A Non-Intrusive Multi-Sensor RGB-D System for Preschool Classroom Behavior Analysis

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Nicholas Robert Walczak

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Nikolaos Papanikolopoulos

May, 2017

Acknowledgements

There are many people that have earned my gratitude for their contribution during my time in graduate school. First I'd like to thank my family for all of their support, especially my parents Ted and Roberta. My parents have always believed in me and supported my decisions and for that I am eternally grateful. Also my wife Amy who has stood by me throughout my time in graduate school, and my son Jacob and daughter Alexandra who joined us towards the end of my Ph.D. I would also like to thank my advisor, Nikos Papanikolopoulos, for seeing my potential and giving me the opportunity to complete this work. There are also many members of the lab who have made this possible. First is Joshua Fasching, who has been working with me since the beginning and contributed immensely to the code-base and also provided many good ideas. Also there is a host of other individuals who directly helped on the project, including Rachel Redmond, Walker Krepps, Danny Tormoen, Ben Bosch, Jack Spruth and others. Without these individuals, much of the foot work involved (setting up equipment, taking recordings, groundtruth labeling) would have been overwhelming.

Abstract

Mental health disorders are a leading cause of disability in North America and can represent a significant source of financial burden. Early intervention is a key aspect in treating mental disorders as it can dramatically increase the probability of a positive outcome. One key factor to early intervention is the knowledge of risk-markers – genetic, neural, behavioral and/or social deviations – that indicate the development of a particular mental disorder. Once these risk-markers are known, it is important to have tools for reliable identification of these risk-markers.

For visually observable risk-markers, discovery and screening ideally should occur in a natural environment. However, this often incurs a high cost. Current advances in technology allow for the development of assistive systems that could aid in the detection and screening of visually observable risk-markers in every-day environments, like a preschool classroom.

This dissertation covers the development of such a system. The system consists of a series of networked sensors that are able to collect data from a wide baseline. These sensors generate color images as well as depth maps that can be used to create a 3D point cloud reconstruction of the classroom. The wide baseline nature of the setup helps to minimize the effects of occlusion, since data is captured from multiple distinct perspectives. These point clouds, collected at almost 30 frames-per-second, are used to detect occupants in the room and track them throughout their activities. This tracking information is then used to analyze classroom and individual behaviors, enabling the screening for specific risk-markers and also the ability to create a corpus of data that could be used to discover new risk-markers.

This system has been installed at the the Shirley G. Moore Lab School, a research preschool classroom in the Institute of Child Development at the University of Minnesota. Recordings have been taken and analyzed from actual classes. No instruction or pre-conditioning was given to the instructors or the children in these classes. Portions of this data have also been manually annotated to create groundtruth data that was used in experiments to validate the efficacy of the proposed system.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Mental health disorders are a leading cause of disability: in 2004 the World Health Organization (WHO) estimated that in United States and Canada 25 percent of all years of life lost to disability and premature (Disability Adjusted Life Years or DALYs[1]) were caused by mental health disorders [1]. In a more recent report, [2] the WHO estimated that mental illnesses account for 37 percent of DALYs, which puts them as the leading cause from non-communicable diseases. This report estimates the monetary cost of mental illness to be nearly $2.5T with an increase to $6T by 2030.

Early intervention is a critical aspect in curbing the detrimental effect of mental illness. The best time to address mental illness is before the appearance of symptoms that disrupt daily life. Medical research has shown that symptoms of mental illness which emerge in childhood and early adolescence are actually the later stages of a process which began years earlier. In recognition of this, the United States National Institute of Mental Health has, as part of its strategic plan [3], the goal of charting mental illness trajectories to determine when, where, and how to intervene.

By the time symptoms of a mental illness begin to manifest, the optimal time to intervene may have already passed. Instead, it is important to look for risk-markers that indicate an elevated risk for development of a disorder. These risk-markers may include genetic, neural, behavioral, and/or social deviations from the norm. While many

[1] The WHO define DALYs as a measure of overall disease burden, expressed as the cumulative number of years lost due to ill-health, disability or early death. It is a combination of the years lived with disability (YLD) and the years of life lost (YLL). YLL uses expected life years to compute the number of years lost due to premature death.

risk-markers cannot be visually observed, there is an important class that can be. For example, the work of Esposito *et al.* [4, 5] uses retrospective home videos to show that gait asymmetry can be used as a risk-marker for the development of Autism. This work and others such as [6], [7], and [8] have performed their studies through painstaking manual coding of video data and observational data. These kinds of studies, while very important, require a large investment in both time and labor.

Modern advances in sensors, computational power, storage, and computer vision techniques can provide the possibility of automating a significant portion of this time consuming work. In addition, an automated system could be installed in a natural environment to provide unbiased data on a large scale.

(a) RGB Example (b) Depth Example

Figure 1.1: One Kinect RGB-D frame taken at the Shirley G. Moore Lab School, yielding both (a) an RGB room image and (b) depth data. Depth is contrast normalized for visualization. (Faces blurred to preserve anonymity. Taken with IRB approval.)

1.1 Motivation

Detecting and tracking people in a natural environment is a challenging problem that has been of interest to the computer vision community for a long time[9]. Any approach to this problem has many difficult problems to overcome: occlusion by other occupants or objects, self occlusion, changing environmental conditions such as lighting or moving objects. Many RGB-based solutions exist, however limiting solutions to RGB cameras imposes unnecessary limitations. The low cost and ubiquity of 3D devices such as the Microsoft Kinect RGB-D device, allows this problem to be approached in new ways. By creating a networked multi-sensor system, some of the classic challenges can be

mitigated. Such a system can be used to monitor child classroom activity over a long period of time. This long-term activity can then be used in various beneficial ways, such as an aid in screening for the development of mental illness, or gathering data to help understand early development and mental illness.

1.2 Thesis Statement

This work describes the development of a multi-sensor multi-modal system for the analysis of behavior in a preschool classroom. The current system consists of 5 RGB-D sensors (initially Microsoft Kinects, which were eventually replaced with ASUS Xtions) and is able to create a fused 3D point cloud representation of an actual preschool classroom. This system is able to identify the occupants of the classroom and track them as they go about their normal activities within the classroom.

This enables the ability to gather behavioral information about the classroom occupants which can be used to detection known risk-markers or learn about new ones. Using the tracking data of each subject, a social graph can be created, which shows the relationship of each room occupant based on their proximity over time. This social graph, along with other behavior metrics, can be explored using a developed interactive visualization tool.

The system is validated on real-world data taken from the the Shirley G. Moore Lab Schoolon an average classroom day. The data was manually annotated to create groundtruth which is compared to the output of the system. This validation shows the efficacy of the methods applied.

1.3 Contributions

The purpose of this thesis is to describe the development and operation of a multi-sensor RGB-D system for behavioral analysis in a preschool classroom. The major contributions of this work include:

- Describing the development of a framework to fuse RGB-D data coming from multiple sensors into a global 3D point cloud using off-the-shelf consumer hardware.

- Developing a background subtraction approach that utilizes 3D information, which outperforms traditional RGB-based methods in experiments.
- Developing a method to utilize the global 3D point cloud for detecting occupants of the room, which involved a novel hierarchical approach.
- Developing a system for longterm tracking of detections (room occupants).
- Developing methods for using tracking data to analyze classroom behavior, such as the development of a proximity-based measure that is used to construction a social relationship graph.
- Developing an interactive visualization tool to allow experts to explore the resulting behavior data, enabling fundamental contributions to behavioral science.
- Developing methods to test and validate system performance, with and without hand-labeled groundtruth.

The rest of this thesis is laid out in the following manner. The next chapter (Chapter 2) describes related work and background. This is followed by Chapter 3 which describes the architecture and details of the proposed system. Following that is Chapter 4 which describes the methodology for validating the system and experiments with groundtruth that show the efficacy of the system. The behavioral analysis portion of the work is then presented in Chapter 5. Finally, the thesis is concluded in Chapter 6, which summarizes the work and also proposes future directions. Note that throughout this dissertation images from a classroom at the Shirley G. Moore Laboratory school are presented. IRB restrictions require that no identifying information is released so images are carefully chosen as to not depict faces of individuals; if faces are presented they are blurred so as not to be identifiable.

Chapter 2

Background

2.1 Risk-Markers and Early Childhood Observations

Prevention is important as intervening earlier, before the onset of a disorder, can allow for better outcomes, both in terms of quality of life for the patient as well the cost of care. The idea of prevention as a tool for public health with regards to mental health is something that has only begun to gain traction in the past several decades [10]. Autism, a disorder that has seen much attention recently, provides a strong example. The first Autism Research Matrix [11] lists the identification of behavioral and biological markers of risk for autism as a top priority. Over the past few years, a number of studies have provided support for early intervention in autism: [12], [13], [14]. The latter study covers more comprehensive and intensive intervention, and shows that children who received the Early State Denver Model (ESDM) intervention showed significantly greater gains in IQ, language and adaptive and social behavior compared to children who received standard treatment in community. Schizophrenia is another example where early intervention can prevent irreversible damage [15].

In many disorders, the key to early intervention is to act in the premorbid phase, prior to when the disorder starts eliciting symptoms (the prodromal phase). This is accomplished by looking for risk-markers: genetic, neural, behavioral, and/or social deviations from the norm that indicate an elevated risk for the development of a disorder. Numerous examples of these can be found in Autism and Schizophrenia. The work in [16] provides a summary of a number of behavioral risk-markers of autism, including

decreased social-communicative behaviors (such as failures to orient to social stimuli), reduced eye contact, and fewer communicative gestures. The work of [4, 5] shows examples of where video analysis of the unsupported gait in toddlers can be used as risk-marker for the later development of Autism.

Many of these studies are conducted by manually coding video data and analyzing the results. Alternatively, activities can be coded by real-time observations. The coding procedures generally consist of looking for the occurrence of particular activities, such as involuntary motions (*i.e.*, various stereotypies in Autism), or responses (or lack of) to social stimuli (*i.e.*, not responding to a name). The coding form in [17] lists activities such as number of smiles, laughs, vocalizations to other children (both initiated by the subject and by other children), and other abnormal movements. This work suggests that the brief videotaped footage of children eating lunch was able to discriminate between the individuals who later developed schizophrenia and those who did not.

Social interactions are a common area to look for risk-markers, especially in Autism. The interaction between young children and caregivers is important yet complex, and in fact is central to the care and education of children under 3 years [18]. Works such as [19] indicate that the presence of a caregiver yields a positive impact, providing children with a "secure base" for interacting with other children. In addition, the acquisition of complex communication skills in children can be aided by the presence of adult caregivers. Other work however, such as [20], indicates that adult caregivers have an inhibiting effect on peer interactions. The work in [8] claims that these theories are not antagonistic, rather they are contextually dependent. This was shown through a painstaking study which required large amounts of operator controlled video monitoring along with manual coding schemes. Another example can be found in [21], which studies the role geography (location) plays in micro-social interactions of young children. The study's data is collected by manual annotation from observers watching the children.

All of these observational studies have provided valuable data to the field of psychiatry, however each one involves an immense amount of work in planning, preparation, setup, observation, and analysis and these studies are limited by the resources available. There are certain aspects that are infeasible to study because of the amount of effort required. Ultimately, there may be many risk-markers that remain undiscovered due to the amount of effort required to perform the requisite studies, and also the feasibility

in detecting the risk-markers once discovered. By using state-of-the-art equipment and algorithms, many of the important aspects of these observational studies could be automated, allowing for a broader range of potential studies. A system, such as the one presented here, that can automate parts of the coding process and create a large corpus of data, could have a large impact on the discovery and screening of visually perceptible risk-markers.

2.2 Behavior Imaging and Image-based Person Detection

The application of computer vision to the detection of risk-markers for neurodevelopmental disorders is a young and growing field. Several groups have begun specifically focusing on recognizing symptoms related to Autism using purely image-based approaches. In [22], Rehg *et al.* monitor a professional and child subject undergoing the Rapid ABC exam for ASD. The work examines methods for predicting the subject's engagement in an activity using audio and visual information. Hashemi *et al.* [23] focus on engagement of the subject being examined in a protocol similar to the Rapid ABC. Their method focuses on modeling the appearance of body part landmarks on the subject's face to measure the exact orientation of the face relative to an object. Their work also examines a separate risk-marker of gait asymmetry in a subject using body pose tracking.

Using the spatio-temporal interest point (STIP) detector of [24], three motor stereotypic behaviors associated with autism were examined in [25] for classification. Their dataset is comprised of YouTube videos of children at various ages performing motor stereotypic behaviors. While the importance of this work is not debated, there are numerous other applications of computer vision to risk-marker detection that are still unexplored.

Outside of applications to child psychiatry, image-based person detection has been a well researched area for a long time. This problem has thousands of papers published, and, while results have vastly improved over time, the problem still remains unsolved. Several recent surveys on single-sensor person detection [26, 27] cover the breadth of approaches to this problem. Perhaps one of the most famous recent approaches can be found in [28], where people detection is performed by using a histogram of oriented

gradient (HOG) feature which is classified using a linear support vector machine (SVM). HOG features still remain popular features to use for person detection, being used as the features in the parts-based deformable models used by [29, 30]. This deformable parts-based model approach achieved the highest average precision on the PASCAL Visual Object Classes challenge [31] 2010 [1] . More recently, the performance of these parts-based deformable models has been eclipsed by convolutional neural networks [32, 33]. When moving from detection on still images to sequences of images (*i.e.*, videos), some form of tracking scheme is required to maintain the identity of detections between frames.

2.3 Detection and Tracking

While track-before detection (TBD) approaches exist (*i.e.*, see TBD chapter in [34]), the standard approach to the tracking problem is to take the input sensor data and create measurements or detections, then associate the observations and measurements, perform track maintenance, and then perform filtering/prediction. This approach is frequently referred to as track-by-detection. Each step in this process is critical but approaches vary immensely, as do the applications of tracking systems.

A well researched combination has been pedestrian detection using RGB images. A well-known example can be found in the W^4 system, [35] which uses monocular grayscale images and models of people based upon shape and appearance. Another seminal example can be found in Pfinder [36], which uses a a multi-class statistical model of color and shape. An example of more modern work can be found in [37].

In a track-by-detection framework, the ability to associate observations to tracks is crucial. Generally this is done by combining various pieces of information together to form an associate score, then this associate score is used to create matches. These matches can be done in a greedy fashion however this does not lead to an optimal solution. Alternatively, the Munkres [38] or Hungarian algorithm can be used to find an optimal matching. This association score commonly consists of features from the tracker (such as position and velocity), but also must contain further information to disambiguate tracks in close proximity. Previously mentioned works have incorporated

[1] http://host.robots.ox.ac.uk/pascal/VOC/voc2010/results/index.html

2D appearance features such as color histograms.

2.3.1 Track/Observation Association Problem

Another attractive feature is region covariance descriptors. Region covariances were first introduced by Tuzel *et al.* in [39] as an appearance descriptor for textures, and they were originally applied to tracking in [40]. More recently, [41, 42, 43] also use probabilistic approaches to track region covariances over the Riemannian manifold to which these descriptors belong. In [41], Khan and Gu use two particle filters - one for tracking an object's bounding box and another for the object's appearance by tracking the region covariances over the Riemannian manifold. Wu *et al.* [43] track covariance descriptors on the manifold using a novel probabilistic approach called Incremental Covariance Tensor Learning (ICTL). Chen *et al.* [42] also provide a probabilistic update method on the Riemannian manifold with a random walk approach. In [44], the authors use region covariances as appearance descriptors in a Multiple-Instance-Learning (MIL) framework for data association and tracking across multiple cameras in a network. Zhang and Li [45] use a combination of Gabor and LBP (local binary pattern) features in their covariance descriptor for re-identifying people across different cameras. Tracking using covariance descriptors is a well researched area and only a small sampling of the literature is presented, for an in-depth survey of tracking techniques in general, see the survey in [46].

Single-sensor tracking systems are abundant, however ultimately a single sensor is insufficient for covering a large area. The field-of-view of a single sensor may not cover the whole area of interest, and then objects in the scene can cause occlusions from a single viewpoint. In smaller settings, like a classroom, with a large number of occupants interacting in complex ways, leveraging the power of multi-sensor systems is imperative.

2.3.2 Multi-sensor Tracking Systems

Multi-sensor systems for object detection and tracking are a widely researched topic. Initially starting with radar systems, there are several books that cover the field in great detail, including [34] and [47]. An early example of a multi-sensor system for use

in a "smart room" is presented in [48]. This "smart room" system uses computer vision-based monitoring to control certain aspects of a simulated living room, such as playback control of a movie when a person gets up or sits down from a couch. The system operates by using two calibrated stereo cameras, which yield depth and color images that can be used to create foreground blobs. These blobs are created by computing a mean and standard deviation for depth, and each individual color channel in the RGB images across 30 frames and comparing pixels to these models. Blob merging techniques are used to merge foreground blobs and the assumption is made that only people will appear in the foreground. Another example of a multi-camera tracking system for "smart room" applications can be found in [49]. While interesting, this approach would not fare well in a cluttered real-world environment like a preschool classroom.

Another multi-camera system is presented in [50], which utilizes an array of 16 wide-baseline stereo cameras. In that system, explicit models of people are built up based on color and "presence" probabilities. Intersections of epipolar lines across multiple cameras are used to determine a 3D location of tracked people on the ground-plane. A Kalman filter is then used to track these locations over time. While the experimental results they present are impressive, their system relies upon particular assumptions about people in the scene (such as the people are standing, not jumping, etc) which are unrealistic for the applications presented here.

A more recent work, [51], presents another multi-camera system for tracking people that incorporates information from a large number of RGB cameras. This work, like many other systems, uses a ground-plane homography to match image-plane locations to a 2d ground plane location. Proprietary background subtraction is used to compute foreground blobs in the image. These foreground blobs are then combined with a generative model to compute occupancy in a quantized occupancy grid on the ground plane. Grid-locations with a high enough degree of occupancy are considered positive person detections, and these are tracked across time.

In [52], Munaro *et al.* describe a method for detecting and tracking people using RGB-D that shares many similarities with the methods presented here. Their detection pipeline works by downsampling the point cloud created from the RGB-D input with a voxel grid filter, removing a ground plane detected with RANSAC, then clustering the remaining points based on Euclidean distance, and then finally running a histogram

of oriented gradient (HOG) based person detection algorithm on the RGB image projections of the clustered point clouds. This approach faces a similar problem to the approach presented here in that multiple people standing close to each other will get merged into one detection. Munaro *et al.* solve this with their sub-clustering step which relies upon detecting heads. This, along with some of their other decisions for removing detections such as detections too close to the ground or too high off the ground, make this approach unsuitable for the presented application. Frequently children will be sitting, laying, or rolling on the ground, and the vast array of different poses the children can assume invalidate the assumptions made in the approach described in [52].

2.4 Person Reidentification

In addition to the association problem, another common issue with tracking systems is reidentification. If a track is lost and reappears, it would be ideal if the new track has its identity matched with the previous track. This can happen if an individual leaves the tracked area, or becomes occluded for a long period of time, or if the system operates over the course of several days. The reidentification process can be cast a clustering problem, where the input is a set of tracks which need to be clustered based on the identify of the occupant being tracked. The overall framework consists of determining what features to extract from the track, how to describe those features, and finally how to match features.

In image-based tracking systems, re-identification typically exploits low-level image features such as color [53], texture, spatial structure [54] or combinations of these [55, 56, 57]. These features are attractive for the same reasons that they see wide spread use in other computer vision areas: they are quick and easy to implement and compute and they provide some level of discriminative power. They can then be encoded with common techniques such as histograms, covariances or fisher vectors. However, these low-level image features do not incorporate other data that may be available with different sensing modalities (like infra-red) that may enable different techniques or depth-sensing devices that could enable 3D information to be used.

In addition to decisions about which features to use and how to encode them, a selection of technique to match the features must be made. This can be thought of

as determining what kind of clustering technique to use. Nearest-neighbor (k-NN) is a straight forward approach as it does not require learning any model *a priori*. Additionally, other model-based approaches can be employed such as K-means and other common clustering algorithms. An important aspect in the process is picking the distance metric to use. Euclidean distance is a common choice, however this is highly dependent upon the encoding scheme used. For instance, with histograms the chi-squared distance metric or histogram intersection are generally more amenable. Additionally, using Euclidean distance for covariance matrices throws away structure since covariance matrices lie on a Riemannian manifold [58]. Instead, the geodesic distance is most appropriate but is expensive to compute, however faster-to-compute approximations exist such as the log-euclidean metric [59] or Jensen Bregmen log-det divergence (JBLD) [60].

Within the reidentification literature, several distinctions between approaches are frequently made. If only image pairs are being matched, this is generally referred to as a *single-shot recognition* approach. If, instead, sets of images are considered, then the methods are referred to as a *multi-shot recognition* approach. Additionally, approaches that make use of *a priori* labeled data are referred to as *supervised* while those that do not are *unsupervised*, similar to the distinction made with clustering algorithms.

Despite the wealth of current research on this topic, many challenges still remain to this difficult problem. A recent survey that focuses on the surveillance and forensic scenarios can be found in [61]. A broader survey on approaches and trends in person re-identification can be found in [62]. Additionally, a whole book has recently been devoted to the problem [63].

Chapter 3

System Setup

This chapter details the setup of the system, and goes through each processing step. Many of the design decisions for this system have been guided by the final goal. From the beginning, the system was designed to be non-intrusive, so the natural environment of the classroom can not be disrupted. This puts a severe limit upon the technologies that can be considered. In particular, no markers should be placed on the subjects in the classroom, meaning that common approaches like fiducial markers could not be used. Additionally, the sensors themselves should only cause minimal disruptions, so anything that is abnormally large and/or bulky, or emits an audible noise should be disregarded. This eliminated all of the current LIDAR-based sensors at the time, although some of the newest LIDAR offerings might be applicable.

The environment of the classroom is also particularly tricky. The first consideration is children versus adults: children are considerably less predictable and the range of acceptable behavior is much wider. In the traditional task of pedestrian detection, subjects follow a fairly narrow behavior pattern: walk down the side walk while avoiding

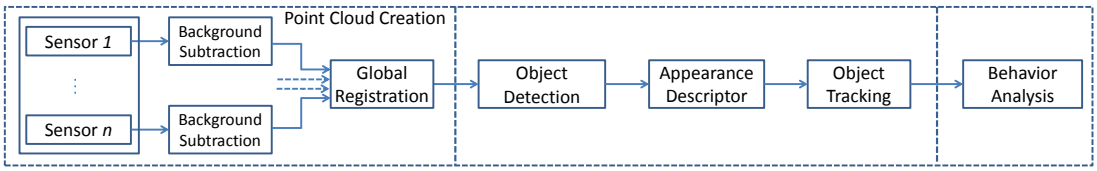


Figure 3.1: A flowchart of the processing pipeline.

other people. Children during free play have a vastly larger assortment of activities, including running, rolling on the ground, wrestling, dressing up, and vigorously interacting with one another. The classroom also contains an assortment of toys, chairs, apparel, and other objects that can add clutter and occlusion to the scene.

