has created a reliable, thermal sensor system to capture the motion of people through doorways. We aim to develop a reliable, real-time algorithm to detect the direction of motion of people passing through in order to track the number of people in the room.

Contents

List of Figures

List of Tables

1 Introduction

In modern efficient buildings, controlling the HVAC system preemptively can be a huge cost saver. Therefore, a senior design team at Boston University is working towards developing sensing technology and algorithms to count the number of people in a room so that the system is always conscious of room occupancy and can adjust system parameters, such as ventilation, accordingly before feedback sensing technology can detect abnormalities, such as rising temperatures of a crowded room. There are a number of ways to detect occupancy. However, this project assumes that if a room has a low number of entry points, counting people can be done at the entries based on who is entering or leaving the room. As such, there would be a continuous knowledge of the number of people in a room. The senior design team has implemented a privacy-inclined low resolution thermal sensor with a field of view of 30x120 positioned at the top of the door frame and looking perpendicularly down. It is capable of capturing a 16x4 pixel array at frame rates of 0.5 to 512 Hz. Using this information, our project is tasked at developing algorithms to count the number of people entering or leaving a room. Specifically, this algorithm will 1) detect the presence of a moving person in the frame and 2) determine the direction of motion.

2 Literature Review

Many background subtraction algorithms have been developed to detect changing pixels in a series of images. A brief review of the more common change detection algorithms can be found in [1]. The most successful amongst these incorporate a background model into the algorithm. The use of a model allows for thresholding of probabilities instead of pixel intensities and is therefore more robust to some variation in the background scene. In addition, foreground models can improve the sensitivity of the change detectors [2]. McHugh et al. suggested a foreground model algorithm that is more general as it is based on spatial neighborhoods. To further improve the discrimination, they also suggest a Markov model so that labels are more spatially coherent [3].

3 Problem Statement

Our approach breaks the problem into two parts: 1) detect a person in the doorway and 2) determine the direction of motion of the person.

3.1 Person Detection

We propose a background subtraction method to detect a person or persons in the frame. In order to accomplish this we make several assumptions about the acquired data. First, we assume that the background has slow temporal dynamics. Second, we assume that the data starts with some number of background-only frames. For business and residential buildings this is likely true if they close for the night. Finally, we assume that the surface temperature of a single body as detected by the sensor has the same variability as the background.

The proposed solution is inspired primarily by the foreground-adaptive background subtraction described in [3]. The background can be modeled by a probability density function estimated at each pixel location $\mathbf{n}$ from the N most recent pixels that were labeled as background:

$$P_{\mathcal{B}}\left(I^{(k)}[\mathbf{n}]\right) = \frac{1}{N} \sum_{i \in \mathcal{B}_k[\mathbf{n}]} \mathcal{K}\left(I^{(k)}[\mathbf{n}] - I^{(i)}[\mathbf{n}]\right)$$

where $I^{(k)}[\mathbf{n}]$ is the temperature at location $\mathbf{n}$, $B_k[\mathbf{n}]$ are the N previous time indices at which the pixel located at $\mathbf{n}$ was labeled background, and $\mathcal{K}$ is a zero-mean Gaussian with variance σ^2. Thresholding these probabilities with threshold θ gives an initial classification of pixels into background or foreground.

The development of the foreground model follows a similar formulation to that above, but the summation is over pixels in the neighborhood of $\mathbf{n}$ that have been already been labeled as foreground. The likelihood ratio of the probability density functions $P_{\mathcal{B}}$ and $P_{\mathcal{F}}$ can be thresholded to determine new labels for each pixel. Furthermore, the number of foreground neighbors a pixel has can be used to adjust the threshold. The more foreground neighbors a pixel has the easier it is to be labeled as threshold. This will contribute to the spatial coherency of the labels. Thus, a pixel is labeled as background if

$$\frac{P_{\mathcal{B}}(I[\mathbf{n}])}{P_{\mathcal{F}}(I[\mathbf{n}])} > \theta \exp\left(\frac{1}{\gamma}(Q_{\mathcal{F}}[\mathbf{n}] - Q_{\mathcal{B}}[\mathbf{n}])\right)$$

where $Q_{\mathcal{F}}[\mathbf{n}]$ and $Q_{\mathcal{B}}[\mathbf{n}]$ are the number of neighbors that are foreground and background respectively and γ is a parameter to control how strongly the threshold adapts. The thresholding step can be done iteratively as labels are updated in each step.

This algorithm will result in labels for each pixel in each frame as background or foreground. The number of foreground pixels in each frame could indicate whether a person is in the frame.

3.2 Direction Discrimination

Emily

4 Implementation

The current implementation of the algorithm is in Matlab. To integrate with the senior design team they would need to be ported into Python.