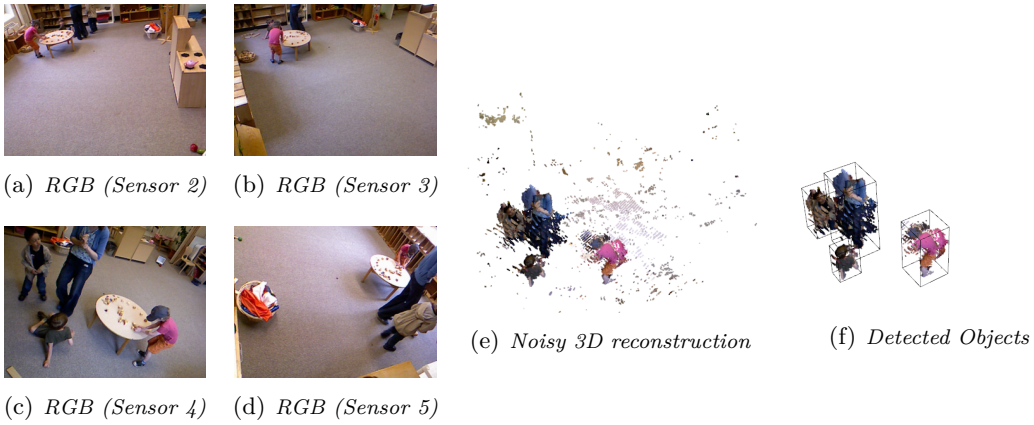
Figure 3.1 gives an overview of the system and defines each conceptual block. The first several sections will describe the sensors, their setup, and how the data is acquired, synchronized and projected into point clouds. This is followed by Section 3.4 which describes how the system is calibrated so that the point clouds generated in a sensors frame of reference can be transformed into the global frame of reference. The final piece of the first block is described in Section 3.5, which describes how static background points are removed. These sections combined encompass point cloud creation, which is the first conceptual block of the processing pipeline. Once the global point cloud has been created, it is processed for object detection (Section 3.6) and tracking (Section 3.8), with appearance descriptors (Section 3.7) computed to aid in tracking and reidentification (Section 3.9).

The first two conceptual blocks in the pipeline are exemplified in Figure 3, which depicts the view from 4 of the sensors, a noisy background-subtracted globally registered point cloud, and then the occupants detected from that point cloud. The final conceptual block, the analysis of the output of detection and tracking, is covered in Chapter 5. Prior to covering analysis, the system is validated in Chapter 4, so that some level of confidence can be gained in the results of the behavior analysis.

3.1 Sensor Selection

The first major design decision of this system was to determine what type of sensor to use. Traditionally, a behavior monitoring system such as this would use a network of calibrated RGB cameras. However, by incorporating depth sensing devices, it is much easier to incorporate 3D structural information into the processing.

Given the the challenging nature of the problem, and the recent development of inexpensive and readily available depth sensors that also provided color (RGB-D), the decision was made to design the system using 3D information. At the time of sensor selection, the newly released Microsoft Kinect (Figure 3.3(a)) was able to generate



(a) *RGB (Sensor 2)* (b) *RGB (Sensor 3)* (c) *RGB (Sensor 4)* (d) *RGB (Sensor 5)* (e) *Noisy 3D reconstruction* (f) *Detected Objects*

Figure 3.2: (a)-(d) Sample simultaneous RGB frames from four of five sensors. (e) The fused RGB and range data forming the (noisy) 3D point cloud rendering the scene. (f) The four persons detected and enclosed in four bounding boxes. [Compare (c), (e) and (f).]

640 × 480 depth and RGB images at approximately 30 frames per second (FPS) for a price was was considerably lower than competitors. The Kinect was based on Prime Sense's Carmine reference device, and since then several different iterations of the same reference device have been released, such as the ASUS Xtion (Figure 3.3(b)).

Compared to LIDAR-based devices, the resolution, frame-rate, and price are very competitive. A Velodyne HDL-64 Lidar[1] (see Figure 3.4) costs tens of thousands of dollars, and uses a spinning array of 64 lasers to generate depth information. It can complete a full 360 degree sweep at 5-20Hz, and while its maximum distance of 120m range compared to the Carmine's 4m is an advantage, the HDL-64 uses a class-1 laser and creates a very audible hum with its visibly spinning head. Given the several major disadvantages (cost, noise, *etc.*), the use of such a device was deemed inappropriate. While the HDL-64 has fallen out of favor compared to its smaller brother, the HDL-32, the same disadvantages still exist.

Another solid competitor to the Carmine RGB+D sensor was stereo camera setups. There are systems such as the Point Grey Bumblebee[2], which use the the geometric relation between the same point imaged in multiple lenses to compute a disparity map.

[1] See: `http://velodynelidar.com/hdl-64e.html`

(a) Microsoft Kinect (b) ASUS Xtion

Figure 3.3: Example RGB-Dsensor, based on the PrimeSense Carmine reference device. ASUS Xtion image taken from ASUS webpage: `https://www.asus.com/3D-Sensor/Xtion_PRO/`

At the time of sensor selection, the cost of stereo cameras was still higher compared to Kinects. Additionally, since stereo vision relies upon matching points in two images, there can be issues in the disparity maps under certain conditions, particularly with lighting.

While the system currently uses Carmine-based RGB-D devices, the processing pipeline is agnostic to the sensor type employed; any device that can generate a colored point cloud could be used. After sensor selection was done, various other suitable sensors have been released, like the Intel R200[3] or the various Orbbec Astra devices[4] . Since this system was designed at a time when consumer depth sensors were just hitting the market, it is inevitable that newer more-competitive sensors will become available. More experiments could be performed with other sensors, however the cost in time and money for such experiments is prohibitive and outside the scope of this work. The system was designed to be modular and extensible and the methods described below are still applicable for any network of devices that can generate a single global point cloud.

3.2 Sensor Setup

The current system consists of five ASUS Xtion RGB-D sensors which are based upon the Prime Sense Carmine reference device and functionally equivalent to the Microsoft Kinect. These devices generate an RGB image and a depth map, an example of which can be seen in Figure 1.1. These sensors are positioned roughly in the corners of a portion

[2] See: `https://www.ptgrey.com/stereo-vision-cameras-systems`
[4] See: `https://software.intel.com/en-us/RealSense/R200Camera`
[4] See: `https://orbbec3d.com/`

of the classroom such that each sensor has a unique wide baseline viewpoint (Figure 4.5 in Chapter 4 gives an indication of camera placement). Sensors were positioned to minimize the effects of occlusion.

Previous iterations of this system [64], had multiple host systems which controlled up to 2 sensors. The current version of this system uses USB to ethernet converters to connect all 5 sensors to a single host controller in a separate room. Each sensor is connected to a single control host running GNU/Linux. The bandwidth requirements of each sensor is such that each USB controller can only support two devices simultaneously. Because of this, the control host has several USB expansion cards installed in order to support all five sensors.

There are also several issues that must be overcome because of the consumer-grade nature of the sensors. The first issue is a lack of `genlock` capability, which means that data feeds from each sensor must be synchronized manually. Recordings for each sensor are initialized simultaneously but since the host operating system is not real-time and the sensor feeds are not synchronized via hardware, there can be small differences in when the frames arrive which requires some form of temporal synchronization. For now, these small temporal inconsistencies have been rectified by manual alignment, but in the future a method for automatic temporal synchronization could be developed. The topic of video temporal alignment is active, and various approaches exist, however for the purposes of creating and testing the initial system, manual alignment has been sufficient.

Figure 3.4: Example of a spinning LIDAR, the Velodyne HDL-64. Image taken from Velodyne's website: `http://velodynelidar.com/hdl-64e.html`

Another issue is that the frame-rate of each device is not guaranteed and can fluctuate, which is another source of temporal misalignment. If frame generation does not meet real-time deadlines, than a particular frame can be dropped, which leads to a variable frame-rate. This issue is dealt with through a method of binning the frames received from each sensor. The recording time is discretized into a series of bins, with each bin being $\frac{1}{30}s$ wide, corresponding to the desired frame-rate. Then, each bin is filled with the closest corresponding frame, or a previous bin's frame if no corresponding frame is found.

More formally, time intervals t_i of one second each are discretized into a sequence of thirty bins b, with each bin representing $1/30^{\text{th}}$ of a second for that sensor. Each frame F from a given sensor recording is then added to a bin b_i if it falls into that bin's time frame as described by the following equation:

$$b_i = \begin{cases} F, & t_i \leq F < t_{i+1} \\ b_{i-1}, & \text{otherwise.} \end{cases}, \tag{3.1}$$

otherwise the previous frame is repeated.

(a) Single Sensor (b) All Sensors (c) Present Person

Figure 3.5: Examples of the effects of sensor overlap on generated depth. In (a), only a single sensor is on and very little depth is lost. When all of the sensors are turned on in (b), areas on the floor are lost, due to the overlap in the structured light pattern. However, when a person walks through the room, as in (c), good depth data is still received for that person.

One further issue relates to the structured light nature of the RGB-D sensors employed. The sensors work by projecting an IR speckled dot pattern, and then capturing that pattern with an IR-filtered CCD camera. The deformation of this pattern can then

be used to compute depth information. However, when multiple instances of these sensors are used, the IR speckled dot pattern from multiple sensors can cause interference. Others, such as [65], have explored ways to reduce this interference.

For the purposes of this system though, given the wide baseline and varying viewpoints of the sensors, objects in the scene largely occlude the the IR pattern from different sensors. There are areas in the room where the patterns do overlap significantly, but in general these areas correspond to regions of little interest, such as the floor. Once a person enters the scene, the amount of pattern overlap is minimized and good depth data is returned. Figure 3.5 illustrates this, by showing a frame where other sensors are turned off, then with other sensors turned on, and finally with people present while other sensors are on.

3.3 Sensor Characterization

All sensors contain some amount of noise in their measurements, and understanding the noise present in the sensor output is an important aspect for achieving quality results. Once the decision was made to use Prime Sense Carmine based devices, effort was spent characterizing that sensor. In order to measure the quality of depth data generated by a Kinect, a sensor was attached to a tripod with an attached plumb-bob used to measure the position of the sensor. The sensor was then used to image a flat surface (a wall) consuming the whole field of view of the sensor at a prescribed distance. At a given distance, the wall is imaged and the depth map is saved, then the distance was varied from 1m to 7m.

A window of image locations in each depth image were chosen that correspond to the wall. The world location of each of these points with respect to the sensor were calculated as:

$$\mathbf{x}_w = d_{i,j}\mathbf{K}^{-1}\mathbf{x}_{img}, \tag{3.2}$$

where, $\mathbf{K}$ is the intrinsic camera matrix for the Kinect's infrared camera, $d_{i,j}$ is the depth measurement at point (i, j), $\mathbf{x}_{img} = (i, j, 1)^T$ is the homogeneous image coordinate of the point, and $\mathbf{x}_w$ is the inhomogeneous world point.

A plane π was then optimally fit to these world points by minimizing the $L2$ norm.

This was done by finding the 3×3 covariance matrix $\mathbf{A}$ for the points, and then finding $\mathbf{v}$, the right null-vector of $\mathbf{A}$. The plane π was then defined as

$$\pi = (v_1, v_2, v_3, -\mathbf{v} \cdot \mu)^T, \tag{3.3}$$

where μ is the mean point location With the best-fit plane π found, the distance of the Kinect from the best-fit plane is simply π_4, since the Kinect is located at the origin of the reference frame.

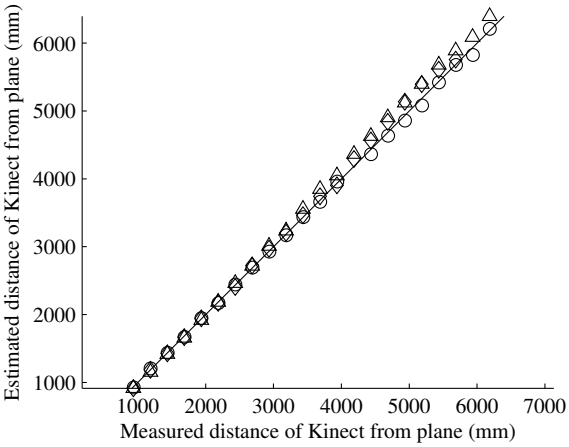


Figure 3.6: Actual distance of the Kinect sensor from a wall versus the estimated distance of the Kinect sensor from the best-fit plane of detected points. Markers of different shape indicate results from different Kinect sensors. The black line shows the expected relationship, where the estimated and actual distance are equal.

The results of this process using three different Kinect sensors are shown in Figures 3.6, 3.7, and 3.8. In each of these plots, a single shape marker was used for each of the Kinect sensors. Figure 3.6 shows the manually-measured actual distance versus the estimated distance to the best-fit plane. The straight line shows the expected equivalent relationship between the points, where measured and estimated distances are equivalent. This plot gives an indication of how accurate the measured depth is as a function of distance. From the plot, it can be seen that from distances less than 3m, all three of the sensors are very accurate, but starting around 3.5m the measured distance starts to vary.

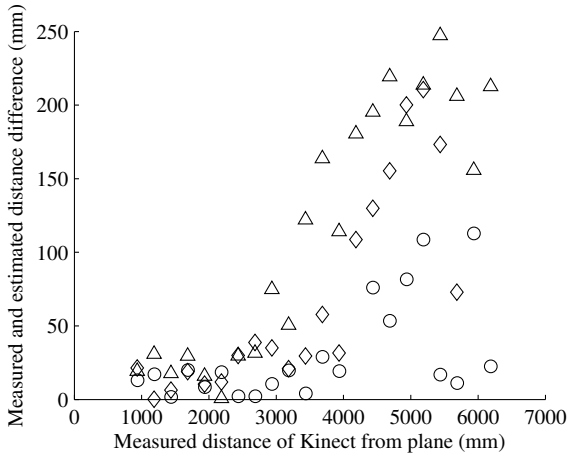


Figure 3.7: Difference between actual physical distance and estimated distance versus actual distance. This plot indicates how the error between actual and measured distance increases greatly as the measured object gets farther away from the sensor. Markers of different shape indicate results from different Kinect sensors.

Figure 3.7 shows the difference between the manually-measured actual distance and the estimated distance (error) as a function of the manually-measured actual distance. Visualizing the data in this way does a better job of showing how the error between the actual distance and the estimated distance grows as distance from the object increases. This plot shows that between 1m and 3m, the noise in the depth measurements tends to be fairly controlled. Unfortunately, beyond 3m, the observed noise amplitude grows swiftly and chaotically. This indicates that reconstruction within the optimal range of 1–3m results in decent reconstruction, while noise accumulates significantly beyond this. It is also worth noting that, while noise in the optimal range range is somewhat limited, measurements are still often off by several centimeters.

The plot in Figure 3.8 demonstrates how self-consistent the distance readings are based on distance. Each point represents the mean distance of the points detected on the wall from the best fit plane for each actual distance. Since the sensor is imaging a flat surface, all points should be measured at the same distance. Computing the mean distance from the best-fit plane gives an indication of how much variance there is in the measured distance. As can be seen, this mean distance is very small for all cases, even though it increases slightly with distance. This indicates that even though the

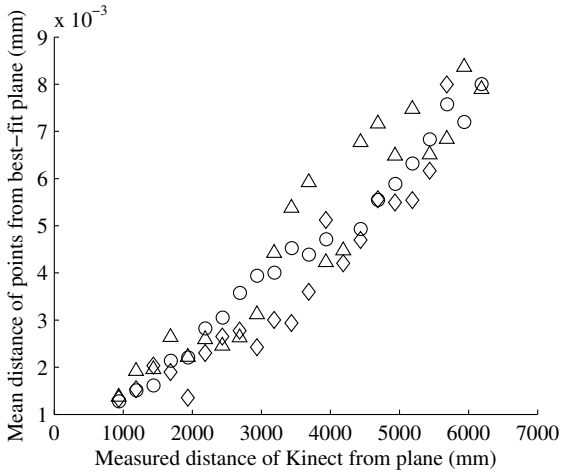


Figure 3.8: Mean distance of the points from their best-fit plane versus the actual distance of the Kinect sensor form the real-world plane. Markers of different shape indicate results from different Kinect sensors.

estimated distance between the sensor and the wall diverges from the actual distance, the actual distances are remarkably consistent.

Figure 3.9: Raw data from a PrimeSense Carmine device of a flat wall at varying distances. Reproduced from [66].

These experiments are consistent with other work that has attempted to characterize the Prime Sense Carmine sensors, such as [66] and [67]. In addition to the depth errors that increase with distance, it also appears that the errors grow further from the camera center. Figure 3.9, reproduced from [66], illustrates this error. Because of these errors,

point clouds generated from multiple sensors may not align properly, even if sensor poses are accurately computed. It is very desirable to correct these distortions, however much of the data was recorded after these methods were widely known about, and calibrating after-the-fact has remained difficult. Instead, the methods employed have been designed to be robust to these errors, and operate despite them. Incorporating these distortion models has been left as future work, that has the potential to increase accuracy beyond what is reported here.

3.4 Point Cloud Creation

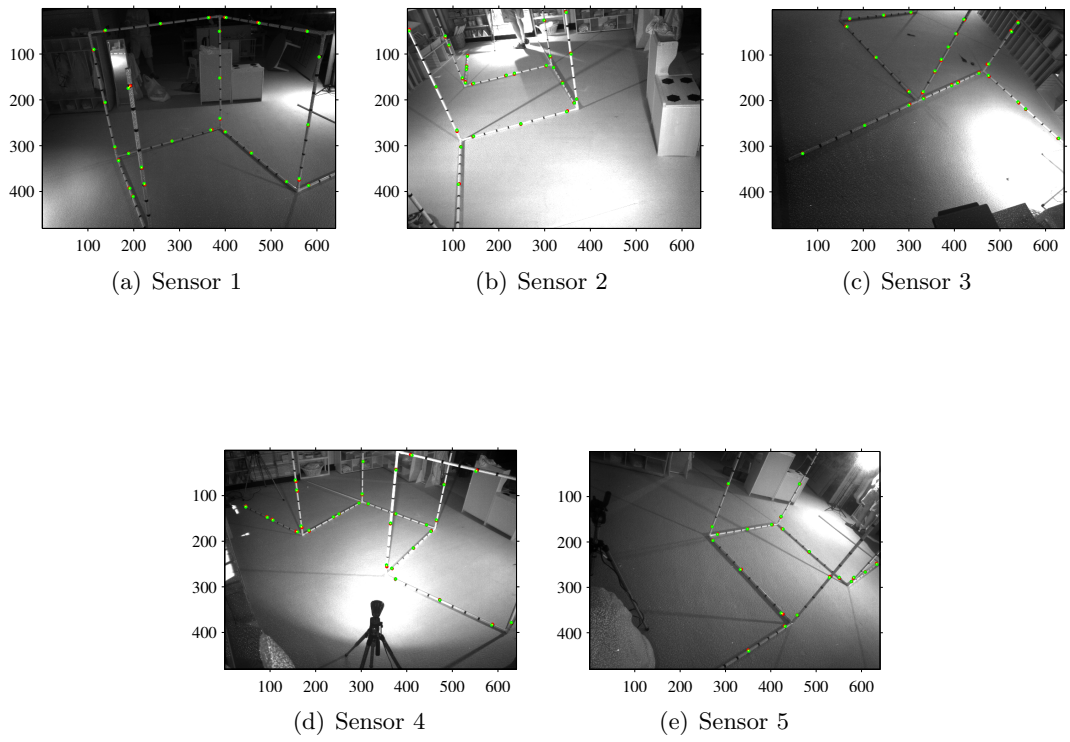
The depth data and RGB data generated by the sensors can be used to create a 3D color point cloud using the following equation

$$\mathbf{x}_c \;=\; \mathbf{K}^{-1}\boldsymbol{x}_c * d\,, \tag{3.4}$$

where $\mathbf{x}_c$ is a simple 3D point, $\mathbf{K}$ is the 3×3 camera matrix (intrinsic parameters), $\boldsymbol{x}_c$ is a homogeneous 2D camera point, and d is the scalar distance from the sensor.

Equation (3.4) can be used to generate a point cloud from a single frame of sensor data, however this point cloud will be in the frame of reference of the sensor. In order to combine the point clouds from each of the five sensors, the pose of each sensor, which consists of a rotation and translation must be known. This rotation and translation is commonly referred to as the extrinsic parameters and, along with the camera matrix (intrinsic parameters), can be found through the calibration procedure. The calibration procedure consists of finding a series of image to world point correspondences, creating a linear system and then solving using a combination of DLT and Levenberg-Marquardt. This procedure is referred to as the Gold Standard algorithm and is described in detail in [68].

A rigid 3D calibration rig was made of PVC piping as long as 1.5m, marked at regular intervals. The rig allowed for an easy mapping between image and world points. The image to world point correspondences were originally found by hand: the calibration rig was marked at regular intervals, so world points could be easily computable and these markings were distinctly indicated in captured images.



(a) Sensor 1 (b) Sensor 2 (c) Sensor 3

(d) Sensor 4 (e) Sensor 5

Figure 3.10: Images from the five RGB-Dinfrared sensors mounted in the lab school used to calibrate the cameras. The calibration rig is clearly shown in each, with the image points and associated reprojected world points used for calibration. Note the very close overlap between the image points and reprojected points.

Figure 3.10 depicts the calibration rig from the perspective of each of the 5 sensors and also indicates the points used for calibration and their reprojected values. This setup not only helps in spanning the field of view of multiple sensors, but also minimizes self occlusion, aiding the use of majority of the marked points on it. This was chosen over a method, such as [69], which requires synchronized cameras to track a calibration target over time. The results of the calibration procedure are summarized in Figure 3.11, where the position and orientation of each calibrated sensor is indicated by a colored pyramid and the points used for calibration are plotted in 3D.

Once the extrinsic parameters from each sensor have been computed, they can then

be used to transform each point cloud in the sensor's frame of reference to the global frame of reference. The globally registered point clouds from each sensor can then be aggregated into a single global point cloud, which can then be used to detect and track occupants of the classroom.

Ideally, the point clouds from each sensor would be registered seamlessly, with no discontinuities, in practice this is not the case. There is some error contributed from the calibration process, which in this case is relatively minor, but there is also distortion from depth image, which was eluded to in Section 3.3. Additionally, some error is added from inexact temporal alignment. The methodology used to detect and track the occupants was designed to be robust to these errors, and later sections will show that the system is able to achieve good performance despite these issues.

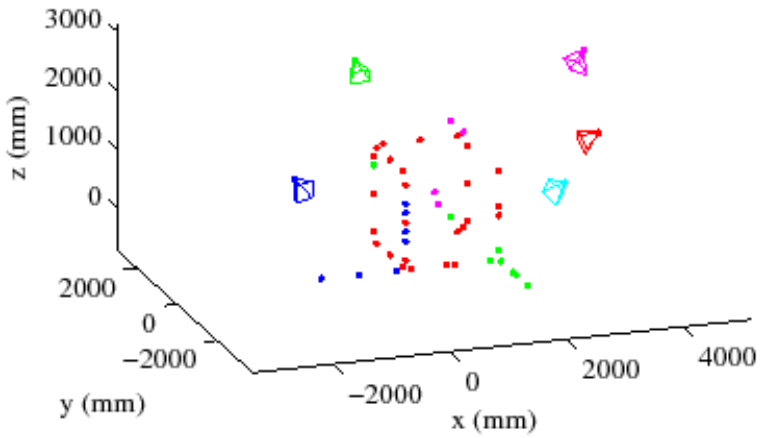


Figure 3.11: 3D reconstruction from calibration of the RGB-D infrared sensors, with sensor locations and poses indicated by pyramids. The world points used for calibration are also depicted, giving an outline of the calibration rig. Different colors separate the points and the sensors they were used to calibrate.

3.5 Background Subtraction

Each sensor generates a 640×480 depth map which can generate a point cloud with up to $307,200$ points, which when combined across five sensors can lead to a global point cloud with over 1.5 million points per frame. To keep computational costs down,

it is important to remove background points before performing expensive operations. Additionally, a large majority of these points belong to parts of the scene that are unimportant to tracking and analyzing occupant behaviors. Background subtraction is a common pre-processing step that is performed in many computer vision applications.

3.5.1 RGB-based Methods

In previous iterations of this system [64, 70], an image-based background subtraction method was employed. Two image-based methods were experimented with, the first [71] is a multi-layer statistical approach that uses a photometrically invariant color measure as well as texture. The second approach was based on robust principal component analysis (PCA) [72], which takes a different approach to background subtraction where the background of an image is considered to consist of a dense image that is either constant or gradually changing, and to consider the foreground as a sparse image that often consists of one or more moving objects. This conceptualization makes the problem amenable to a PCA-based solution, since the background can be considered to lie in a low-dimensional space and the foreground pixels are outliers to that space. Solving this problem can be represented mathematically as

$$\begin{aligned} \underset{A,E}{\text{minimize}} \quad & \|E\|_F \\ \text{subject to} \quad & \text{rank}(A) \leq r, \\ & I = A + E, \end{aligned} \tag{3.5}$$

where I is an input image, A is a low-rank approximation of I, E is the error in the low-rank approximation, $\|\cdot\|_F$ is the Frobenius norm, and r is an upper-bound on the rank of A. In this formulation, A represents the background (the static or gradually changing portion that lies in a low dimensional subspace) while E represents the foreground (the outliers to the low dimensional space). This problem could be solved with singular value decomposition (SVD), as with traditional PCA, however this generally yields poor results. Instead, a robust PCA solution needs to be considered.

The work in [73] presents a method of solving the robust PCA problem using convex optimization that minimizes a combination of the nuclear norm and the L1 norm, referred to as inexact ALM (Augmented Lagrangian Multiplier). K training frames are

sampled uniformly from the video sequence, and the frames are vectorized and concatenated into a matrix $A \in \mathbf{R}^{N \times K}$. N is the number of pixels in each color frame ($N = 230,400$), and K is usually of the order of $100 - 200$ frames.

For the inexact ALM approach to Robust PCA, A is decomposed as the sum of a low-rank matrix L, which represents the stable background, and a sparse error matrix S, which accounts for the foreground regions. The following objective was optimized:

$$\begin{aligned} \underset{L,S}{\text{minimize}} \quad & \|L\|_* + \lambda \|S\|_1 \\ \text{subject to} \quad & A = L + S. \end{aligned} \tag{3.6}$$

$\| \cdot \|_*$ represents the nuclear norm, a convex relaxation to the rank of L, and λ is a regularization parameter. From the low-rank matrix L, the "dictionary model" U of the background can be computed from the singular value decomposition of $L = U\Sigma V^T$.

A test frame $\mathbf{x}$ is decomposed as

$$\mathbf{x} = \underbrace{UU^T\mathbf{x}}_{\text{background}} + \underbrace{U_\perp U_\perp^T \mathbf{x}}_{\text{foreground}}. \tag{3.7}$$

Since this process was performed on color images, the foreground error image still consists of three channels. This is merged into a single-channel foreground error score at each pixel by taking the maximum across the three color channels. The foreground score image is then binarized using a threshold γ. The image intensity is normalized to be in $[0, 1]$, and $\gamma = 0.12$ was empirically selected. The binary foreground mask is then up-sampled through nearest-neighbor interpolation to the original image size, and used to filter out background points in the point clouds. Figure 3.12 shows a sample frame decomposed into the background and a thresholded foreground mask.

One issue with this approach is that the method in [73] presented a batch approach to the robust PCA problem, which does not work well on long sequences of video. This batch approach also presents a scaling issue, as longer frame sequences cannot be considered as they become too expensive in terms of computation and memory. An online approach, such as in [74], would be more suitable, however it was not tested as RGB-based approaches were found to be too error prone. Overall, RGB-based approaches did not compare favorably to approaches that incorporated depth information, as covered in Chapter 4 Section 4.5.

Figure 3.12: Sample frame demonstrating the robust PCA background subtraction step, splitting an input test frame (left) into the background model (middle) and foreground error. The foreground error is thresholded to yield a binary foreground mask (right).

A more traditional approach to background subtraction is to create a statistical model of the background for each pixel. A pixel in a query image is then compared to the model for that pixel and a probability of belonging to the background model is computed. These methods can have a static model, or can incorporate update steps to update the model online. One such approach, proposed in [71], was initially considered. This method uses local binary patterns and a photometrically invariant color measure to build statistical models for each pixel. Given an image sequence, $\{I^t\}_{t=1,...,N}$, the background model at timestep t is defined as $\mathcal{M}^t = \{M^t(x)\}_x$, where x is a pixel in the image. The per-pixel model is then defined as

$$M^t(x) \;=\; \{\, K^t(x)\,,\; \{m_k^t(x)\}_{k=1,...,K^t(x)}\,,\; B^t(x)\,\},$$

where $K^t(x)$ is a scalar that denotes the number of $m_k^t(x)$ modes, and the first $B^t(x)$ modes represent stable background observations. Each mode is defined as

$$m_k \;=\; \{\, I_k, \hat{I}_k, \check{I}_k, LBP_k, w_k, \hat{w_k}, L_k\},$$

where I_k is the average RGB vector, $\hat{I}_k$ and $\check{I}_k$ are the estimated maximal and minimal RGB vectors, LBP_k is the average local binary pattern, w_k denotes the weight factor, and $\hat{w_k}$ is the maximal value to which m_k belongs, with $k = 1, ..., K^t(x)$. $L_k = 0$ implies the mode does not belong to a stable background layer.

Once the model is updated, a background distance map is created, which is analogous to a foreground probability map. This distance map consists of the distance to the

closest mode for each pixel in the input image. If the matching mode does not belong to a known background layer ($L_k = 0$ and $k > B^t(X)$) then the distance is set above the foreground threshold. The distance equation between a pixel and the modes at that pixel location as well as the model update equation can be found in [71]. Figure 3.13 shows a sample frame using this multi-layer statistical approach to background subtraction.

Using either of these RGB-based background subtraction methods, a binary mask can be created on the depth map. This binary mask can be used to determine if a point should be projected into a point cloud. By applying the background subtraction mask in this way, the number of points that are projected is reduced, which saves on computational time spent creating point clouds, as well as time spent considering points in later steps.

3.5.2 3D Approach

There are, however, several drawbacks to purely image-based approaches. First, methods that are robust for complex scenes rely on deep statistical modeling and extensive image analysis, which in turn incurs considerably computational expense. Additionally, cameras relay no explicit data on *where* the background lies, merely images, and so are profoundly affected by occlusions, appearance or lighting changes, reflections, *etc.* In the end, the performance offered by purely image-based methods for this particular environment is only mediocre, which is detailed in Chapter 4 and in previous work [75]. Particularly, variations in illumination and background regions of high texture caused

(a) *RGB Image* (b) *Foreground scores* (c) *Thresholding applied*

Figure 3.13: Example frame of background subtraction using a multi-layer statistical approach.

numerous false positives in RGB-based background subtraction.

Instead of using solely RGB images, the 3D point clouds can be leveraged to create a background model in 3D. A simple *computationally economical* approach discretizes the 3D space into a regular grid of 3D volumes $V_s = \{v_1, v_2, ..., v_n\}$ (commonly referred to as voxels). To initialize the 3D model, 3D data points are first recorded while the room is vacant. Any voxels that contain 3D points in the empty room are then marked as occupied. Our classroom model then becomes this occupied subset of the voxel grid.

Essentially, the process will partition the observed scene V_s into two disjoint spaces, the background/obstacle space V_b and the foreground/free space V_f, where $V_s = V_b \cup V_f$ and $\varnothing = V_b \cap V_f$. Later, when creating foreground point clouds during the school day, any points that lie in V_b can be removed. This yields a robust method that is quick to apply, as it only requires a linear scan through all the points and a constant-time look-up for each point.

In addition to measuring voxel occupancy based solely on spatial information, color information can also be used. For instance, the mean and covariance of the RGB values of points within a voxel can be computed. To determine if an input point belongs to the background, the voxel it belongs to is first determined. If that voxel is occupied in the model, the color of the input point is then compared to the color distribution in the voxel using Mahalanobis distance,

$$\text{dist}(x_q, x_m) = \sqrt{(x_q - x_m)^T \Sigma^{-1} (x_q - x_m)}, \tag{3.8}$$

where x_q is the RGB vector of the query point, x_m is the mean RGB vector for the voxel, and Σ is the RGB covariance. If the distance exceeds a certain threshold, then the input point does not match the background model.

However, in practice applying color in this way was not very practical. By incorporating this information, the application of the background model was slowed considerably: using a simple occupancy model is quick because query is a simple constant-time index look-up and comparison. When incorporating color information, a more complicated computation must be made and when this gets applied across every point in the global point cloud, these costs can add up. Additionally, through experimentation, applying the color was not as beneficial as was considering a more robust method of

computing detection from the background-subtracted point cloud. Some experiments on this will be presented in Chapter 4.

3.6 Detection

In a track-by-detection approach, the first step is generation of detections. For this system, this problem is considered as a clustering problem: given all of the points from the background-removed global point cloud, how can those points be clustered together to form a detection for a specific object or person. In previous iterations of this system, such as [64] and [76], this clustering was performed based on Euclidean distance. Points within an ϵ distance of each other qualified as connected, and all connected points were grouped as an object. The method was straightforward but relied upon an unrealistic assumption of clear physical separation between objects. In addition to requiring strong assumptions, the method is also computationally slow.

Clustering is a topic that has been well researched and many solutions exist, however each method has advantages and disadvantages. For instance, the most popular clustering technique, k-means[77], requires the number of cluster centers to be know *a priori*. While k-means would be a quick and efficient method, the requirement of knowing the number of cluster centers is not suitable in this situation, since the number of clusters (people) will be unknown and can fluctuate over time.

Another popular clustering method is spectral clustering [78]. In spectral clustering, the input is a weighted undirected graph, $G = (V, E)$, with vertices V, edges E and weight matrix $W = [w_{ij}]$, where $w_{ij} = \text{sim}(v_i, v_j)$ for some similarity measure sim. This graph is then partitioned into a series of subgraphs. The method first takes the graph Laplacian, $L = D - W$, where D is the degree matrix, a diagonal matrix where $d_{ii} = \sum_j w_{ij}$, and then computes the top k eigenvectors of L. Let U be a matrix whose columns are the top k eigenvectors, the rows of U can be considered points in $\mathbb{R}^k$ and those points can be clustered with k-means to produce k clusters. This method has been widely used with good results, however there is still the issue of knowing how many clusters to have, as well as having to compute eigenvectors for a large matrix.

3.6.1 Hierarchical Graph-based Detection

Spectral clustering is intrinsically linked to the mincut problem, since relaxations of several normalized mincut methods are equivalent to spectral clustering using different graph Laplacians. As with spectral clustering, the problem is to find a partition of the graph that minimizes

$$\text{cut}(A_1, ..., A_k) = \frac{1}{2}\sum_{i=1}^{k} W(A_i, \bar{A}_i), \tag{3.9}$$

where A_i represents a partition of the graph, and $W(A, B) = \sum_{i \in A, j \in B} w_{ij}$ is a measure of connectedness between graphs A and B. Unfortunately, this formulation of mincut leads to poor solutions, since it generally just separates one vertex from the graph. This problem is solved by normalizing mincut, with one popular approaching being the normalized graphcut, Ncut, which was popularized for image segmentation in [79]. Here, the problem is formulated as

$$\text{Ncut}(A_1, ..., A_k) = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \bar{A}_i)}{\text{vol}(A_i)}, \tag{3.10}$$

where $\text{vol}(A_i)$ is the volume of the graph, measured as the sum of the weight of the edges in A_i. While these formulations are nice since the number of clusters is incorporated in the minimization, solving these normalized graph cut problems is NP hard [80]. In [79], an approximation is presented that requires solving an eigenvalue problem. The work of [81] presents a different approximation that does not require the expensive operation of computing eigenvalues. This technique forms the basis of a clustering method presented here.

The graph is formed by finding the ϵ-nearest-neighbors to each point in the global point cloud. In practice, ϵ is set to 8, a value that was found empirically. Vertices in the graph represent points, and edges are computed as

$$v_i = [\, X, Y, Z, \, R, G, B \,], \tag{3.11}$$

$$w(v_i, v_j) = \|v_i - v_j\|_2, \; \forall \; i, j \in E. \tag{3.12}$$

This graph is initially segmented using the efficient graph-based method proposed in [81]. Prior to forming the initial graph, various filters are applied to reduce noise; these are detailed in [64].

The results of this segmentation yield small clouds, on the order of ten to a few hundred points, referred to as supervoxels. The graph-based segmentation method minimizes intra-class variance while maximizing inter-class variance so supervoxels are clusterings in space with very similar local color distributions (*cf.* Figure 3.14). This supervoxel approach is taken to provide better clusterings of the points. Without an *a priori* model of the clusters, it is impossible to perform perfect clustering using purely bottom-up, low-level features. This allows for the creation of small heterogeneous clouds, which can then be further clustered.

The supervoxels next undergo a filtering process to remove noise. They are first filtered for sufficient size, currently set to ten points. Additional filtering using optical flow was tested (Bruhn's method, [82]), but detection of active children was not greatly improved while inactive children could be lost.

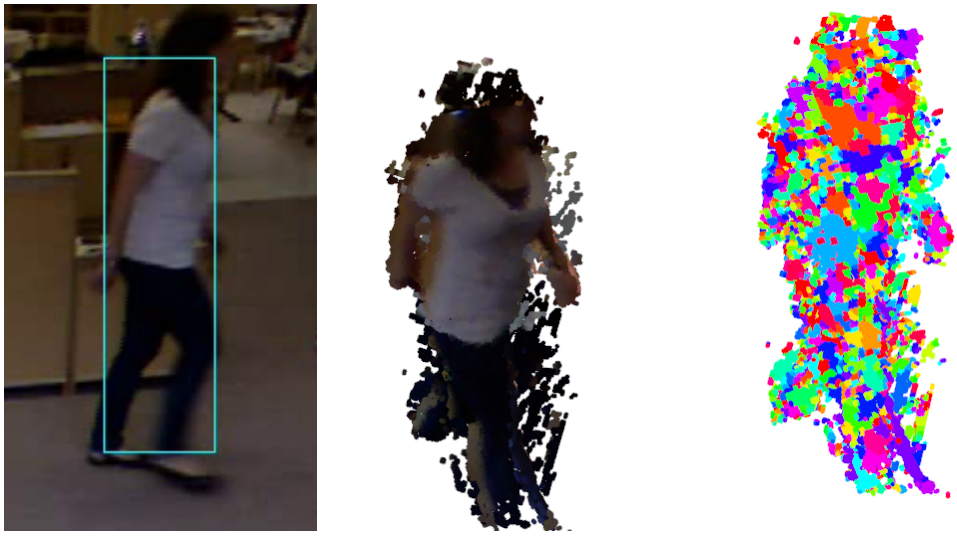
The remaining supervoxels are then subjected to a second round of graph-based segmentation. Here, the edge weights between supervoxel S_1 and supervoxel S_2 are defined as

$$v_i = [X, Y] \tag{3.13}$$

$$w(S_1, S_2) = \max \|v_i - v_j\|_2, \quad \forall v_i \in S_1, \forall v_j \in S_2. \tag{3.14}$$

The graph is again segmented, using the aforementioned method, and resulting clusters are accepted as objects of interest. The resulting objects are subjected to second-round filtering, again based on object point cloud size. This method was first described in [70], and was used in subsequent work [75]. Throughout this work, this method will be referred to as HGB, for hierarchical graph-based.

While this graph-based approach proved to be effective, there were still issues with the approach. Given the size of the point clouds, even efficient approximations to the max-cut problem are still computationally intense. Additionally, graph based methods do not apply any geometric constraints upon the computation of the supervoxels. For instance, ideal supervoxels should not cross object boundaries (*i.e.*, all points in a single



(a) *RGB (Sensor 1)* (b) *Noisy 3D reconstruction* (c) *HGBSupervoxels*

Figure 3.14: Images illustrating supervoxel generation using hierarchical graph-based detection. For reference, (a) shows the detection bounding box from one of the five views, (b) is one perspective from the merged global 3D point cloud for this person, and (c) shows the computed 3D supervoxels visible from the same viewpoint. A connected color blob denotes a single supervoxel. (Colors may get reused however for displaying supervoxels that are fully separated and disconnected.)

supervoxel should belong to only one object) and they should tend towards continuity. In light of this, a new method to compute supervoxels has been employed, Voxel Cloud Connectivity Segmentation (VCCS) ([83]), which has more stringent geometric constraints and better meets the previous criteria.

3.6.2 VCCS Detection

Supervoxels are constructed by clustering voxels from a voxelized version of the input point cloud where each voxel is $4cm^3$ (this is a distinct voxelization from that used for background subtraction). From this voxelization, an adjacency grid can be created, which will be used as part of the supervoxel creation process. This can be done efficiently by creating a kd-search tree on the voxelized point cloud and then for each voxel finding voxels whose centers are within $\sqrt{3} * 4cm$ of the input voxel. To start the supervoxel creation procedure, initial supervoxels must be defined. This is done by defining a voxel

grid with a seed resolution of $30cm^3$, then for each voxel in this rougher voxelization the voxel from the finer $4cm^3$ voxelization that is closest to the center of the voxel is chosen as a seed.

Once seed points have been defined, supervoxels are formed by clustering in a 39 dimensional feature space, defined as

$$\mathbf{F} = [x, y, z, L, a, b, FPFH_{1..33}], \tag{3.15}$$

where x, y, z are the spatial coordinates, L, a, b are color in CIELab space, and $FPFH_{1..33}$ are the 33 elements of Fast Point Feature Histograms (FPFH), a local geometrical feature proposed by Rusu *et al.* [84]. The clusterings are formed by starting at the voxel nearest the cluster center and flowing outward to adjacent voxels. The distance is computed between the current voxel and the supervoxel center and if the distance is the smallest the voxel has seen, its label is set. Once all adjacent voxels have been checked for the current supervoxel, the algorithm proceeds to the next supervoxel, such that each level outwards from the center is considered at the same time for all supervoxels. This yields two attractive properties of supervoxels:

1 The supervoxel labels cannot cross object boundaries that are not adjacent, because only adjacent voxels are considered

2 Supervoxel labels will tend towards continuity, since labels flow out from the center of each supervoxel.

An example of object detection can be seen in Figure 3.15, which depicts the detection on the image plane, in the point cloud, and also indicates the supervoxel membership. The quality of the supervoxels can be compared between this detection method and HGB by comparing Figures 3.15 and 3.14. Particularly, the VCCS voxels tend to be larger and and subsequently conform better to different parts of the detected person.

Once the supervoxels are created and filtered, the remaining supervoxels are again clustered to create detections. Our previous work used a second pass of graph-based clustering with edge weights based on the maximum distance between two points between two supervoxels. This relied on the assumption that a standing person would have a dense cluster of points when projected onto the X-Y plane, however this caused issues

(a) *RGB (Sensor 1)* (b) *Noisy 3D reconstruction* (c) *VCCSSupervoxels*

Figure 3.15: Images illustrating supervoxel generation using VCCS. For reference, (a) shows the detection bounding box from one of the five views, (b) is one perspective from the merged global 3D point cloud for this person, and (c) shows the computed 3D supervoxels visible from the same viewpoint. A connected color blob denotes a single supervoxel. (Colors may get reused however for displaying supervoxels that are fully separated and disconnected.)

with situations like adults leaning over children. Instead, the density-based clustering method DBSCAN ([85]) was employed. Distance between supervoxels is computed as the L_2 spatial distance between supervoxel centroids.

Still viable resulting clusters then undergo a third and final round of filtering based on number of points: clusters with fewer than 3000 points are removed. The final foreground clusters are then considered as object detections that will be used in further stages of the processing pipeline. Throughout this work, this detection approach will be referred to as VCCS, since it utilizes the VCCS method for computing supervoxels.

3.7 Appearance Descriptors

Once objects are detected, additional information is required to help track and re-identify the object. While the motion of the object is sufficient to track it in situations without very much clutter, when many room occupants are playing together in a small area, additional methods are necessary to disambiguate detections and also to potentially split or merge detections. Additional descriptors can be computed that describe

the appearance or shape of the detection, so that it is easier to match tracks to detections, and also so that tracks can be merged if an occupant leaves the observed area. Many options exist for such descriptors, both from a traditional RGB camera perspective and also from a 3D color point cloud perspective.

3.7.1 Color Histograms

One straight-forward and common approach is to use color histograms [86]. A histogram is a way to represent the distribution of values in a function, and if normalized can approximate the probability distribution of those values. Formally, for a discrete function $f : x \to V$, where $x \in X$ and $v \in V$, a histogram of f accumulates the occurrences of values in the range of f. Thus, a histogram can be defined as $h_f : v \to Z^*$, where $v \in V$ and Z^* is the set of non-negative integers and $h_f(v)$ is the number of elements $x \in X$ such that $f(x) = v$.

Histograms have a number of properties that make them attractive as a descriptor. First, since they discard information about the domain, it means that they are invariant to any one-to-one transformation of the domain of the original function. Additionally, they are fast to compute, requiring just simple accumulation in a single pass over the data. It is also easy to control the size of a histogram descriptor by varying the bin size.

The drawback of histograms is that they throw away spatial information. The accumulated bins throw away information about neighboring values, so if co-occurring values are important, that information is lost. Additionally, discretization error can be introduced if bin sizes are too large, so tuning the proper bin size can be important. Histograms are also not amenable to Euclidean distance; other more appropriate similarity measures can be used, but often different similarity measures perform better in different applications making the appropriate distance measure difficult to ascertain.

Performance of joint color histograms are explored in Chapter 4. These joint histograms are computed using the red, green and blue color values from each point in an object's point cloud. Since the illumination levels in the classroom fluctuate, the hue, saturation, value (HSV) colorspace was also considered, since the global illumination changes should only affect the illumination (value) channel, rather than the hue and saturation channels.

Two similarity measures were employed in this work. The first is based on the

histogram intersection kernel, which is defined as

$$s(x, y) = \sum_{i=1}^{n} \min(x_i, y_i), \tag{3.16}$$

where x and y are two histograms with n bins. This is also sometimes referred to as the min kernel, as it's the summation of the min value between each bin in the two histograms. This similarity measure has been shown to perform well in image classification tasks.

Another way to measure similarity between two histograms is based on the chi-square kernel, which is defined as

$$s(x, y) = 1 - \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{\frac{1}{2}x_i + y_i}, \tag{3.17}$$

where x and y are two histograms with n bins. However, this kernel is only conditionally positive-definite, whereas a positive definite version is given in [87] as

$$s(x, y) = \sum_{i=1}^{n} \frac{2x_i y_i}{x_i + y_i}. \tag{3.18}$$

3.7.2 Region Covariance Descriptors

While histograms can describe appearance in terms of color, it could be advantageous to combine multiple different facets into one compact descriptor. All of this heterogeneous information can be efficiently and compactly encapsulated within a region covariance descriptor (RCD). These descriptors were first introduced by Tuzel *et al.* in [39] as an appearance descriptor for textures, and they were originally applied to tracking in [40]. Region covariance descriptors are computed by defining a feature vector which is computed for each point in a region of interest and then computing a covariance for that feature vector across all points. Our feature vector is defined as

$$\begin{aligned} f_s(p) = [&X,\ Y,\ Z,\ R(p),\ G(p),\ B(p),\ \partial_x I(p),\ \partial_y I(p),\\ &\partial_{xx} I(p),\ \partial_{yy} I(p),\ \partial_{xy} I(p),\ ||\partial I(p)||,\ atan(\partial_y I(p), \partial_x I(p)], \end{aligned} \tag{3.19}$$

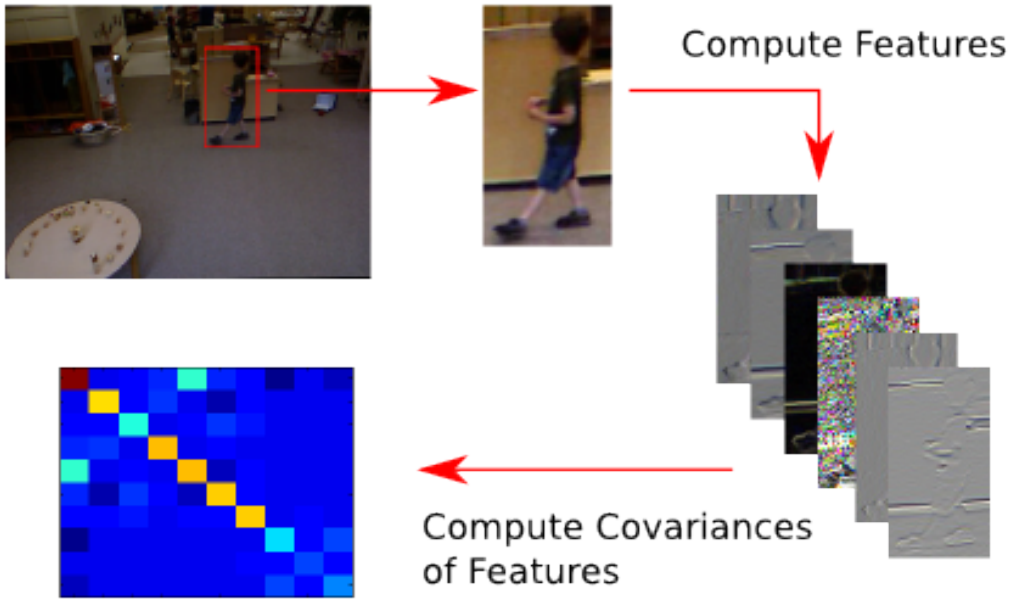


Figure 3.16: Outline of the procedure for computing a covariance descriptor. For a given region of interesting, a series of feature maps can be computed. This creates a feature vector for each pixel in the region of interest, and a covariance can be computed on the set of all of those feature vectors. This yields a compact descriptor in the form a $D \times D$ matrix, where D is the number of features computed.

where p represents a point from an object cloud situated at $\{X, Y, Z\}$ in the global reference frame. This point is projected back to the image plane for sensor s, where the color and intensity image derivative information is collected. This feature vector will yield a 13×13 covariance descriptor for each sensor that observes the detected object. An outline of this procedure is provided in Figure 3.16. Non-singular covariance matrices are symmetric positive definite matrices that reside in the space Sym_d^+ whose distance between two matrices is not accurately defined by the Euclidean distance. One efficient approximation of distance between covariance matrices is the Log-Euclidean metric [88],

$$d_{LE}(P, Q) = \|log(P) - log(Q)\|_F, \tag{3.20}$$

where $P, Q \in Sym_d^+$. This approximation was chosen for computational efficiency as $log(P)$ can be precomputed and used for several comparisons.