4.1 Data Acquisition

The senior design team has a the thermal sensor setup and given us access to record data from it so that we can continue to collect data on top of what they have already shared with us. We plan to collect a few trials of realistic but potentially difficult situations such as a person lingering in the doorway or multiple people passing through the door in quick succession.

The data is recorded into text files at a rate of 8-12Hz. The data we currently have was recorded at 8Hz though the Occusense team has updated the sampling rate to 12Hz.

4.2 Person Detection

This algorithm is implemented in the file backgroundSubtraction.m. It takes as input a structure of parameters and a 3-D array of temperatures over the two spatial and one time dimensions. This implementation could be modified to run in real time and, depending on the size of the history for the computation of the background PDF, would not require too much memory.

4.3 Direction Discrimination

5 Experimental Results

5.1 Another section

Section with a figure (Fig. 1).

6 Conclusions

References

[1] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "A Novel Video Dataset for Change Detection Benchmarking," *IEEE Transactions on Image Processing*, vol. 23, pp. 4663-79, Nov. 2014.

[2] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis, "Background and Foreground Modeling Using Nonparametric Kernel Density Estimation for Visual Surveillance," *Proceeding of the IEEE*, vol. 90, pp. 390-393, July 2002.

[3] J. McHugh, J. Konrad, V. Saligrama, and P. Jodoin, "Foreground-Adaptive Background Subtraction," *IEEE Signal Processing Letters*, vol. 16, pp. 390-393, May 2009.

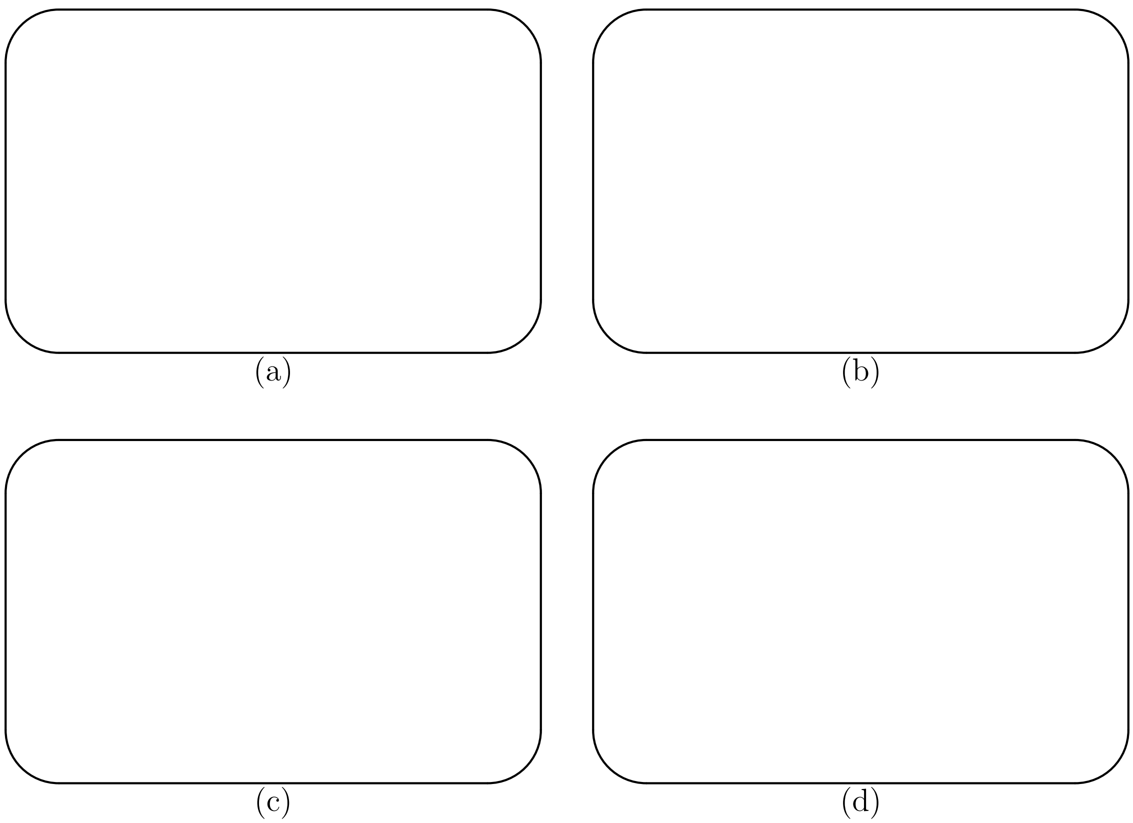

Figure 1: Block diagram: (a) one; (b) two; (c) three, and (d) four.