In addition to computing an appearance descriptor across the whole object point cloud, the cloud could be split into different components with a descriptor computed for each component. Early human tracking systems, such as Pfinder [36] used this approach with color histograms, splitting the region of interest into layers and computing a color histogram for each layer. A similar approach could be followed here, however there is a question of what axis to perform these splits along. With pedestrians, an assumption can be made that the camera is roughly perpendicular to the ground plane which pedestrians are walking upon, and so it makes sense to split layers along the vertical axis of the image plane. A similar assumption could be made in 3D by layering along the vertical Z-axis. However, since this system does not work with pedestrians, the Z-axis is not guaranteed to be the best axis to split along. Instead, principal component analysis (PCA) can be used to find the axis of highest variance for a given point cloud. Unfortunately, this requires some relatively computationally expensive operations to be performed. Experiments using this methodology will be presented later, however the benefits did not outweigh the computational costs.

A more natural way to perform these splits is to directly use the supervoxels computed during the segmentation process. These supervoxels were designed to agglomerate homogeneous regions of color adhering to geometric constraints. In theory, these supervoxels should encapsulate some meaningful information about the object they are a part of. Additionally, since they are already computed as part of the detection process, there is no additional computational overhead to use them when computing descriptors, other than the cost of additional descriptor computation.

3.8 Tracking

Tracking of detections (individuals) in the scene was performed using a Kalman filter. The Kalman Filter is a method for fusing *a priori* information about a dynamic system (the system model) with measurements to achieve an optimal estimation of system state. The Kalman Filter is the optimal estimator under linear (*i.e.*, a linear dynamic system) and Gaussian assumptions (i.i.d. Gaussian noise). Given a discrete-time stationary

dynamic system with stationary noise,

$$x_t = Fx_{t-1} + w \tag{3.21}$$

$$y_t = Hx_t + v, \tag{3.22}$$

$$E(ww^T) = Q, \tag{3.23}$$

$$E(vv^T) = R, \tag{3.24}$$

$$E(wv^T) = 0, \tag{3.25}$$

the Kalman Filter is defined as

$$P_t^- = FP_{t-1}^+F^T + Q \quad \text{Predicted (\textit{a priori}) estimate covariance} \tag{3.26}$$

$$\hat{x}_t^- = F\hat{x}_{t-1}^+ \quad \text{Predicted (\textit{a priori}) state estimate} \tag{3.27}$$

$$K_t = P_t^- H^T(HP_t^- H^T + R)^{-1} \quad \text{Optimal Kalman gain} \tag{3.28}$$

$$\hat{y}_t = y_t - H\hat{x}_t^- \quad \text{Innovation} \tag{3.29}$$

$$S_t = HP_t^- H^T + R \quad \text{Innovation covariance} \tag{3.30}$$

$$\hat{x}_t^+ = \hat{x}_t^- + K_t(\hat{y}_t) \quad \text{Updated (\textit{a posteriori}) state estimate} \tag{3.31}$$

$$P_t^+ = (I - K_tH)P_t^- \quad \text{Updated (\textit{a posteriori}) estimate covariance,} \tag{3.32}$$

with initial values

$$\hat{x}_0^+ = E(x_0), \tag{3.33}$$

$$P_0^+ = E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T]. \tag{3.34}$$

The state for this system consists of 3D position, velocity and shape. The point cloud corresponding to each detection was used to create a 3D centroid (position) - $\mu = [\mu_x, \mu_y, \mu_z]^T$. The shape of a detection was modeled by an ellipsoid, represented by the 3×3 spatial covariance of the points in the point cloud:

$$\Sigma = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_z^2 \end{bmatrix}. \tag{3.35}$$

The state vector $\mathbf{x}$ in the Kalman filter consists of the centroid position μ, the velocity of centroid $\dot{\mu} = [\dot{\mu}_x, \dot{\mu}_y, \dot{\mu}_z]^T$, and the 6 independent parameters in the spatial covariance:

$$\mathbf{x} = \left[\mu_x, \mu_y, \mu_z, \dot{\mu}_x, \dot{\mu}_y, \dot{\mu}_z, \sigma_x^2, \sigma_y^2, \sigma_z^2, \sigma_{xy}, \sigma_{xz}, \sigma_{yz}\right]^T. \tag{3.36}$$

In addition to the 3D location of objects tracked, the 3D shape is also of interest for purposes such as size characterization (tall vs short, a good indicator of child vs adult), as well as filtering out objects that are unlikely to be people. Observations consist of position and shape parameters. The state transition matrix Φ and observation matrix H were given by

$$\Phi = \begin{bmatrix} \mathbf{I}_3 & \mathbf{I}_3 & \mathbf{0}_{3\times 6} \\ \mathbf{0}_{3\times 3} & \mathbf{I}_3 & \mathbf{0}_{3\times 6} \\ \mathbf{0}_{6\times 3} & \mathbf{0}_{6\times 3} & \mathbf{I}_6 \end{bmatrix}, \tag{3.37}$$

$$H = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 6} \\ \mathbf{0}_{6\times 3} & \mathbf{0}_{6\times 3} & \mathbf{I}_6 \end{bmatrix}. \tag{3.38}$$

The process noise covariance Q was assumed to be uncorrelated with a differing parameter for position, velocity and shape. Likewise, the measurement noise covariance R was also assumed to be uncorrelated with differing parameters for position and shape. The process noise parameters were empirically set to $\sigma_p^Q = 0.5$, and $\sigma_v^Q = 0.75$ and $\sigma_s^Q = 1$ for position, velocity and shape respectively. The measurement noise parameters were empirically set to $\sigma_p^R = 0.55$, and $\sigma_s^R = 0.1$ for position, and shape respectively. In all of the above, $\mathbf{I}_n$ refers to the $n \times n$ identity matrix and $\mathbf{0}_{m\times n}$ refers to the $m \times n$ matrix of all zeros.

3.8.1 Track/Observation Matching

Two fundamental issues with tracking by detection arise in track/observation association and track management. To deal with the the association problem, the appearance descriptors are leveraged. In each frame, a decision has to be made about which observation to match to which tracker. This is solved by computing a dissimilarity score between each track and observation. The dissimilarity score used is

$$D(tr, obs) = D_{pos}(tr, obs) + \alpha D_{shape}(tr, obs) + \beta D_{app}(tr, obs) \tag{3.39}$$

$$D_{pos}(tr, obs) = \sqrt{(obs_{pos} - tr_{pos})^T \Sigma_{pos} (obs_{pos} - tr_{pos})} \tag{3.40}$$

$$D_{shape}(tr, obs) = \sqrt{(obs_{shape} - tr_{shape})^T \Sigma_{shape} (obs_{shape} - tr_{shape})} \tag{3.41}$$

$$D_{app}(tr, obs) = \max_{i,j} || \log tr^i_{app} - \log obs^j_{app} ||_F \tag{3.42}$$

This can be summarized as a weighted combination of the maximum log Euclidean between appearance descriptors and the Mahalanobis distances between the 2D position and shape parameters of the tracker and observation. The log Euclidean distance for appearance descriptions consists of Frobenius norm on the difference between the matrix logarithms for the RCDs for the tracker and observation. The covariances for the Mahalanobis distances, Σ_{pos} and Σ_{shape}, come from the covariance in the Kalman filter. The parameters α and β were set empirically and used to shift the scale of each term so that none of the terms overpowered the others. Dissimilarity scores that exceed a threshold are removed from consideration. The associations are then made in a greedy fashion. Experiments were performed using the Munkres algorithm [38], although in practice the greedy method worked just as well.

3.8.2 Track Management

Track management is handled by computing a track score based on a log-likelihood ratio of whether a track is a true target or a false alarm ([34]). This track score is used to determine when tentative tracks (tracks created from unmatched observations) become confirmed tracks (those tracks which are considered to correspond to a true target), and when tracks should be deleted. The score is computed by recursively summing log-likelihood contributions from the process model and sensor model, or a reduction if a track is not matched on a particular timestep. The initial track score for a track is a function of position: if tracks are formed near the entrances/exits to the room, then the track has a far higher initial track score than if the track appears elsewhere. One issue with track scores is that, if the scores are allowed to grow unbounded, then it can be impossible to remove tracks that have persisted for a large number of time steps. To

prevent this, a maximum ceiling of 15 is imposed on the track score.

3.8.3 Observation Splitting

Given the nature of the classroom, occupants are frequently in very close proximity, or even touching, each other. This can lead to one detection consisting of several merged detections, which then causes issues with tracking. This is overcome by a mechanism to split apart these merged detections. Since the final clustering step in creating detections is based on spatial density, it's possible for two detections that are close in space to get merged into a single detection. To deal with this, the tracking portion contains a mechanism to split apart these merged detections. Once a track has been established, it is unlikely to completely vanish, so, for a given frame, if a track does not have a matched observation, there is evidence of a merged detection. A single observation is considered to consist of multiple merged detections if it meets the following criteria:

$$D(tr_m, obs) - D(tr_u, obs) < T_s \tag{3.43}$$

$$D(tr_u, obs) < T_d, \tag{3.44}$$

where tr_u is an unmatched tracker (a tracker without a matched observation), and tr_m is the tracker that is matched with *obs* which is the observation with the smallest distance to tr_u. That is, an observation *obs* is considered to consist of multiple merged detections if it is the closest match to an unmatched tracker and its distance is less than a threshold, and the distance between *obs* and it's matched tracker is less than a threshold.

Additionally, bounding volume information is considered. For a tracker without a matched observation, the bounding volume from the previous matched observation is updated using the tracker's velocity information. This bounding volume is compared with the bounding volumes of the observation with the smallest distance. If this bounding volume overlap is below a threshold, and the distance is below a threshold, the observation is considered to consist of merged detections.

If any observations are marked as consisting of multiple detections, they are then split and a new association is computed. Detections are split using K-means [77] on

spatial distance, where K is taken to be the estimated number of detections contained in the single detection, as described above.

3.9 Person Reidentification

While the system as described tracks occupants well, with some tracks persisting for several minutes despite occlusions, there are some unavoidable situations that can lead to distinct tracks for the same individual. Occupants that leave the observed area for a period of time and then return will result is one example. Furthermore, conceptually the system should operate over multiple days and weeks, in order to gather long-term behavior information. This creates a difficult across-days matching problem.

This creates difficult re-identification challenges. Occupants may change appearance while not tracked. This is obvious for across-day matching, as apparel will change. However, even within the same day children may change apparenl – for example when preparing to go outside, coming back in from outside, or from costumes as part of a dress-up game.

Facial recognition provides an appealing method for performing re-identification, as faces are something that will not change on a day-to-day basis. However, facial recognition presents challenges as well. First, the resolution of the sensors and distance of the occupants means that the resolution of the occupant faces is generally not very high. Also, there may be situations where room occupants are not positioned for the sensors to get a good view of faces (although this is partially mitigated by having multiple sensors). Because of these issues, facial recognition has not yet been explored with this system, although it could be a suitable possibility for future directions.

Instead, experiments have been performed using the same descriptors that were explored to aid in tracking. As an individual is tracked over time, all of the per-frame appearance descriptors are collected, and can then be used to build an appearance model for that individual. These appearance models can then be compared between each track and based on this affinity different tracks could be grouped together.

However, this still presents some challenges. Performing an all-to-all descriptor comparison between two tracks is computationally expensive, especially considering that

each track could have thousands of descriptors which could require eigenvalue decompositions (such as with the Geodesisc distance for covariance descriptors). Additionally, some method for dealing with the resulting affinity matrix must be devised. This could be as simple as taking a min, max or average of all the of distances, however each of these configurations should be tested.

Instead of an all-to-all comparison, a more compact model could be constructed. A simple example of this would be computing a single mean descriptor from all of the descriptors for an individual track. Then, only one descriptor needs to be compared between each track.

The downside of this approach is that a single mean descriptor may not be descriptive enough – it may not properly model all of the variations of appearance that could be present within the track. An example could be when a child, while being tracked, dons a costume – or removes a previously acquired costume. Instead, the space spanned by the various feature descriptors needs a more complex model.appearance.

3.9.1 Dictionary Learning

One approach to creating a more complex model is to create a collection of representative or basis vectors that describe the feature space. This is essentially a linear regression approach, $D\alpha = x$, where x is a vector that is modeled by D using some encoding α. In this sense, D is referred to as a dictionary which is a set of basis vectors that can be linearly combined into a weighted sum to describe another vector.

While it is possible to use a prescribed set of functions to define a dictionary, learning the dictionary can yield a data-driven result that is much better at descibing the space of the data. Classical dictionary learning approaches begin with a finite training set of vectors, $X = \{x_i\}_{i=1}^{n}$ where n is the number of vectors of length m. A dictionary is found by optimizing the function:

$$f(D) \triangleq \frac{1}{n} \sum_{i=1}^{n} l(x_i, D), \tag{3.45}$$

where $D \in \mathbb{R}^{m \times k}$ and l is a loss function. Ideally, the loss function will be small if D is a ‘good‘ representation of the input vectors.

The use of sparsely coded overcomplete learned dictionaries is a well-known approach

that has met with great success. Early work [89] theorized that this may be an approach used by V1, the primary visual cortex of the brain. Additionally, it has been applied to image denoising [90], mosaicking, inpainting [91] and image classification [92, 93].

One potential formulation of this problem is

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\mathbb{R}^{k\times m},D}{\text{minimize}} \quad & \sum_{i=1}^{n} \|x_i - D\alpha_i\ |_2^2 + \lambda\|\alpha_i\|_1 \\ \text{subject to} \quad & d_j^T d_j \leq 1 \quad \text{for all} \quad j = 1,\ ...,\ k. \end{aligned} \tag{3.46}$$

This results in a joint optimization problem in D (the dictionary learning aspect) and α (the sparse coding aspect). The constraints on the dictionary atoms are important to prevent the dictionary from growing arbitrarily large and hence making $\boldsymbol{\alpha}$ arbitrarily small. Since this joint optimization is nonconvex, it is commonly solved by holding one parameter fixed while optimizing for the other parameter and then alternating between the two parameters.

Significant effort has been spent on considering different variations of the loss function (such as using the l_0 norm instead of the l_1 norm for the regularizer, which affects sparsity) and approachs to solving the optimization [94], [95], [96]. Such a learned dictionary encodes information from all of the input vectors and can create lossy reconstructions of those vectors using a sparse representation. Additionally, unlike with decompositions based on principal component analysis and its variants, the learned dictionaries do not impose that the basis vectors be orthogonal, which allows more flexibility for the representation to be adapted to the data.

Commonly, these dictionaries are used for classification, where a dictionary is learned for each class. In [97] Guha and Ward explore different methods for classification using dictionaries determined for sparse representation of a signal. Rather than using dictionaries for classification, in this thesis the goal is instead to summarize each track with a learned dictionary and compute simlarities between track dictionaries to see if they should be clustered. This is more a clustering problem than classification, similar to the work described in [98].

In this approach, a dictionary D_k is learned for each track using the online dictionary learning framework of [96]. For each track k with n_k observations, the descriptors for each observation can be vectorized to create the set $X_k = \{x_{j,k}\}_{j=1}^{n_k}$. This feature set is

then used to learn a dictionary, D_k, for each track.

For comparing two dictionaries, a standard similarity measure is defined as

$$S(D_i, D_j) =< D_i, D_j >= tr(D_i^T D_j), \tag{3.47}$$

where $< \cdot, \cdot >$ represents the matrix inner product, which is the same as the trace of the the product of the two dictionaries. In essence, this is the sum of the cosine similarity between dictionary atoms.

One issue, however, with this similarity measure is that it imposes a structure on the dictionary atoms: the first atom in D_i is always compared with the first atom in D_j. Since the atom order is arbitrary, this work instead uses the maximum element value from the matrix product $D_i^T D_j$, which returns the cosine similarity of the two most similar atoms. Using this methodology, dictionaries can be computed for covariance descriptors by vectorizing the matrix, or can be learned directly on histograms.

For each method of comparing a descriptor, there is also a question of how to link tracks once an affinity matrix between all tracks is computed. This becomes a clustering problem, as a decision must be made about which tracks to link together, or which tracks should remain distinct. The most straight-forward approach to this would be to make these determinations by hand, which is an approach that was done in order to create groundtruth for experiments. Given the number of occupants in the classroom, this could remain an acceptable approach, although adding automation is attractive. In addition to a purely manual approach, using the appearance descriptors to make recommendations would greatly speed up manual-matching. Beyond the manual approaches, experiments were performed with common clustering techniques such as k-means and spectral clustering.

Chapter 4

System Validation

Chapter 3 describes the overall system, from how the data is generated and processed through to the outputs of the system. This chapter deals with how to evaluate the performance of each component of the system. Additionally, this chapter contains some justifications for the approach taken for different system components. While this chapter presents system validation prior to behavioral analysis, it is important to note that system performance is cumulative. Poor performance in earlier steps of the system can cause issues down the line. Additionally, missed detections and mislabeled tracks reduce the reliability of input data into the behavioral analysis components, so validating the system prior to that work is imperative.

The first several sections of this chapter cover how the groundtruth is generated and also how performance is measured. This includes Section 4.1, which describes how groundtruth data is generated, Section 4.3 which describes how the system is evaluated on a per-sensor basis, and Section 4.4 which describes how the tracking portion of the system is evaluated. These performance metrics are then used in the latter part of the chapter, which describes the actual results from these evaluations.

4.1 Groundtruth Generation

In order for this system to be validated, a method for creating groundtruth data needed to be developed. The necessary groundtruth data is multi-faceted: it is necessary to know the location of individuals in a given RGB frame, it is also necessary to track

the identify of these individuals across frames, and finally it is necessary to know the position of these individuals in 3D. By having this information, the ability to detect and localize classroom occupants can be measured and additionally the ability to track the identity of occupants can also be measured. All of these aspects are important for creating confidence in the behavioral analysis aspects.

A straight-forward way to represent this information is to create a segmentation mask on the RGB image. This segmentation mask gives a pixel-level indication of the identify of an occupant: each pixel in the mask is labeled as either a 0, which indicates that pixel does not belong to an occupant, or it can consist of non-zero id that indicates that pixel belongs to a specific individual. These ids are kept consistent throughout all labeled frames so that the identify of each labeled individual is kept consistent. An example of labeled segmentation masks can be seen in Figure 4.1.

While the creation of this groundtruth labeling program greatly aided in the generation of groundtruth, it still takes a significant amount of time to label each frame and adjust the mask such that it is accurate. Considering that this needs to be done across 5 sensors generating 30 frames per second, it was decided not to label each individual frame. Instead, frames were labeled every 2 seconds, or 60 frames. The underlying assumption is that the positions of room occupants are not likely to change drastically within 2 seconds, and so by validating data at a period of 2 seconds still yields sufficient validation of the system. The groundtruth labeling program was designed to facilitate this, by allowing a frame skip parameter to be set.

Using this groundtruth labeling program, three consecutive recordings from the Shirley G. Moore Laboratory School have been labeled for groundtruth. These recordings are the first three in a set of eleven recordings that were taken in the morning before class began, during a free-play session. During this time, children were arriving for the day, and were allowed to play freely until the start of class. This leads to a highly challenging unstructured environment. The remainder of this chapter will present groundtruth validated results using these first three recordings, referred to as recording 1, 2 and 3. Recording 1 consists of 7853 frames, recording 2 consists of 8093 frames, and recording 3 consists of 6673 frames, for a total of 22619 frames for validation.

Recordings were split into approximately 5 minutes of data due to restrictions on the OpenNI API. The version that was used had a 32-bit value to keep track of frame

(a) *Original RGB Images*

(b) *Images with Segmentation Masks*

Figure 4.1: Images illustrating the segmentation masks created with the groundtruth labeling program. The first figure shows the original RGB images from 3 sensors, while the second figure shows the original images overlaid with the labeled segmentation masks. Segmentation masks are color coded based upon the ID of the person the mask belongs to.

locations within the file. This means that only 4GB of data could be addressed and after 5 minutes the recorded file sizes grew past 4GB, which then caused failures in playing back the recorded files.

Since the RGB and depth images are registered, any segmentation mask from an RGB image can also be applied to the corresponding depth map. This allows only groundtruth information to be used when generating point clouds and the generated points can also be labeled with the identify of the individual that they come from. By using this, groundtruth point clouds can also be created.

To be able to create these segmentation masks in a reasonable amount of time, a program had to be developed. Because data is recorded from multiple sensors simultaneously, this groundtruth labeling program was designed to be able to handle multiple RGB videos simultaneously. The program can take N input videos and display the current frame for each video. Each of these images can be annotated by the user to

create the appropriate segmentation mask. To aid in the initial creation of a mask for an individual, the interactive segmentation method GrabCut [99] was used.

To create a segmentation mask, a user first annotates a region of interest with a rectangle. Once the region of interest has been created, the user can mark pixels as belonging to either foreground or background and once enough pixels are marked, a round of GrabCut can be run, which will compute a boundary between the foreground and background. Frequently, the initial mask created by GrabCut will need to be adjusted, so this marking and mask generation procedure can be iterated upon.

Still, the final resulting mask is not always precise and because of this the groundtruth labeling program also allows the user to manually adjust the mask. Users have a brush, whose size can be adjusted, that can be used to set the label of a pixel manually. The current label can be set to 0 to clear a mask, or it can be set to the desired ID for a specific individual. This manual adjustment is often required so that the mask lines up well with the depth image: a small error in the segmentation mask could lead to a large error in the groundtruth 3D position of the individual. This is because a depth pixel close to the boundary of a person could actually belong to an object much further away from the person. If that object is included in the groundtruth of the individual, it can make their bounding box much larger than it should be.

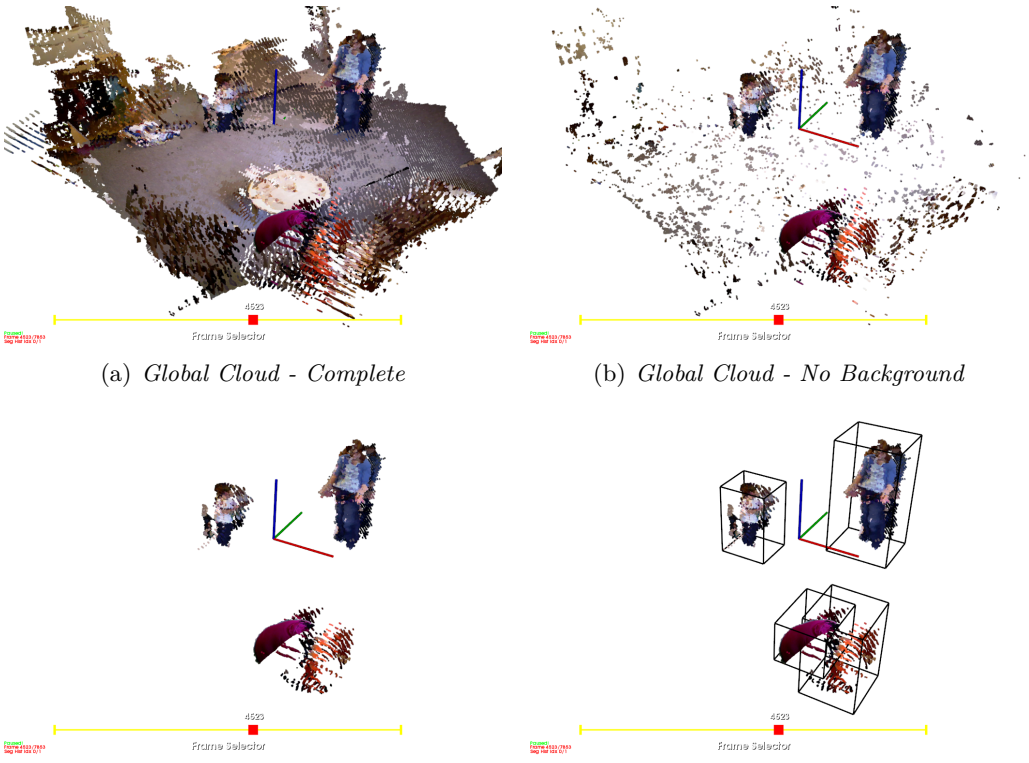
4.2 Data Visualization

An important aspect in validating the performance of this system is being able to visualize the results of different processing steps. Much of the data in this system is represented as 3D point clouds and often each point in the cloud has several pieces of metadata associated with it. Being able to visually verify that data fits expectations is important to ensuring that the system is operating as expected.

Because of this, significant effort was devoted to developing a custom 3D visualization program, which was internally referred as *display_fusion* since it was used for displaying a fusion of point clouds. This program has been used extensively in debugging the system, ensuring that the 3D visualizations of data accurately fit expectations. Initially, *display_fusion* was developed such that it can display point clouds from multiple sources, such as clouds from different sensors or clouds from different algorithm

configurations. In addition to displaying the point clouds, it could also display the RGB frame for a sensor at a particular frame, which was useful for determining the RGB data that generated the point cloud. Figure 4.2 shows an example of what the *display_fusion* program looks like without the RGB camera view, depicting an example global cloud and a global cloud with background removed as well as the objects detected in that particular frame.

As the system grew, so did the things that *display_fusion* could display. The program can toggle modes through displaying the global point cloud to only displaying the detected object point clouds. Bounding volumes can be toggled on/off for detected objects, and those bounding volumes can be labeled with the detected object id as well as the tracked object id. Object point clouds can be colored in a variety of ways,



(a) *Global Cloud - Complete*

(b) *Global Cloud - No Background*

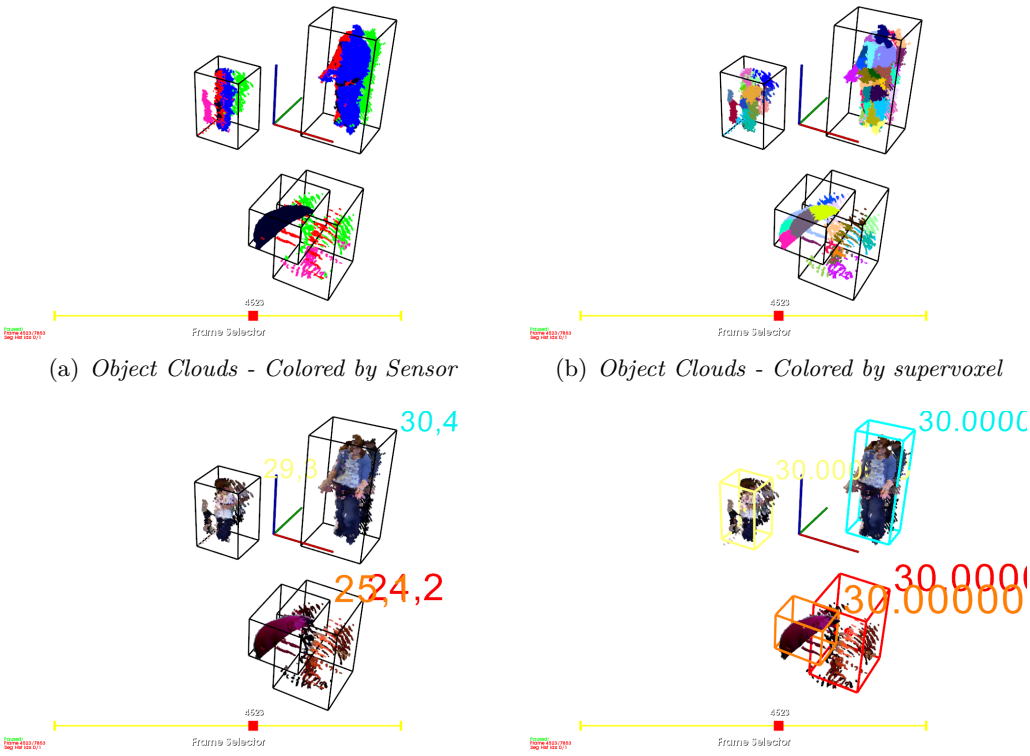
(c) *Object Clouds*

(d) *Object Clouds - Object Bounds*

Figure 4.2: Screenshots that show the 3D visualization from *display_fusion* for the global point clouds, first with no background removed and second with background removed.

from randomly, to being colored based on their object id or based on the sensor that generated the point. Coloring can also be based upon supervoxels, so that individual supervoxels for a detected object cloud can be visualized. Tracker state can also be visualized, such as the covariance, or the state of the tracker (position, shape, track score, *etc.*). Figure 4.3 shows some example screenshots from *display_fusion* that show examples of the different kinds of data that can be visualized.

In addition to all of the different ways to visualize data for an individual frame, the program contains a multitude of ways to control playback. A progress bar along the bottom indicates the current frame as well as the relative location of that frame within



(a) *Object Clouds - Colored by Sensor*

(b) *Object Clouds - Colored by supervoxel*

(c) *Object Clouds - IDs*

(d) *Object Clouds - Tracking State*

Figure 4.3: Screenshots that show the 3D visualization from *display_fusion* for various data about the object point clouds. (a) depicts the object clouds colored by the sensor that generated the points. (b) depicts the object clouds colored by their constituent supervoxels. (c) depicts object clouds with their tracked ID and detection ID. (d) depicts object clouds with their tracker state (centroid, shape, and track score, colored by ID).

the whole recording. The current frame indicator can be clicked and dragged to easily change the frame while there is also a way to enter the precise frame a user might wish to jump to. Finally, the recording can be played forward at 30FPS, or it can be played backwards. The program also allows the user to individually step forward or backwards, one frame at a time.

Besides *display_fusion*, numerous other visualization programs have also been written to facilitate drawing different file formats. For instance, a program was written to display individual frames of depth, or any other "raw" frame format that was written by the project. These frames could be colored in different ways, and also frames from two different depth or raw streams could be drawn side-by-side, optionally with a visualized frame difference. Overall, the multitude of different visualization tools created help give an indication that the algorithms within the system are performing as expected.

4.3 Evaluating Per-Sensor Performance

The object detection portion of the pipeline can be tested by creating segmentation masks and comparing them to groundtruth. To create segmentation masks, the points in the point cloud belonging to a detected object can be reprojected back into an image, which creates a mask that indicates which pixels belong to a particular person. A comparison of these masks to hand-labeled groundtruth masks can then broadly follow the performance detection measures of the PASCAL Visual Object Challenge (VOC) [100], where a bounding rectangle is computed for a detected image and then compared to a groundtruth bounding rectangle computed from a segmentation mask. The overlapping area, a_o, of these two bounding boxes is then calculated to be

$$a_o = \frac{\text{area}\,(B_p \cap B_{\text{GT}})}{\text{area}\,(B_p \cup B_{\text{GT}})}, \tag{4.1}$$

where B_p is the predicted bounding rectangle and B_{GT} is the groundtruth rectangle. If this a_o value exceeds 0.50, the detection is deemed a true positive (TP).

This metric can be used to compute true positives, TPs (computed values matched to groundtruth values), false positives, FPs (computed values with no matching groundtruth values) and false negatives, FNs (groundtruth values not matched with computed values).

True negatives, TNs, were ignored, since for this data the *vast* number of true negatives would completely drown out the other measurements (typically, with 100,000s of TNs per TP). To evaluate performance, values for precision, recall, and F_1 (the harmonic mean of precision and recall) are computed as

$$\mathsf{precision} = \mathsf{TP} \;/\; (\,\mathsf{TP} + \mathsf{FP}\,), \tag{4.2}$$

$$\mathsf{recall} = \mathsf{TP} \;/\; (\,\mathsf{TP} + \mathsf{FN}\,), \tag{4.3}$$

$$\mathsf{F}_1 = 2\mathsf{TP} \;/\; (\,2\mathsf{TP} + \mathsf{FN} + \mathsf{FP}\,). \tag{4.4}$$

4.4 Evaluating Tracking Performance

In addition to testing the person detection component, it is also necessary to evaluate the tracking performance. One popular set of metrics for evaluating a multiple object tracking (MOT) are the CLEAR MOT metrics [101]. The important considerations for such MOT metrics are to measure a tracker's ability to precisely determine object locations and consistently track object configurations. The CLEAR MOT metrics consist of the MOTP, or multiple object tracking precision, and the MOTA, or multiple object tracking accuracy. The MOTP is defined as

$$\text{MOTP} = \sum_{i,t} \frac{d_i^t}{\sum_t m_t}, \tag{4.5}$$

where d_i^t represents the distance between the i^{th} detection and its hypothesis in timestep t and m_t is the number of ground-truth detections in timestep t. This measures the total error in estimated position averaged by the number of matches, and indicates the ability of the tracker to estimate precise objection positions.

The MOTA is defined as

$$\text{MOTA} = 1 - \frac{\sum_t (FP_t + FN_t + MME_t)}{\sum_t g_t}, \tag{4.6}$$

where t indicates the timestep, FP is a false positive which occurs when a computed detection occurs with no corresponding groundtruth detection, FN is a false negative

which occurs when a ground-truth detection has no corresponding computed detection, MME is a mismatch error which occurs when the track ID for a tracker changes, and g_t is the number of ground-truth detections at timestep t. This metric accounts for all object configuration errors made by the tracker over all frames, and gives an intuitive measure of the tracker's performance.

4.5 Validating Background Subtraction

Using the evaluation methods outlined in Section 4.3, experiments were performed on roughly five minutes of data recorded during a normal class at the Shirley G. Moore Laboratory School. These validation experiments used recording 3. There are 5 people who are observed throughout the course of the test set; 3 children and 2 adults. The first child enters the scene from the playground and sits down to play at a table. Shortly after, 2 more children enter and start to interact with the first. Later, an adult comes in and sits down at the table to play with the first child. To perform experiments, video data was hand-labeled using the program described in Section 4.1.

First, results from per-sensor segmentation masks are considered. This is performed by running the processing pipeline through to the detection phase. The points from detected objects can be reprojected back to the image plane to create segmentation masks for comparison with groundtruth. By holding the detection method the same while varying the background subtraction methods, a suitable performance comparison can be made.

Three different background subtraction methods are considered: first image-based background subtraction (IMBGS), then voxel-based background subtraction (VBGS) and finally voxel-based with a color model (CVBGS). The image-based background subtraction method used was the multi-layer statistical approach, described in Chapter 3 Section 3.5. For computing object detections, the HGB method described in Chapter 3 Section 3.6 was used. For IMBGS, the foreground probability threshold was set to 0.12. This was found empirically and the results were not very sensitive to the value selected. Some experiments were run with Otsu's method [102] to automatically find a threshold for each frame, but this did not positively impact the results.

For the VBGS method, the voxel size was set to 10cm^3. This size provided a reasonable trade-off between removing background while not removing too much foreground. In general, as the resolution is increased more background is removed but there is also more foreground that is removed as well. For the CVBGS method, a larger resolution of 20cm^3 was used, since foreground points could be distinguished by their color distributions. Additionally, a threshold of 65 was applied to the Mahalanobis distance. These parameters were all found empirically; while they may not be completely optimal they illustrate the strengths and weaknesses of the different methods presented.

Table 4.1 shows the precision, recall, and F_1 for all five sensors across the different methods. In general, the precision score is hurt by the fact that the groundtruth only includes people while the detection method employed does not explicitly model people leading to false positives. Additionally, the HGB method is not as robust to noise as VCCS and so it generates more detections on bits of noise. There are ways to overcome these false positives such as using tracking information; if an object remains in the same location indefinitely it is clearly not a person. Alternatively, a model of a person could be built and detected objects could be compared to this model. Some experiments which incorporate person detection are presented in the next section.

In these experiments, the focus was instead on recall, which is a measure of, out of all the possible positive examples, how many were correctly detected. This is reasonable given the application since a missing detection is far more detrimental than having a false detection. For behavior analysis, which is the next step in the pipeline, it is possible to deal with false detections but missed detections are considerably more problematic.

From the table, it can be seen that the IMBGS method performs moderately well on several sensors, but is surpassed by the voxel-based methods. Note however, Sensors 2 and 3 have notably lower recall. This is a result of the activity in the room being further away than the recommended range of the Kinect sensors. To account for this, experiments were rerun with a depth limit placed on the groundtruth; groundtruth objects further away than the recommended range of the sensor were ignored. The results are presented as IMBGS-Depth, VBGS-Depth and CVBGS-Depth.

While performance evaluation on a per-sensor basis gives a good sense of how well the algorithm is performing, it does not measure how well the system is achieving the end goal of detecting people in a scene. That is, a person may not be detected in one

Performance Per Sensor

Sensor	Method	F_1	Precision	Recall
	IMBGS	0.5269	0.5116	0.5432
	IMBGS-Depth	0.6278	0.6324	0.6232
1	*VBGS*	0.7053	0.5794	**0.9012**
	VBGS-Depth	0.7209	0.6019	0.8986
	CVBGS	0.7474	0.6514	0.8765
	CVBGS-Depth	**0.7531**	**0.6559**	0.8841
	IMBGS	0.1327	0.4828	0.0769
	IMBGS-Depth	0.3182	0.3500	0.2917
2	*VBGS*	0.6049	0.6901	0.5385
	VBGS-Depth	**0.7692**	0.7143	**0.8333**
	CVBGS	0.5436	**0.7429**	0.4286
	CVBGS-Depth	0.7037	0.6333	0.7917
	IMBGS	0.1117	0.4074	0.0647
	IMBGS-Depth	0.3636	**0.6667**	0.2500
3	*VBGS*	0.5175	0.6379	0.4353
	VBGS-Depth	**0.7369**	0.6364	**0.8750**
	CVBGS	0.4419	0.6477	0.3353
	CVBGS-Depth	0.6364	0.5000	**0.8750**
	IMBGS	0.5886	0.6200	0.5602
	IMBGS-Depth	0.5515	0.8065	0.4190
4	*VBGS*	0.8305	0.7819	**0.8855**
	VBGS-Depth	**0.8328**	0.7861	**0.8855**
	CVBGS	0.8121	0.8171	0.8072
	CVBGS-Depth	0.8146	**0.8221**	0.8072
	IMBGS	0.5188	0.7600	0.3938
	IMBGS-Depth	0.5905	0.6242	0.5602
5	*VBGS*	0.7366	0.6277	**0.8912**
	VBGS-Depth	0.7536	0.6543	0.8883
	CVBGS	0.7989	0.8968	0.7202
	CVBGS-Depth	**0.8088**	**0.9214**	0.7207

Table 4.1: Summary of results for bounding-box detections on each sensor. *IMBGS* denotes results using the image-based background, subtraction, *VBGS* denotes results using the voxel-based background subtraction with a voxel size of 10cm^3, and *CVBGS* denotes results using the voxel-based background subtraction with a color model and a voxel size of 20cm^3. The *-Depth* suffix denotes results where the groundtruth is depth limited.

XY Centroid Performance

Method	Performance	10cm	20cm	30cm	40cm	50cm	60cm	70cm	80cm	90cm	100cm
IMBGS	**Precision**	0.6667	0.7273	0.7403	0.7403	0.7425	0.7436	0.7458	0.7479	0.7479	0.7500
	Recall	0.5882	0.7843	0.8382	0.8382	0.8480	0.8529	0.8627	0.8725	0.8725	0.8824
	F_1	0.6250	0.7547	0.7862	0.7862	0.7918	0.7945	0.8000	0.8054	0.8054	0.8108
VBGS	**Precision**	0.6107	0.6421	0.6471	0.6507	0.6507	0.6507	0.6507	0.6507	0.6542	0.6600
	Recall	0.7843	0.8971	0.9167	0.9314	0.9314	0.9314	0.9314	0.9314	0.9461	0.9706
	F_1	0.6867	0.7485	0.7587	0.7661	0.7661	0.7661	0.7661	0.7661	0.7735	0.7857
CVBGS	**Precision**	0.8786	0.8945	0.8955	0.8986	0.8990	0.8990	0.8990	0.8990	0.8995	0.8995
	Recall	0.7451	0.8725	0.8824	0.9118	0.9167	0.9167	0.9167	0.9167	0.9216	0.9216
	F_1	0.8064	0.8834	0.8889	0.9052	0.9078	0.9078	0.9078	0.9078	0.9104	0.9104

Table 4.2: Summary of results for XY location detections. *IBGS* denotes results using the image-based background, subtraction, *VBGS* denotes results using the voxel-based background subtraction with a voxel size of $10cm^3$, and *CVBGS* denotes results using the voxel-based background subtraction with a color model and a voxel size of $20cm^3$.

sensor but is detected in another sensor and hence the person can still be tracked over time. To evaluate this, a centroid for the detected person is computed and the X,Y location of that centroid is compared to groundtruth X,Y locations. These centroids are then compared and if a computed centroid is within a particular distance threshold of a groundtruth centroid the detection is considered a success. This differs from the tracking performance metrics mentioned in Section 4.4, in that these detections are not tracked between frames and this just represents how well the detections are localized.

Table 4.2 shows results from evaluation on the *XY* location of detected people. The threshold for a correct detection is varied from 10cm to 100cm. A resolution of 40-60cm is a reasonable discretization since it approximates the space a person would occupy without touching another person. None of the methods perform particularly well at 10cm, however by 40cm both voxel-based methods are able to get a recall in the low 90's. The results also show that the higher resolution of the CVBGS is able to remove much more of the background, yielding a strong precision, while still maintaining enough foreground to achieve a strong recall. The precision of the VBGS method is lower, since

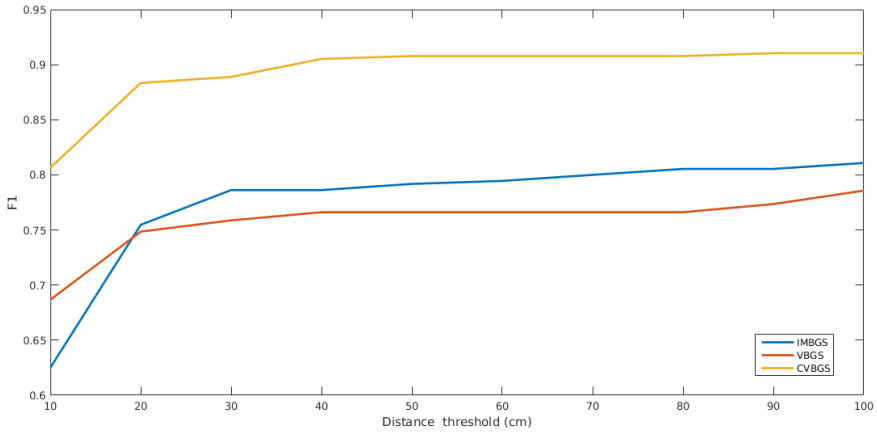


Figure 4.4: Plot of results for XY locations. *IBGS* denotes results using the image-based background, subtraction, *VBGS* denotes results using the voxel-based background subtraction with a voxel size of 10cm^3, and *CVBGS* denotes results using the voxel-based background subtraction with a color model and a voxel size of 20cm^3.

more background is present and creates false positives, but the recall is higher than any other method. The IMBGS method is able to achieve better precision than the VBGS method, however this is at the cost of recall. In addition to Table 4.2, the F_1 from this table has been plotted and is depicted in Figure 4.4.

From these results, it can be seen that both voxel-based background subtraction methods perform better than using image-based background subtraction for per-sensor metrics. Additionally, the performance sees a noticeable increase once the VCCS detection method is used, as is shown in Section 4.7. Unfortunately, this implementation of CVBGS did not consistently out perform VBGS. Some of this lies with noise in the color data, since voxels are combining points generated from different sensors there may be different variations in the color which muddy the color distribution. Applying some color normalization across sensors could help with this, as well as trying a different color model such as HSV. Some experiments were performed along this line, although improvements in performance were never seen.

Ultimately, adding in color information significantly impacts performance as the VBGS method just requires an index look-up in a binary array for each point whereas

with the color information more data needs to be stored for each voxel, and that information requires more computation to compare. With the VCCS detection method, the computation of object detections is more robust, and ultimately the VBGS method was chosen. Additionally, the tracking performance increases significantly with the VCCS detection method, such that it performs better in all cases.

4.6 Comparing Against RGB-based Person Detection

The previously described detection methods are purely bottom-up: they start with the individual points and combine information together based on lower-level information to form detections. They do not utilize any *a priori* information in a top-down fashion. Given the advancement of person detection methods, it is reasonable to think that using an *a priori* model of a person could aid in the performance of this system. A pitfall with this approach is that it can be difficult to learn a model that works well in the chaotic and unstructured environment of the classroom. Children behave differently from adults, and also have different physical features.

To provide top-down analyses to assist in 3D segmentation, work from modern person detectors on images has been adapted. Typically for an input image, such detectors return a collection of bounding boxes enclosing the minimal areas determined to display a person. In this approach, the person detection algorithm is applied to each image view. For each image (from each view), a corresponding detection image is created by concatenating each bounding region. This detection image is then used as a detection scheme to retain only points that belong to a person.

Two modern person detectors (PDs) were examined using the learned models provided by the respective authors. In [103], a cascade of LogitBoost classifiers were learned on covariances of image features mapped to a linear space. Faster than a similar method by [58], it learns weak classifiers on submatrices of the full covariance of features over a detection window. This cascade is then applied to a sliding detection window over the input image and the highest scoring detection boxes are kept as output.

The second method tested was a deformable parts-based model for object detection proposed in [29]. There, a class model is defined by a coarse *root filter* along with several, higher resolution *part filters*, along with a spatial model that weights the part

Performance Per Sensor

Sensor	Method	F_1	precision	recall
	HGB	0.2957	0.1799	**0.8300**
1	*pca-filter* ($\tau = 0.93$)	0.5097	**0.8718**	0.3601
	pca-filter ($\tau = 0.7$)	0.5092	0.7337	0.3899
	pdm1	0.6024	0.8038	0.4817
	pdm2	**0.7387**	0.6960	0.7870
	pdm2-pca ($\tau = 0.7$)	0.5734	0.8689	0.4279
	HGB	0.2345	0.1470	**0.5790**
2	*pca-filter* ($\tau = 0.93$)	0.5200	0.8726	0.3704
	pca-filter ($\tau = 0.7$)	0.5191	0.8424	0.3751
	pdm1	0.5105	**0.9545**	0.3484
	pdm2	**0.6058**	0.9260	0.4501
	pdm2-pca ($\tau = 0.7$)	0.5418	0.9396	0.3806
	HGB	0.2870	0.2084	**0.4607**
3	*pca-filter* ($\tau = 0.93$)	0.2931	0.8754	0.1760
	pca-filter ($\tau = 0.7$)	0.3272	0.8606	0.2020
	pdm1	0.3277	**0.9088**	0.1999
	pdm2	**0.5500**	0.8531	0.4058
	pdm2-pca ($\tau = 0.7$)	0.3392	0.8533	0.2117
	HGB	0.5145	0.4066	**0.7005**
4	*pca-filter* ($\tau = 0.93$)	0.2649	**0.9203**	0.1547
	pca-filter ($\tau = 0.7$)	0.3236	0.8930	0.1976
	pdm1	0.3839	0.8301	0.2497
	pdm2	**0.6344**	0.7710	0.5389
	pdm2-pca ($\tau = 0.7$)	0.3712	0.9017	0.2337
	HGB	0.4555	0.3277	**0.7469**
5	*pca-filter* ($\tau = 0.93$)	0.1775	**0.9237**	0.0982
	pca-filter ($\tau = 0.7$)	0.2426	0.9125	0.1399
	pdm1	0.2737	0.8822	0.1620
	pdm2	**0.4965**	0.6332	0.4083
	pdm2-pca ($\tau = 0.7$)	0.2540	0.8879	0.1482

Table 4.3: Summary of results for varying on the baseline method (referred to above as HGB). The simple PCA-based heuristic method is denoted *pca-filter*, where τ is the threshold parameter. The use of the person detection method described in [103] is referred to as *pdm1*, the second person detection method [29], as *pdm2*. The latter was also used in combination with a PCA-based heuristic and shown as *pdm2-pca*.

locations. An object class—in this case people—was trained on input images using a latent support vector machine (LSVM).

While person detectors are potentially a powerful means of ensuring the desired content is extracted from the point cloud data, determining these detections in the separate views can be time consuming. Additionally, the orientations presented to the

multiple sensors can differ widely from the provided pre-trained PD models. Given this, it is attractive to try a simple heuristic to incorporate the top-down information in lieu of more complex person detection algorithms. One simple model assumes a range for a person's tall to narrow aspect ratio. In some experiments, PCA was applied to the segmented object clouds and clouds having invalid ratios of their Z principal components below a threshold were eliminated.

The purely bottom-up detection strategy HGB was examined first and is referenced as HGB. The simple PCA-based heuristic method is denoted *pca-filter*, where τ is the threshold parameter. The HGB detection method, combined with the use of the person detection method described in [103], is referred to as *pdm1*, while the second person detection method [29], as *pdm2*. The latter was also used in combination with a PCA-based heuristic and shown as *pdm2-pca*.

Summaries for the different methods tried are shown in Table 4.3. From this, it can be seen that *baseline* provides moderate recall although the precision is very low. This is due to the number of foreground non-person objects that it segments. The subsequent methods perform much better in terms of precision since they remove many of these false positives. This is desirable as false detections are detrimental to latter portions of the processing pipeline.

Average accuracy numbers for state-of-the-art segmentation methods on challenging datasets like the PASCAL VOC 2011 Segmentation Challenge [100] (a dataset comparable in difficulty to the IRB-restricted classroom recordings) show equivalent performance. The highest average accuracy for PASCAL VOC was 0.433, while the highest for the person class was 0.464). Of variants explored, *pdm2* provided the best results across all sensors, exceeding those of the other person detectors in terms of accuracy.

Misleadingly, *baseline* appears to dominate among the recall values. This however is due to *baseline* being overly permissive in qualifying segmented objects, and in the recall measure not penalizing any false positives. The variation yielding the best precision was split between *pca filter* and *pdm1*. This likely comes from them both being very selective and retaining only the very best candidates, and false positives not being a part of the precision calculation.

Overall, the best method tested has been *pdm2*. It showed superior accuracy across the different sensor views as well as acceptable measures for precision and recall. However,

this approach generally provides too poor of a recall to be very effective. These results all get surpassed by replacing HGB with VCCS, however they represent an important developmental step of this system.

4.6.1 Direct Person Detection Comparison

While the previous section outlines an approach to combine person detection scores into the detection pipeline, it is also possible to directly compare the detection bounding boxes against the reprojected segmentation masks from the VCCS detection method. As opposed to the previous section, where person detection scores were used to aid in the detection step, it is also interesting to compare the detection methods presented here directly to RGB-based person detection algorithms.

To do this, the deformable parts-based person detection algorithm mentioned in the previous section is used [29]. In particular, the latest release of this code was used: release 5 [30]. This was trained using the PASCAL VOC 2010 person dataset [31]. While this is not the most accurate model to use, it does contain a lot of generic images of people, and creating a person model specific to this application is itself a unique challenge.

To perform this comparison, bounding-boxes from the RGB-based person detection algorithm are computed and compared to groundtruth bounding boxes. The detection algorithms presented in this paper are used to created segmentation masks on the image plane and from these bounding boxes are computed, as described in Section 4.3. These bounding boxes are compared to the groundtruth, and matches are only accepted if the bounding boxes overlap by 50% or more. Experiments were run with all three groundtruthed datasets, and the performance from each sensor is averaged together for each dataset.

Table 4.4 presents the results from these experiments. It is clear from these results that the RGB-based methods perform poorly for this application. In particular, the recall rates are significantly lower, which is quite problematic. It should be noted that there are more modern and powerful person detection algorithms, however this method was state-of-the-art several years ago. Additionally, as was mentioned, the trained models are also not completely accurate. Still, this experiment shows that using standard person-detection algorithms are not guaranteed to perform well in this difficult

(a) Recording 1

Performance Per Sensor

Method	F_1	precision	recall
PD	32.8%	46.7%	25.2%
HGB	65.0%	52.6%	86.7%
VCCS	85.5%	82.8%	89.7%

(b) Recording 1

Performance Per Sensor

Method	F_1	precision	recall
PD	35.3%	47.8%	28.0%
HGB	66.6%	56.6%	78.7%
VCCS	83.5%	80.9%	86.6%

(c) Recording 3

Performance Per Sensor

Method	F_1	precision	recall
PD	20.7%	35.4%	14.6%
HGB	66.7%	54.9%	86.7%
VCCS	79.2%	74.0%	86.7%

Table 4.4: Summary of performance comparisons using RGB-based person detection. Performance is computed for all 5 sensors and then averaged together for one value per recording.

application. There are many aspects that can confound purely RGB-based methods, including lighting, and a rapidly changing dynamic environment.

Another aspect that can make a direct applicable of RGB-based person detection algorithms is run-time. The average per-frame processing time for the parts-based deformable models person detection is 5.2014s. This is just the time to perform person detection on a single image, and this would have to be performed on 5 images per 'frame'. Even on a multi-core machine, this would mean that the per-frame processing time at best would be around 5s, which means the system would take far too long to process.

4.7 Detection and Tracking Validation

The previous sections showed experiments that were done in the past that helped shape the development of this system. This section will now present validation results from the latest iteration of the system, and present the most comprehensive evaluation of the system. Results will be presented on the aforementioned three datasets.

(a) Recording 1

Performance Per Sensor

Sensor	F_1	Precision	Recall
1	83.1%	73.9%	95.1%
2	91.2%	83.8%	100%
3	86.9%	86.7%	87.1%
4	82.5%	93.0%	74.1%
5	83.7%	76.6%	92.1%

Tracking Performance

MOTP	MOTA	Miss Rate	FP rate
5.63cm	90.0%	1.51%	1.81%

(b) Recording 2

Performance Per Sensor

Sensor	F_1	Precision	Recall
1	83.7%	77.4%	91.1%
2	89.1%	84.0%	94.7%
3	68.8%	73.3%	64.7%
4	85.9%	84.9%	86.9%
5	89.9%	84.8%	95.7%

Tracking Performance

MOTP	MOTA	Miss Rate	FP rate
4.60cm	96.5%	0.5%	0.99%

(c) Recording 3

Performance Per Sensor

Sensor	F_1	Precision	Recall
1	68.5%	55.4%	89.9%
2	69.2%	66.7%	72.0%
3	94.0%	91.9 %	96.1%
4	66.7%	58.3%	77.8%
5	97.8%	97.8%	97.8%

Tracking Performance

MOTP	MOTA	Miss Rate	FP rate
4.44cm	98.0%	0%	1.0%

Table 4.5: Summary of performance for current methods.

The detection portion of the pipeline can be tested by reprojecting bounding boxes on the image plane and comparing them to groundtruth bounding boxes. Evaluation is performed as in the PASCAL Visual Object Classes (VOC) Challenge [31], where the area

of overlap is computed and an overlap of 50% is required for a true positive (TP). False positives (FP) occur when computed values have no matching groundtruth values and false negatives (FN) when groundtruth values are not matched with computed values. These can then be used to compute the standard values of precision, recall, and F_1 (the harmonic mean of precision and recall).

(a) Euclidean

Performance Per Sensor

Sensor	F_1	Precision	Recall
1	69.8%	56.6%	91.6%
2	81.3%	74.4%	90.0%
3	77.5%	71.9%	84.3%
4	81.0%	83.4%	79.6%
5	65.2%	50.6%	95.6%

Tracking Performance

MOTP	MOTA	Miss Rate	FP rate
7.55cm	41.5%	1.4%	50.2%

(b) HGB

Performance Per Sensor

Sensor	F_1	Precision	Recall
1	62.3%	47.9%	90.1%
2	67.9%	55.3%	88.9%
3	70.3%	64.4%	78.7%
4	58.5%	49.4%	73.5%
5	71.4%	59.9%	89.0%

Tracking Performance

MOTP	MOTA	Miss Rate	FP rate
8.54cm	13.1%	7.9%	54.5%

(c) VCCS

Performance Per Sensor

Sensor	F_1	Precision	Recall
1	78.4%	68.9%	92.0%
2	83.2%	78.2%	88.9%
3	90.5%	86.4%	95.2%
4	83.2%	84.0%	82.6%
5	78.4%	78.7%	79.6%

Tracking Performance

MOTP	MOTA	Miss Rate	FP rate
4.89cm	94.8%	0.7%	1.3%

Table 4.6: Summary of performance across different detection methods.

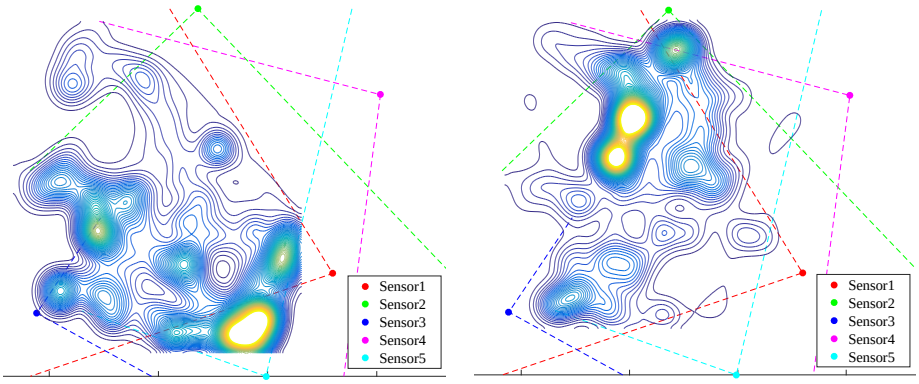
To evaluate tracking performance, the common CLEAR MOT metrics MOTA and MOTP are used [101], as described in Section 4.4. MOTP, multiple object tracking precision, measures the total error in estimated position averaged by the number of matches, and indicates the ability of the tracker to estimate precise objection positions.

MOTA, multiple object tracking accuracy, accounts for all object configuration errors made by the tracker over all frames, and gives an intuitive measure of the tracker's performance. Additionally, miss rate and false positive rate are computed.

The results are summarized in Table 4.5 and Figures 4.5, and 4.6. One thing to note in the performance table is that, despite the per sensor recall rates, the miss rate for the tracking in each recording is quite low. This illustrates the advantage of multiple sensors, as a detection may be missed in several sensors but it's likely at least one sensor will have a positive detection.

In additional to performance on the previously described system, results from previous iterations of this system are presented to provide a point of comparison. For these comparisons, the methods for generating detections were adjusted. In the first method, referred to as Euclidean, detections are formed from the background subtracted global point cloud by performing clustering on Euclidean distance between points: any points that are within an epsilon Euclidean distance are considered part of the same object. Additionally, to test the idea of hierarchically computing supervoxels and then clustering supervoxels into detections, results are presented using the hierarchical graph-based (HGB) approach described in previous work [70] as well as in Chapter 3 Section 3.6. The results of these comparisons are presented in Table 4.6. Performance results were computed across the same three recordings presented in Table 4.5, however the results were averaged together to provide a single result for each method.

Figure 4.5 shows contour plots that depict the placement of the sensors with respect to the activity of the classroom where red denotes areas of high activity and blue represents areas of low activity. The areas of high activity in recordings 1 and 3 end up occurring near the table seen in Figures 3(a)-(d); the table ends up being a focus of activity for many of the classroom occupants. In Figure 4.5(c), the two contour regions branching out from the table area are entrances/exits to the classroom area. The branch off to the left side of the diagram leads to another area of the classroom while the branch that heads to sensor 2 ends in a door to the playground. The first recording contains a number of children playing further back in the classroom who then venture into the area of recording; this recording contains nine distinct children and three adults. The second recording contains nine distinct children and four adults. The majority of the third sequence consists of several children entering the room and heading over to the table to



(a) Recording 1

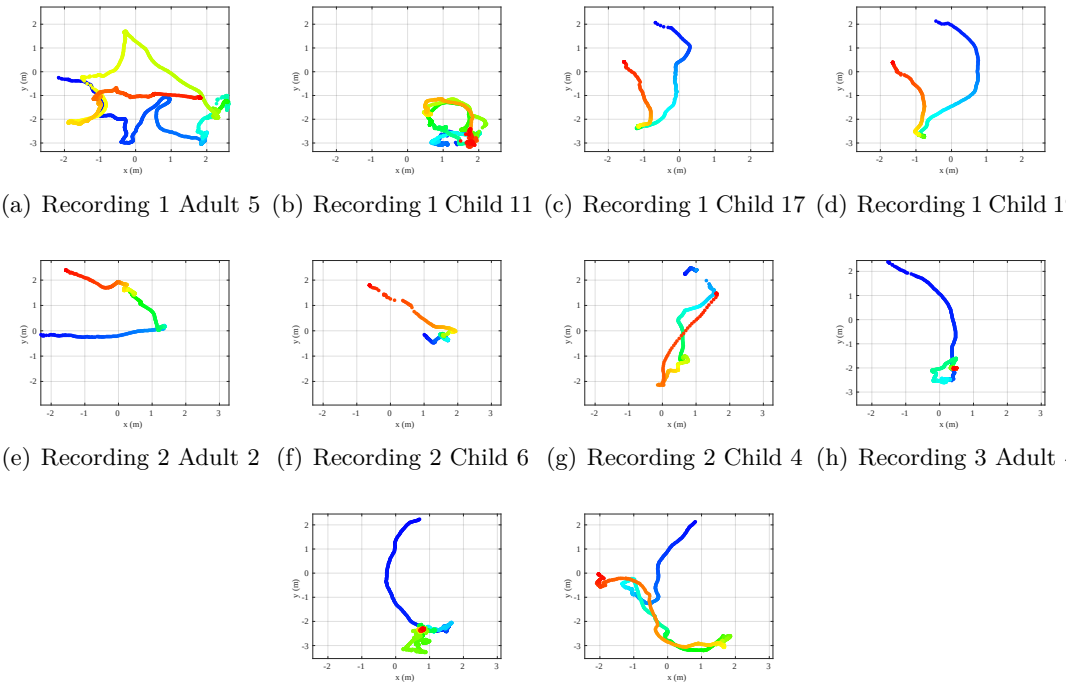
(b) Recording 2

(c) Recording 3

Figure 4.5: Sensor placement and activity levels. The contours denote regions of activity with blue corresponding to the lowest activity and red the highest. The large dots represent sensor locations and the dashed lines show their fields of view.

play, eventually being joined by an adult who sits at the table. The third recording has four distinct children and two adults (although one child and one adult are only visible for a few seconds as they pass through the room).

Example trajectories of ten occupants are shown in Figure 4.6, with the trajectories being color coded such that blue represents where the track begins and red where the track ends. The trajectories belong to individuals who are deemed to have a high relationship in Figure 5.2. Figure 4.6(i) depicts the trajectory of a child who enters from the playground and then sits at the table to play, while Figure 4.6(h) is the trajectory of an adult who enters the room and sits at the table to play with Child 1. Figure 4.6(b)



(a) Recording 1 Adult 5 (b) Recording 1 Child 11 (c) Recording 1 Child 17 (d) Recording 1 Child 19

(e) Recording 2 Adult 2 (f) Recording 2 Child 6 (g) Recording 2 Child 4 (h) Recording 3 Adult 4

(i) Recording 3 Child 1 (j) Recording 3 Child 3

Figure 4.6: Top-down view of tracking results on several individuals. Tracks begin as blue dots and transition to red by the end.

belongs to a child who enters with an orange cloak from a dress-up area, and he plays in the area around the table. Adult 5 from recording 1, depicted in Figure 4.6(a), enters the room and interacts with Child 11, eventually removing the cloak. The tracking system is able to maintain the identity of Child 11 through this de-cloaking. Figure 4.6(e) and (f) also depict another interesting interaction, where Adult 2 enters from further into the classroom and walks up to Child 6 then kneels down and hugs the child. The identify of both occupants is successfully maintained throughout this interaction.

The quantitative evaluation of system performance is summarized in Table 4.5. Per-sensor performance tends to vary between sensors, which is generally related to the sensors position with respect to the activity in the room. Frequently, activity may be occurring either at the edge of a sensor, or far away from it. In general, the recall rates for each sensor are good, which is important as the system should prioritize not missing observations in order to provide accurate behavioral trends. Additionally, the

benefit of combining information from multiple sensors is shown in the tracking performance, where all metrics are quite good. The MOTA is brought down a bit because of misidentification errors, which generally occur when an occupant leaves the room and returns.

Comparing the different detection method performances in Table 4.6, it is clear that the current method performs the best. The two previous detection methods frequently suffer from a large number of false positives, which bring the MOTA down. In additional to better performance, the average frame processing time is significantly lower for the current method. The Euclidean method takes an average of 2.3 seconds across all processed frames, while the HGB method takes 0.84 seconds. The current method, VCCS, is the fastest at 0.54s per frame.

This timing represents the total time spent processing each from, including reading in data from disk, generating the per-sensor point cloud, combining into the global cloud, detecting objects, tracking, and writing out any results. It should be noted that approximately 0.1s are spent just on input/output, which is a part of the processing that is unaffected by algorithm speed. Additionally, while certain aspects of the processing pipeline are performed in a multi-threaded fashion, the vast majority of the processing isn't designed to take advantage of parallelism, which represents a significant avenue of speed increase.

4.8 Evaluating Appearance Descriptors

The previous sections have evaluated all of the early portions of the system processing pipeline. Now, some analysis on the performance of the appearance descriptors is given. For this evaluation, the region covariance descriptors, RGB-based color histograms, and HSV-based color histograms discussed in Chapter 3 Section 3.7 will be used. The body of possible descriptors is enormous; this is not meant to be an exhaustive analysis, but is meant to indicate that these initial steps are valid. The appearance descriptors perform important roles in this system – this analysis provides a baseline so that further work can develop and select descriptors that improve performance.

There are two ways in which validation is performed using these descriptors. The first utilizes the groundtruth described in this chapter. For each frame, as part of

the tracking procedure, there is a track-to-observation matching that must be made, and the appearance descriptors play a large role in this. As part of that, the tracking performance of different descriptors can be compared to see if one performs better than the other. Computational time can also be compared, as a quicker-to-compute but less accurate descriptor may actually be appealing if the time savings is significant enough.

It is also desirable to be able to validate the performance of the system in the absence of groundtruth. Ideally, this system will run on a large amount of data that would be completely infeasible to completely groundtruth. Still, users would desire confidence that the system is performing well. One approach to this would be to give a visual representation of the separability of the descriptors. By projecting the descriptors down into a smaller space, either 2 or 3 dimensions, the degree of separability between the descriptors can be visually approximated. Furthermore, by performing these embeddings upon descriptors within a single track, there may be groups of appearances within a single track. These groups could yield meaningful information about the track itself, but also could indicate if there is an identity switch.

4.8.1 Validation via Tracking

One way to validate the performance of the appearance descriptor is to see how altering the appearance descriptor affects tracking performance. If there is an ambiguity between which observations to match to a track, then the appearance descriptor ideally should help make the correct match. If a correct match is made, this will be reflected by a smaller number of mismatch errors, which is then reflected in a higher MOTA.

Experiments were performed by running the same detection and tracking methods but varying the computed appearance descriptor. Three different appearance descriptors were used: region covariance descriptors, RGB color histograms, and HSV color histograms. These descriptors are described in Chapter 3 Section 3.7. For covariance descriptors, the log euclidean distance measure was used and for color histograms both histogram intersection and chi-squared kernels were used. Additionally, experiments were performed using no appearance descriptor to test both how much the appearance descriptor was contributing, as well as to measure how much additional time was spent because the appearance descriptor was used.

Tables 4.7 and 4.8 summarize the results of these experiments, where the first table

(a) Recording 1

Descriptor Performance

Descriptor	Distance	Split-method	MOTP (cm)	MOTA	Miss Rate	FP Rate
cov	logeuc	no-split	5.63	89.43%	1.51%	1.81%
histHSV	histInt	no-split	5.54	80.06%	1.21%	12.99%
none	none	no-split	5.97	78.85%	1.51%	11.48%
cov	logeuc	supix	5.42	78.55%	5.14%	4.53%
hist	chiSqr	no-split	5.96	75.83%	1.21%	15.71%
hist	histInt	no-split	5.95	75.83%	1.21%	15.71%
histHSV	chiSqr	no-split	5.94	75.53%	1.21%	15.71%
hist	chiSqr	supix	6.27	66.77%	0.30%	21.45%
histHSV	chiSqr	supix	6.30	66.16%	1.21%	22.36%
hist	histInt	supix	6.52	63.44%	0.30%	23.57%
histHSV	histInt	supix	6.31	63.14%	1.21%	24.47%

(b) Recording 2

Descriptor Performance

Descriptor	Distance	Split-method	MOTP (cm)	MOTA	Miss Rate	FP Rate
histHSV	chiSqr	no-split	11.24	92.47%	0.65%	4.52%
histHSV	histInt	no-split	11.24	92.47%	0.65%	4.52%
cov	logeuc	no-split	11.24	91.61%	0.65%	4.95%
hist	chiSqr	no-split	11.29	91.61%	0.65%	4.73%
hist	histInt	no-split	11.30	91.61%	0.65%	4.73%
histHSV	histInt	supix	11.32	91.61%	0.65%	4.73%
hist	chiSqr	supix	11.28	90.75%	0.65%	5.16%
hist	histInt	supix	11.28	90.75%	0.65%	5.16%
none	none	no-split	11.30	90.75%	0.65%	5.16%
histHSV	chiSqr	supix	11.27	90.54%	0.65%	5.16%
cov	logeuc	supix	11.21	89.25%	2.58%	4.95%

(c) Recording 3

Descriptor Performance

Descriptor	Distance	Split-method	MOTP (cm)	MOTA	Miss Rate	FP Rate
cov	logeuc	no-split	4.43	98.02%	0.00%	0.99%
hist	chiSqr	no-split	4.44	98.02%	0.00%	0.99%
hist	histInt	no-split	4.43	98.02%	0.00%	0.99%
histHSV	chiSqr	no-split	4.44	98.02%	0.00%	0.99%
histHSV	histInt	no-split	4.44	98.02%	0.00%	0.99%
none	none	no-split	4.43	97.03%	0.50%	0.99%
cov	logeuc	supix	5.51	85.64%	9.90%	0.99%
histHSV	chiSqr	supix	6.01	63.86%	0.00%	25.25%
histHSV	histInt	supix	6.01	63.86%	0.00%	25.25%
hist	chiSqr	supix	7.63	39.60%	0.00%	43.56%
hist	histInt	supix	7.59	38.61%	0.00%	43.56%

Table 4.7: Summary of performance for different appearance descriptors. Sorted by MOTA.

(a) Recording 1

Descriptor Timing

Descriptor	Distance	Split-method	Time (s)
hist	histInt	no-split	0.5192
none	none	no-split	0.5356
hist	chiSqr	no-split	0.5368
histHSV	histInt	no-split	0.5430
histHSV	chiSqr	no-split	0.5457
cov	logeuc	no-split	0.5842
cov	logeuc	supix	0.6559
hist	histInt	supix	0.7068
hist	chiSqr	supix	0.7190
histHSV	chiSqr	supix	0.7778
histHSV	histInt	supix	0.7781

(b) Recording 2

Descriptor Timing

Descriptor	Distance	Split-method	Time (s)
hist	histInt	no-split	0.4049
hist	chiSqr	no-split	0.4163
histHSV	histInt	no-split	0.4217
histHSV	chiSqr	no-split	0.4272
none	none	no-split	0.4280
hist	chiSqr	supix	0.4695
hist	histInt	supix	0.4754
histHSV	histInt	supix	0.5002
histHSV	chiSqr	supix	0.5129
cov	logeuc	no-split	0.5221
cov	logeuc	supix	0.5536

(c) Recording 3

Descriptor Timing

Descriptor	Distance	Split-method	Time (s)
none	none	no-split	0.4212
histHSV	chiSqr	no-split	0.4264
hist	histInt	no-split	0.4277
histHSV	histInt	no-split	0.4307
hist	chiSqr	no-split	0.4323
cov	logeuc	no-split	0.5192
hist	chiSqr	supix	0.5230
hist	histInt	supix	0.5234
histHSV	histInt	supix	0.5311
histHSV	chiSqr	supix	0.5320
cov	logeuc	supix	1.1575

Table 4.8: Summary of timing for different appearance descriptors. Times are presented as the average number of seconds it takes to process each frame. Sorted by shortest time.

presents the MOTP, MOTA, miss rate and FP rate for different combinations and the second table presents the runtimes. The results in these tables are ordered by performance (MOTA for tracking). Runtime is presented as the average time it takes to complete process each frame.

The most obvious result from these tables is that the splitting the object clouds into supervoxels for descriptor computation is very detrimental to performance. It is consistently towards the bottom, and is always worse than not splitting at all for the

respective descriptor. This is counter to expectations. One would expect that using the whole object cloud would have some confounding information, as several different pieces of information would be combined together (*i.e.*, blue colors from jeans would be combined with the shirt color, combined with skin color, combined with hair color). The expectation would be that, by splitting the object cloud into sub-components, it would lead to more accurate descriptors that describe specific components and that the supervoxels would be a natural choice for this splitting. Additionally, the computation times for performing supervoxel descriptor comparisons are significantly higher.

Another important aspect of these results is that not using any descriptors tends to perform relatively well, sometimes even performing better than histogram descriptors. One explanation for this is that using the shape information can help disambiguate between two potential matches for a given track. Additionally, there may be fewer situations where there is a true ambiguity between two potential observations.

It is also intriguing to look at the timing for no descriptor, as it is not always the fastest as one might expect. While no descriptor is competitive for the fastest method, it is sometimes beaten out by the fast-to-compute histogram descriptors. The cause for this lies in observation splitting – each time an observation must be split additional time is spend on that particular frame, splitting the observation and re-running descriptor computations. If the usage of descriptors leads to more accurate observation/track matching, then the number of splits required could be reduced. For instance, one erroneous split (a single person giving two detections) could potentially be propagated for a number of frames.

While reasonably good performance can be achieved without computing appearance descriptors, it should be noted that appearance descriptors are useful for more than just tracking. This will be addressed more in the following sections, where the discriminative power of the different descriptors is explored and results from person reidentification is discussed. Furthermore, there are likely better descriptor choices, that could perform even better.

These results also show that the region covariance descriptors are a suitable choice. They generally perform the best, although they are not always the quickest descriptors to compute. The usage of supervoxels ends up costing significant computation time and does little to improve performance.

4.8.2 Validation via Graph Embedding

While the previous section presented results that utilized the groundtruth available from detection and tracking validation, it is also important to have ways to instill confidence in the tracking results and appearance descriptors that does not require groundtruth. It will be infeasible to groundtruth all of the data that the system processes, so at some point mechanisms need to be created that allow the system to be verified without groundtruth.

One way to perform this verification is to explore the quality of the tracks. One aspect to avoid is mismatch errors or identity swaps: if the tracks of two individuals cross a common problem problem with tracking systems is to swap the track identifies. The appearances of these two mismatched identifies are going to differ. Hence, if such an identity switch occurs, one would expect this to create different clusters with the appearance descriptors.

The difficulty here is that the appearance descriptors lie in a high dimensional space and hence are difficult to visualize. There exist, however, numerous way to embed higher dimensional spaces into lower ones. One common approach to this is principal components analysis (PCA) [104]. PCA computes a set of orthogonal basis vectors and then uses the basis vectors with the higher variance to project into a lower dimension.

Another recently popular method is t-SNE, which attempts to capture both local and global structure of high dimensional data [105]. This means that local neighborhood relationships between data points are preserved while attempting to keep the overall global structure. This is performed by modeling the probabilities that individual data points are neighbors to each other. Unfortunately, the machinery of this method requires that the neighborhood relationships are measured by Euclidean distance.

Another method is ISOMAP [106]. ISOMAP is a nonlinear dimensionality reduction technique that attempts to preserve the structure of the manifold in which the high-dimensional data lies on. It achieves this by trying to approximate the geodesic distance by using distances between neighboring points. Since the geodesic distance is locally-linear, the distance of neighboring points can be used directly, but the geodesic distance from faraway points can be approximated by looking at hop-distance between neighboring points.

Since ISOMAP attempts to approximate the geodesic distance, it is a suitable approach for embedding region covariance descriptors, and is the method used here. What this allows is to take all of the appearance descriptors for a particular track and then embed them into a 2 dimensional space. If images can be sampled for some of these points it can give several several interesting pieces of information for a track. Several examples of these images can be seen in Figure 4.7.

Figure 4.7(b) is an example where this approach is informative. Looking closely, there appear to be 3 distinct identities to this track: one is an adult woman with a blue

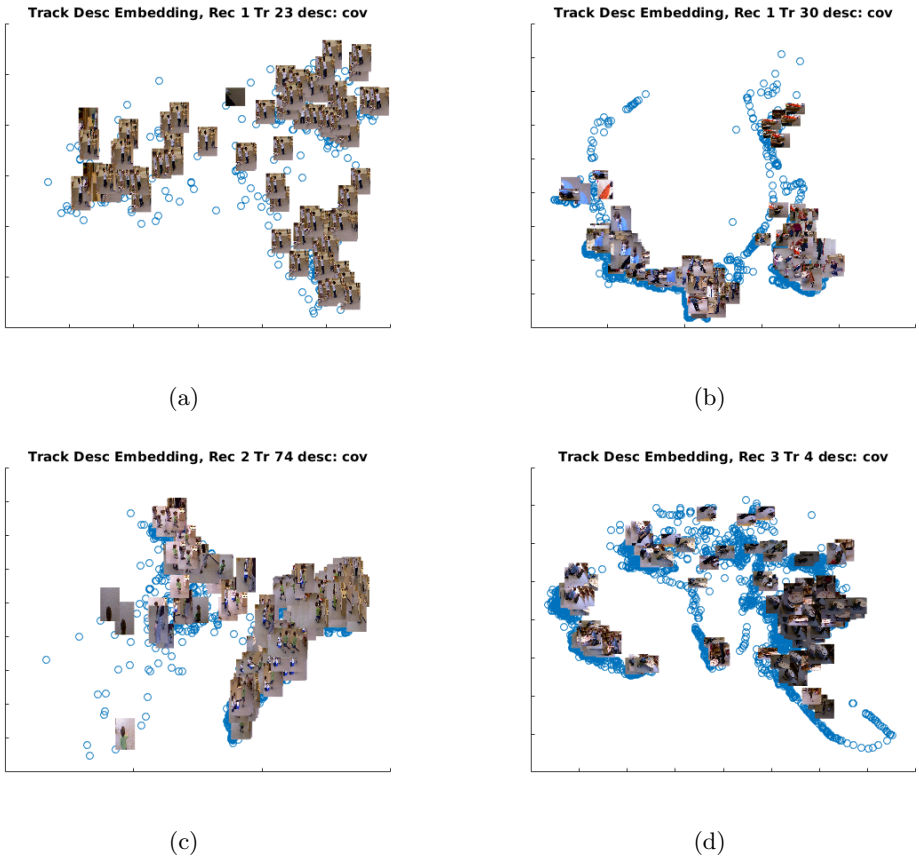


Figure 4.7: 2D embeddings of all descriptors for several tracks across recordings. Axes labels are omitted, as the scale is arbitrary. Note: the images are created by padding the detection bounding box. Pixels for appearance descriptors are computed from detected points, hence not all pixels shown are used.

sweater, another is a young girl with a dark blue dress, and another is a few frames from a basket of clothes. The other embeddings do show clusters of descriptors, however they don't correspond to different activities and poses as much as one might hope.

4.9 Evaluating Person Reidentification

4.9.1 Reidentification Groundtruth

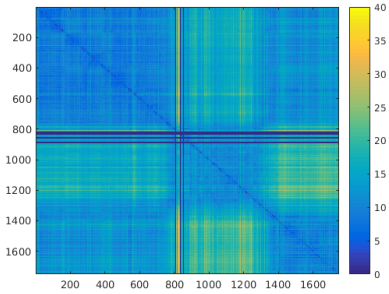
Several different applications had to be created in order to facilitate the generation of groundtruth for person reidentification. In this situation, it is necessary to know how the computed descriptors should be grouped, which is information that was not contained in the previous generated groundtruth.

The first step in this process is generating a way to summarize the identify of a computed track. This step is non-trivial, as the track could consist of thousands of observations. Some way to summarize this information into a way that is instantly recognizable to a human is required.

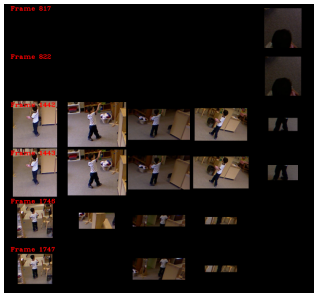
This problem can be aided by use of the appearance descriptors. As the appearance varies across observations, the computed descriptors will also vary. By examining groups of the most similar descriptors and least similar descriptors, an estimate of which track observations can be achieved. In order to examine these relationships, self similarity matrices are computed which are essentially an all-to-all affinity matrix. For each appearance descriptor in the track, that appearance descriptor is compared to all other appearance descriptors. From this a self-similarity matrix is constructed.

Using this self similarity matrix, the frames which have the largest or smallest differences can be found. From these, the top 5 most similar and top 5 least similar descriptors can be used to pick frames to view observations from. Images from each sensor at these frames can be combined to created composite 'identity' images for a track. Several examples of these identity images paired with their tracks self similarity matrices are shown in Figure 4.8. Note that, the top 5 highest/lowest descriptor comparisons are picked, each of which have two frames associated with them, and duplicate frames are not drawn, so the identity images do not always contain the same number of example frames.

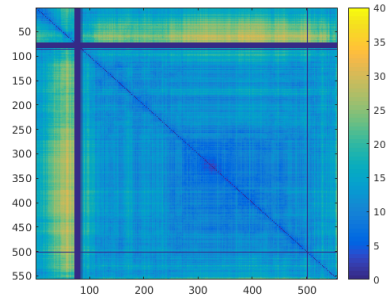
These track identity images create a good summary for identifying an individual



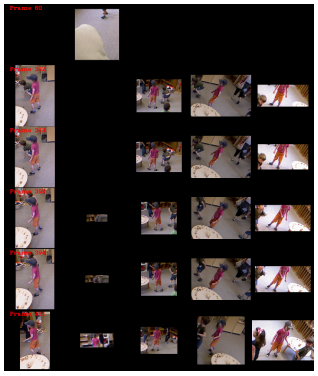
(a) Recording 1, Track 1 SSM



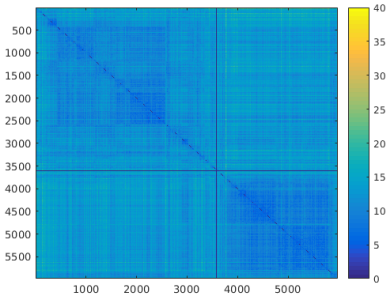
(b) Identity Image



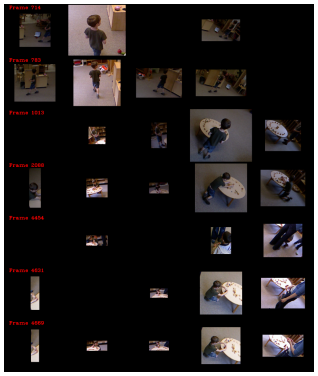
(c) Recording 2, Track 5 SSM



(d) Identity Image



(e) Recording 3, Track 4 SSM



(f) Identity Image

Figure 4.8: Track self-similarity matrices and identity images. Identity images depicting frames from each available sensor for the most dissimilar pairs of track descriptors. Frames/sensor combinations where there was no detection are left blank. Identity images shown from the most dissimilar descriptors, since it yields more variety.

in a track. By using these programs, a second application can be created to be able to manually reidentify tracks within a single recording. This program consists of two panes: one that displays the track identity image for the current track to match, and then a second pane where the user can flip through tracks to match to. The program allows the user to use the left/right arrow keys to cycle through currently unmatched tracks within the recording, and the M key allows the currently selected tracks to be matched together. The Enter key can then finish the matching for the currently selected track. The A/D keys then allow the user to change the currently selected track, if they wish to begin matching a new track.

This program allows for manual matching within a single recording, however once each recording has been manually matched, another program is used to match all tracks across each recording. This program works similarly to the previous one, having two panes: one for the currently selected track to match to, and one for candidate matches. However, in this program, the Up/Down arrows can be used to move between recordings for the candidate match. Using these programs, it is then possible to manually match all of the computed tracks together.

4.9.2 Reidentification Results

When it comes to evaluating reidentification algorithms there are two popular metrics. The first is referred to as Rank-1 accuracy and is essentially the same as classical classification accuracy. In this case, there is a groundtruth reidentification and Rank-1 accuracy measures how many times the computed match is the same as the groundtruth match. Since reidentification is a difficult and challenging problem, high Rank-1 accuracy is notoriously difficult to achieve.

It can also be beneficial to look at how the performance changes as the performance restrictions are relaxed. This leads to the idea of the cumulative match characteristic (CMC). If the algorithm is instead allowed to return a ranked list of possible matches, from which an operator can manually choose the correct match, it is possible for the algorithm to still greatly aid in the reidentification process. The question then becomes how high do true matches typically appear in the ranked list. This is what the CMC measures, by computing whether a true match exists in the top 1 ranked list, top 2 ranked list, top 3 ranked list, and so on. This is also similar to a precision-recall curve

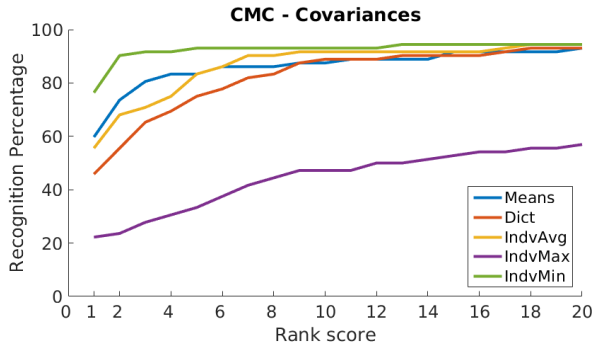
which is commonly used in classification problems. Likewise, other metrics can also be derived from the CMC such as area under the curve (AUC) and also expected rank (on average, how far down the list is the true match).

For this work, CMCs have been computed for all of the previously mentioned appearance descriptor combinations. The CMCs are depicted in Figure 4.9, where the plots are split into (a) region covariance related descriptors, (b) RGB histogram based descriptors, and (c) HSV histogram based descriptors.

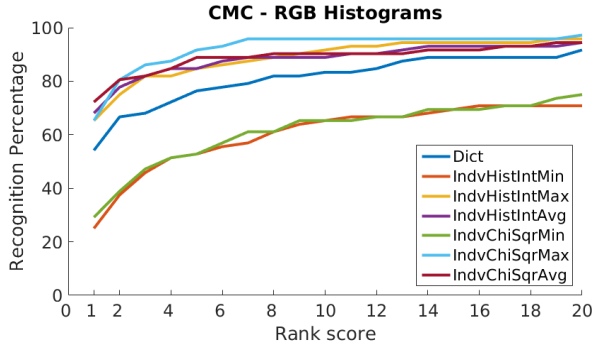
One issue that arises when trying to determine whether two tracks should be matched is determining how to compute distance between the two tracks. Since each track consists of a number of observations and each observation has an appearance descriptor associated to it, there are numerous ways to go about computing a similarity measure between the two. One approach is to perform an all-to-all comparison between each descriptor in both tracks, which results in an affinity matrix, but there is still a question of what to do with the resulting affinity matrix. Several approaches that come from cluster analysis can be used: either min, max or average distance [107]. Here, either the minimum, maximum, or average distances from the all-to-all comparison are used.

The problem with performing an all-to-all comparison is that it is computationally expensive, especially as the number of descriptors is increased. Instead, it can be useful to take all of the descriptors from one track and summarize them into a single descriptor or combination of descriptors. One approach to this is dictionary learning (see Chapter 3 Section 3.9). Another simple approach is to just create a mean descriptor from all of the descriptors for a track. For region covariance descriptors, this can be achieved with the Karcher mean [108].

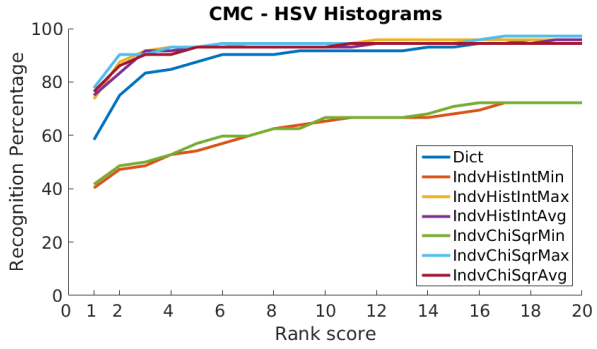
Some interesting trends appear when examining the CMCs. First, many of the descriptors end up performing well, although none ever achieve perfect accuracy. While not generally performing the best, dictionaries generally perform well. Despite not achieving the best performance, the gain in computation time could make them worthwhile. Additionally, there are potentially parameter changes that could affect performance, as well as more strict ways to learn the dictionaries, such as the methods in [98].



(a)



(b)



(c)

Figure 4.9: Cumulative Matching Characteristics for differing descriptors and comparison methods. Ranks vary from 1 to 20.

Chapter 5

Behavior Analysis

The overall motivating factors for this work are two-fold. First, the creation of a system that can aid mental health providers in screening for early signs of mental illnesses. Second, the creation of a platform for gathering data to aid in the discovery of new risk-markers for the development of mental illness. A fundamental issue to consider is that the broad scope of this work raises the challenge of identifying which behaviors to measure that would be relevant to many different types of mental illness and not just one disease area. Another fundamental challenge is that the design of the system can limit the kind of data that can be generated; for instance, there is a need to balance the desire to capture micro motor movements at high resolution versus the desire to capture system-wide information at lower resolution. In the present work, the design favors a system that could optimally capture larger movements and interactions within the broader system, as a starting point for generating some basic behavioral information that would be useful to mental health professionals.

Therefore, the current implementation of the behavioral analysis portion of this system had two goals. First, the measurement of several fundamental aspects of child behavior, as described in the following paragraphs. Second, the creation of an interactive system to provide an avenue of exploration for the generated data. By presenting the user with an interactive visualization, the data can be analyzed and adjusted to reveal conclusions that may not have been obvious otherwise.

The selection of which fundamental aspects of child behavior to target for this pilot

behavioral analysis system was conducted in collaboration with mental health professionals. A key principle in the selection was to identify behavioral measures that are basic in nature, that can be reliably measured at low resolution, and that may be relevant to a broad range of mental health functioning. Through an iterative process that included having the interdisciplinary members of the team view the video data together, the following measures were selected: position, velocity, social relationship (defined by distance between individuals), group behavior, and activity levels.

From the previous stages of the processing pipeline, the position of individual room occupants is known; from that information, velocity can be computed. Using this position and velocity information, several measure are computed: group behavior pertaining to social interaction, and behavior related to individual activity levels. The social information can be of great interest, as this can be difficult to ascertain without close observation. While the classroom teachers often have an overall feeling of which children are friends or not, they also need to manage a large number of children so detailed knowledge of child social interaction is not feasible to measure. By computing a social graph of the room occupants, it can be possible to observe and measure the kinds of relationships the room occupants have, which is an important aspect, as described in Chapter 2, and it can be possible to explore different aspects of these relationships.

In addition, the activity level of individual children can be important, both on a macro and micro level. On a macro scale, a hyperactive child may spend a lot of time running around, and moving quickly between activities. On a micro level, a child may be jittery, or have small motor ticks, or stereotypies.

Once data has been collected and these two measures computed over a number of recordings, managing and presenting all of that data can be difficult. To facilitate this, an interactive tool was developed. The social graph across all processed recordings is the core of this tool; the social graph is displayed, which can be manipulated, and then the various pieces of information in this graph can be drilled down into to provide more information.

The next two sections describe how the social graph is computed and also how activity level is measured. This is followed by results from the three groundtruthed recordings. After that, the interactive visualization tool is described along with discussion from data across 10 different recordings.

5.1 Measuring Social Relationship

The first step in identifying social groups is to define a measure of association that can be computed between tracked subjects. By accumulating this social association over time, a social graph can then be computed. The approach employed here is similar to [109] in that a connectivity score is assigned between two people at a given time through a function, f_{group}, which is a measure that incorporates relative position between two subjects i and j as well as the headings of each subject:

$$f_{\text{group}}(i,j) = \exp\left(-\frac{1}{2}\mu_d^T (C_i^T C_j)^{-\frac{1}{2}} \mu_d\right), \tag{5.1}$$

where μ_d is the difference between the centroid locations of the Kalman tracker for person i and person j and C_i is defined in Equation (5.2).

To incorporate heading information, a 2D oriented Gaussian kernel is used. The covariance of the Gaussian kernel is defined as

$$C_i = \begin{bmatrix} \frac{cos\theta_i^2}{\sigma_x^2} + \frac{sin\theta_i^2}{\sigma_y^2} & \frac{-sin2\theta_i}{2\sigma_x^2} + \frac{sin2\theta_i}{2\sigma_y^2} \\ \frac{-sin2\theta_i}{2\sigma_x^2} + \frac{sin2\theta_i}{2\sigma_y^2} & \frac{sin\theta_i^2}{\sigma_x^2} + \frac{cos\theta_i^2}{\sigma_y^2} \end{bmatrix}. \tag{5.2}$$

The parameters σ_x and σ_y were set such that they incorporate the assumption that people in a group tend to walk side by side. A corresponding kernel w.r.t. θ_j was also computed, with a covariance C_j as defined by Equation (5.2). These covariances were blended together to form the kernel shown in Equation (5.1).

The instantaneous similarity score between any two individuals in the observed scene is accumulated over time to yield a pairwise affinity score between them. This yields an $n \times n$ affinity matrix W representing the interactions between all the observed people. The affinity matrix is then normalized such that each row sums to 1, yielding an asymmetric affinity W_{asym}. The i^{th} row in W_{asym} represents the distribution of interaction individual i has with everyone else in the scene (including themselves). The symmetrized version of this matrix W_{sym} is used to generate an embedding graph, where each individual is a node and the edges represent the interactivity between them.

5.2 Adult vs Child Classification

Inspired by the psychiatric work discussed in Chapter 2, distinguishing between child-child social relationships and child-adult relationships is an important aspect for behavioral analysis. Since all room occupants are tracked, some method to distinguish between child and adult tracks is required. Given the age differences between adults and child, height is a good attribute to make the distinction between child and adult; at preschool age even the tallest child will not be as tall as even short adults. Height is also a readily available statistic from the tracking, so it represents a straightforward and reliable way to perform this binary classification.

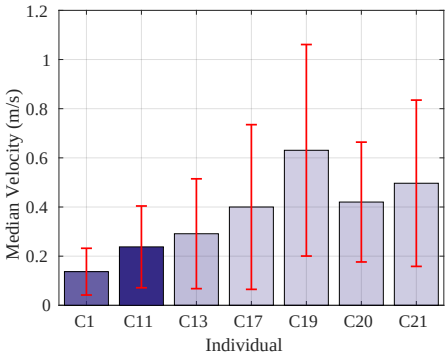
One approach, which was taken in earlier iterations of this system, is to approximate height with the σ_z^2 shape parameter within the Kalman filter (see Equation (3.36)). The height of a person is approximately equal to $6\sigma_z$, which corresponds to the 3σ-bounds of their point cloud distribution. A threshold height of $1.5m$ corresponded to $\sigma_z^2 = 0.0625$, and was used to distinguish the tracked individuals as adults or children. While this approach worked as an initial implementation, it is not very straightforward, as it uses a roundabout way to approximate height, and it turned out not to be as robust as would be desired.

While the Kalman Filter based tracker does not estimate height directly, this information can be approximated in another way from the observation bounding box. This data can then be stored as metadata along with the tracker and then used later for adult/child classification. While the height of the observation bounding box is not guaranteed to be the same as actual height of the individual (this might be true assuming pedestrians, but if the individual is not standing parallel to the Z-axis, their height is not the same as the bounding box height), this measure proved accurate in practice.

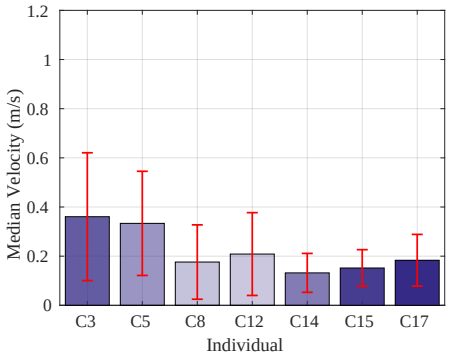
Ideally, if a track ever exceeds a height threshold, it would be considered an adult and if the threshold is never exceeded it would be considered a child. In practice, however, this is not robust, as it is possible an observation of a child could contain some spurious noise that makes the observation appear taller than the child actually is, or the observation could be merged with several individuals (including an adult). In order to make the classification more robust, each observation in the track is thresholded based upon bounding box height, which yields a sequence of 1's and 0's. The mean of that

sequence is then computed, which represents the percentage of observed frames that the track exceeded the height threshold.

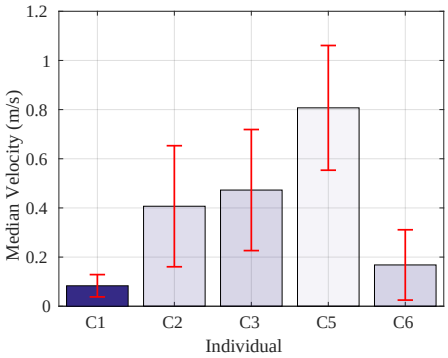
A second threshold is then applied, so that if the percentage of observed adult frames is above that threshold then the track is classified as an adult. This allows some frames to be classified as an adult, but if the track largely consists of child observations, it will still be classified as a child. In practice, a height threshold of $1.3m$ was applied and a track had to consist of 25% adult observations in order to be classified as an adult. These parameters yielded a 100% accuracy in classifying adult vs child tracks in the available data.



(a) Recording 1



(b) Recording 2



(c) Recording 3

Figure 5.1: Median velocities tracked children. Transparency indicates relative amount of time each child is observed; darker bars were observed for longer. The error-bars show the median absolute deviation. For recordings with a larger number of children, only the top 7 longest observed children are shown.

5.3 Quantifying Activity Level Through Velocity

Automatic analysis of the activity of a child would be a powerful tool, however this is difficult to achieve in practice and represents a considerable amount of work. Activity recognition is a very active field of research, and despite numerous innovative approaches performance on benchmark datasets still remains low. Therefore, a robust activity recognition system is outside the scope of this work. Instead, information related to activity is extracted which can be scrutinized directly or could be used indirectly to determine which sections of video to watch.

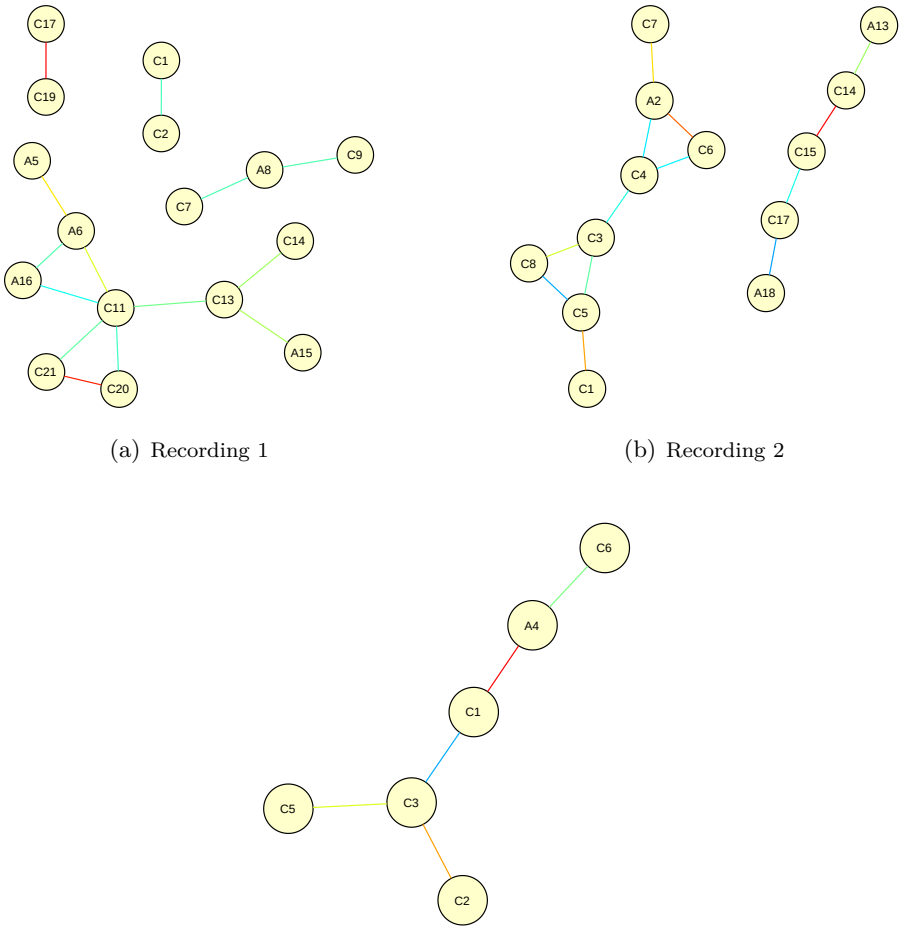
An important indicator of activity is velocity. A hyperactive child may spend much of their time running around the classroom, which will be reflected in a higher velocity. Even a child that is sitting and playing can have a different variance in velocity based on whether they are sitting still or moving around while they play. Velocity is also an attractive measure is its computation straightforward given the nature of the system: since occupant positions are tracked over time, the velocity of those occupants is easily computed.

A simple way to quickly summarizes an occupants activity level is to take the the median velocity over the period in which they were tracked. This will indicate if the track corresponds to a higher or lower activity level. In order to give an indication of how much the velocity varies over the course of observation, the median absolute deviation is also calculated. Additionally, the interactive data visualization tool allows the user to view the velocity plot over the course of the whole track.

5.4 Computed Results

Using the measures described in the previous sections, behavior analysis metrics can be computed on the groundtruth data presented in Chapter 4. The data in this chapter comes from the first three recordings, which have corresponding groundtruth. Additional data is incorporated in the next section, however it is beneficial to examine the behavioral analysis results on the groundtruthed data first, since there is a level of confidence in the input data.

Figure 5.1 depicts the median velocity of the tracked children. Due to the number of children in some recordings, the median velocity graphs were limited to the seven



(a) Recording 1

(b) Recording 2

(c) Recording 3

Figure 5.2: Graph embeddings from the symmetric affinity matrix W_{sym} depicting the interaction between different individuals. Prefix C indicates child, A indicates adult.)

most observed children. Since the recordings were taken during a free play session, the children in the classroom are playing in an unstructured environment, and their level of activity can vary drastically, frequently leading to large fluctuations in velocity. In particular, note that Child 1 in recording 3 has a small mean absolute deviation, since the child spends much of the observed time sitting at the table.

Figure 5.2 depicts the social relationship measures discussed above for all three recordings. For the graph embedding, nodes without edges have been omitted and

additionally some edges have been filtered to remove weak social relationships. Of note, for recording 2, there is a strong link between Child 1, who enters the room and sits down at the table to play and Adult 1 who enters the room and sits at the table to play with Child 1. These graphs also correspond to the same identities used in Chapter 4 Figure 4.6.

5.5 Interactive Visualization

In addition to the metric described above, an interactive tool was also developed to allow the social graph to be explored in more depth. This tool is based around an interactive view of the social graph, which represents the main screen of the program. Instead of representing nodes by a generic id, they are instead visualized with a particular image from one of the tracks, which allows for easy identification of the individual being represented. The positions of the graph nodes can be manually adjusted by clicking and dragging, plus several default graph layouts are possible (circular generally ends up being the most easy to read).

In addition to adjusting the location of the nodes in the social graph, the tool also allows the user to manipulate which edges are displayed. There are sliders that control both the minimum and maximum relationship score for the visible edges. Additionally, the user can double-click on nodes to only display edges connect to that node, useful for summarizing social relationships for a particular individual. As data is collected over a longer period of time, it is important give filtering options to the user, as the number of edges can quickly make it hard to interpret the social graph.

While a single edge between two nodes in the social graph indicates a relationship summary between two individuals, the details of the individual frame affinities can also be of interest. If two children share a strong summary relationship, it can be interesting to see how those affinities change over time, for example did they spend a lot of time together with a weak relationship or was there a few shorter periods with stronger affinities. Is a relationship for one child stronger, because that child had fewer other relationships? These kinds of questions can be answered by clicking on individual edges, which then brings up a plot of the recordings which contain non-zero affinities for the two individuals, and plots those affinities over each frame of the recording.

Drilling into the edge data is important, but the tool also presents more details on the tracked individuals. By right-clicking on the node pictures, a context menu presents options for various other pieces of information about individuals. Some of this information is useful for debugging purposes, such as as the height and σ_z^2 shape parameter which can be plotted against the child/adult threshold. This information is useful in debugging how well the child/adult classification parameters are working. Other information includes the velocity of the individual over the course of each recording, which frames within a recording the individual was tracked, and which recordings the individual appeared in along with their track ids.

Another useful feature for exploring an individual is the tracked frames affinity plot. While clicking on the social relationship edges allows a user to plot the affinity scores between two specific individuals, it may also be interesting to plot the affinity scores for a single individual across each recording they are tracked in. In the tracked affinity plot, the affinity scores between the target individual and every other individual they have a relationship with are plotted separately. Additionally, a dashed red line indicates which frames the target individual is tracked in, which can indicate frames where the individual is tracked but does not have a social relationship with anyone else.

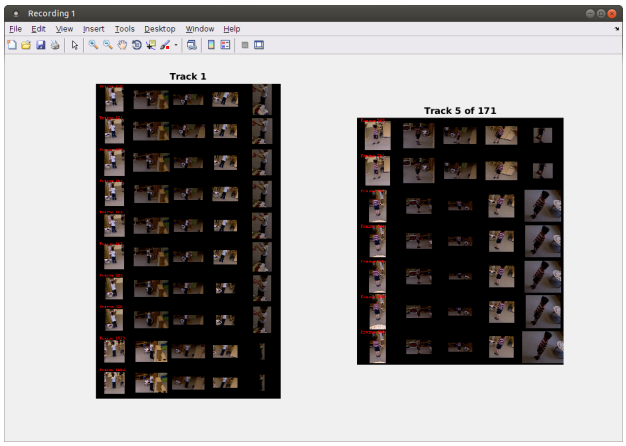
The last, and arguably most important, aspect of the interactive visualization tool is that each plot of information that is depicted allows the user to click and select windows of time, which get represented by red and green vertical bars. Once a window of time is selected, the user is asked for a sensor, and the video from that sensor for that time window is displayed. This allows a user to find an interesting period of time within the processed data, and view video data from that time from multiple perspectives. For instance, if two children have a high social affinity for a time period, a video of that sequence can be quickly returned. Additionally, if a child has weak or no links to anyone, a user can quickly see the times when that child was observed alone, and watch video pertaining to that time.

5.5.1 Reidentification

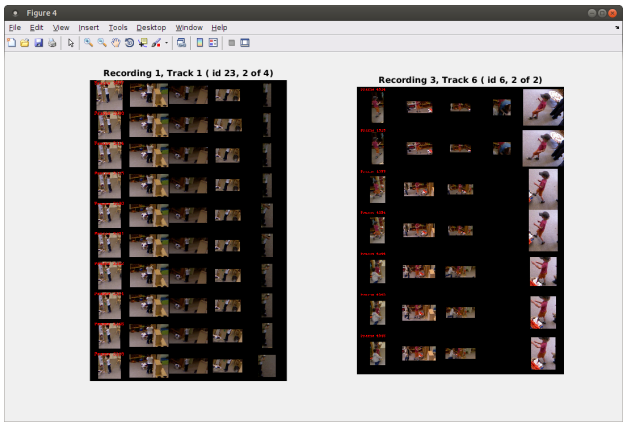
While Chapter 3 details some approaches to automating person reidentification, a manual approach to reidentification was taken here. This ensures that tracks are accurately matched, which is important for the behavioral analysis portion. In order to perform

this reidentification, several different applications were developed.

The first program allows manual reidentification within a single recording. A key to making the manual reidentification process manageable is to reduce the number of possibilities that need to be considered when making matches. This could be achieved by utilizing track descriptor comparisons, as previously discussed, however right now the matching process is purely manual. In order to reduce the number of possibilities, first tracks within a single recording are matched. The within-recording matching tool



(a) Within-Recording Matching



(b) Between-Recording Matching

Figure 5.3: Example images of the tools developed for manually matching tracks. (a) Tool for matching tracks within a single recording. (b) Tool for matching tracks between recordings.

presents two track summary images, one on the left which indicates the track being matched to. The user can then scroll through potential matches on the right using the left/right arrow keys. Pressing the 'M' key matches the currently selected tracks together and then prevents the matched track from being displayed again. Additionally, the 'W' and 'D' keys allow the user to change the currently selected track to match, in cases where it's advantageous to match other tracks first. When all matches are made for the currently selected track, the 'Enter' key is used to finishing matching and move on to the next track. Once all tracks are matched for the current recording, the program ends. See Figure 5.3(a) for examples of this tool.

Once matches are made within all recordings, tracks must be matched between recordings. A second between-tracks matching tool was created which operates similar to the within-tracks matching tool. The primary difference is that the tool must allow switching between recordings for candidate track matches and it also must allow switching the summary image within the currently matched track. In addition, it can give some summary information about the candidate track matches, such as the number of matches they have within the current recording. Figure 5.3(b) shows an example of this tool.

In addition to tools for performing manual matches, several other tools with a similar template were also created. First, tools were also created to be able to review the matches once made. This allows a user to go back and verify that the matches were correct. Both of these programs operate similar to the matching versions, sharing the same interface. Additionally, a tool was created to label tracks within a recording as non-person or not. Tracks marked as not being people were not displayed for matching in the between-recording matching tool. With 26 matched person tracks, only two matched non-person tracks had to be removed.

5.6 Discussion

A total of 11 recordings taken from the Shirley G. Moore Laboratory school were processed using the described system. Due to the sensor placement, one of the recordings was ignored for analysis because a large amount of activity in the recording took place at the very edge of the field of view of several sensors. This causes occupants to blink

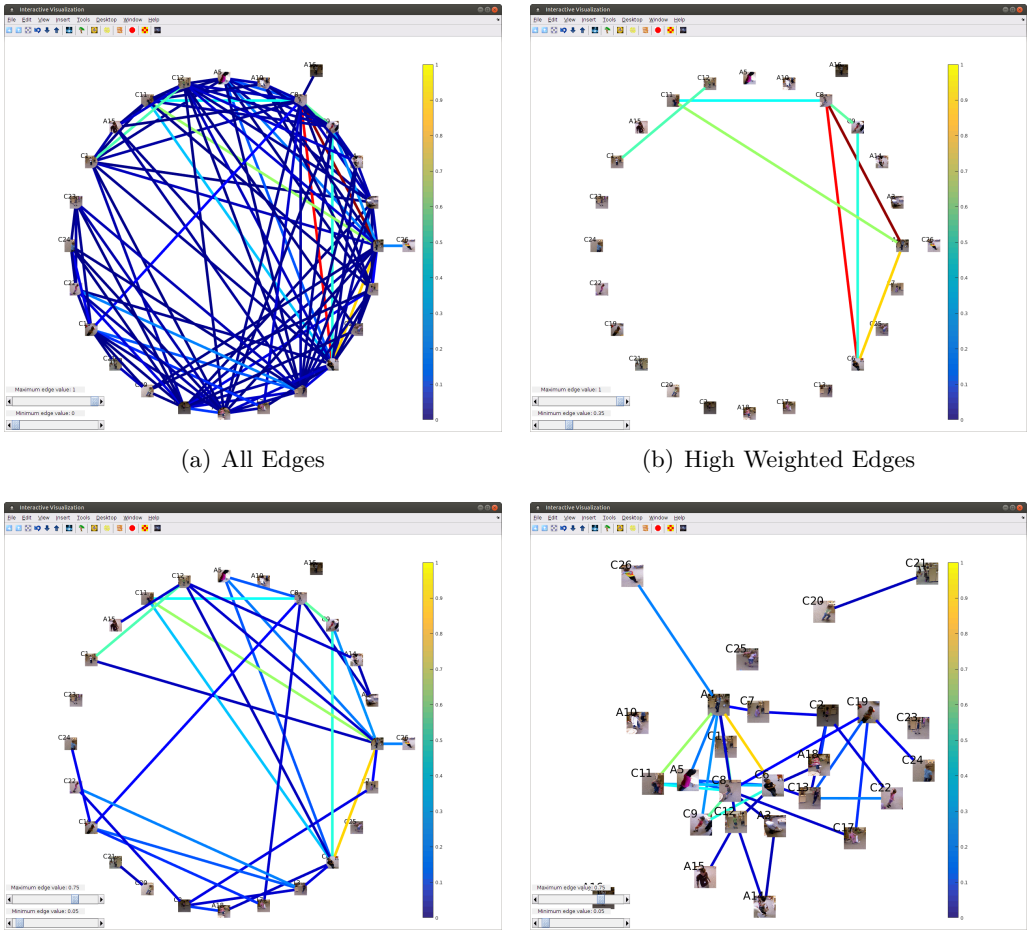
in and out of existence and lead to an abnormal amount of label switching making the data unreliable. These kinds of issues can be resolved by having more rigorous sensor placement, an aspect that would be improved in later iterations of this system. Of the remaining 10 recordings, a total of $75,173$ frames were processed, which amounts to over 40 minutes of recording time. After manually reidentifying the tracks and removing non-person tracks, 26 tracks remain. Table 5.1 indicates the total number of observations for each of these tracks.

Using the methods discussed in this chapter, all of the processed and reidentified recordings had social affinities computed and were fed into the interactive visualization tool. An overview of the resulting social graph is depicted in Figure 5.4.

Using the data derived from the interactive tool, several things are immediately obvious. First, there are two individuals in the recording time that only have one relationship edge: A16 and C26. Conversely, there are also a few individuals who have very strong relationships with another individual, for instance C8 has strong links to both C6 and A4. Besides that, it can also be seen that there are numerous weak

Number of Observations for Each Track			
C1	5,727	A3	2,007
C2	4,003	A4	18,800
C6	15,852	A5	6,285
C7	1,998	A10	518
C8	20,710	A14	810
C9	4,862	A15	1,658
C11	10,770	A16	196
C12	6,134	A18	7,237
C13	13,662		
C17	1,898		
C19	6,124		
C20	672		
C21	337		
C22	2,166		
C23	2,929		
C24	1,040		
C25	206		
C26	1,909		

Table 5.1: The number of observations for each track after manual reidentification.



(a) All Edges

(b) High Weighted Edges

(c) Filtered Edges

(d) Alternate Layout

Figure 5.4: Graph embeddings from the symmetric affinity matrix W_{sym} from the interactive visualization tool using manually reidentified data from all recordings.

relationships amongst most of the rooms occupants, so many so that it can be difficult to see the graph without filtering some of those weak relationships.

To illustrate the power of this tool, first the individuals with only one edge will be considered. Why are these individuals not related to more room occupants? First, consider plots of their affinity scores depicted in Figure 5.5(a)-(b). As depicted in the figure, these affinity plots are not particularly helpful in this case, other than to give a more detailed description of the graph edges. However, looking at the observed frames is a bit more useful, as depicted in Figure 5.5(c)-(d). From the tracked frames plots it

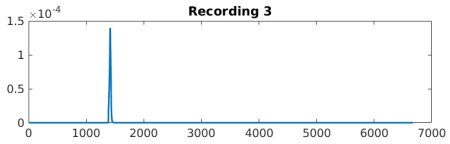
can be seen that these two occupants are not observed for a very long period of time (recordings without observations are not displayed). Using the interactive tool, a user can then click on time window of interest and view video different sensors.

By viewing the videos (which have an example frame depicted in Figure 5.5(e)-(f)), a user can discover the cause of these low social affinities for the two tracks. In the case of A16, the individual is only in the observed area very briefly, while walking from further into the classroom to the playground (recording 3, frames 300-450, the first spike in the tracked frames graph) and then back again (the second spike, recording 3 frames 1350-1450). While this occurs, only one child is in the room, and is sitting across the observed area so only a very minuscule affinity is computed. For C26, again the child is only observed for a short period of time (a single recording, for about a minute). This ends up being the only time the child is observed, which leads to a single edge in the social graph. This could represent an interesting time to watch the video, as it shows a lone adult and child interacting, and is the only time that child is observed.

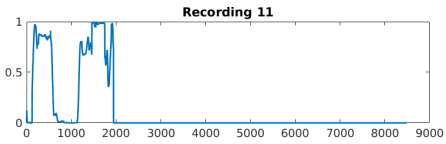
While this particular example may not seem rewarding, it illustrates several important points. One is that the data comes from a regular preschool classroom with a normal population of children performing their standard routines: it is quite possible that for any particular period or recording there may not be extraordinary data. Additionally, these tools provide another way to gain confidence in the output of the system. Any abnormalities in the system output could be discovered through the interactive visualization tools, and these tools could help to debug these abnormalities.

Another thing to note is that there are a lot of very small social affinities, which can be filtered out by adjusting the minimum threshold for graph edges. By adjusting the minimum threshold and examining the social graph, other individuals can be found who have only a single strong social affinity, for example see Figure 5.4(b). From this, it can be seen that C1 shares a singular moderately high social relationship with C12. This relationship is examined in further detail in Figure 5.6, with (a) depicting the social graph thresholded and centered on C1.

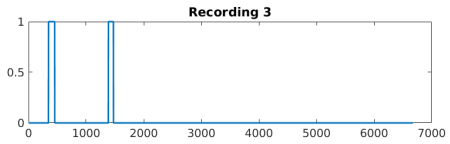
The first step in exploring this relationship is to look at the social affinities for C1 plotted against the tracked framed, depicted in Figure 5.6(b). Recording 1 contains several sections were C1 is tracked, and also has some sporadic relationships with other individuals, however Recording 2 is where C1 has sustained large affinities with C12.



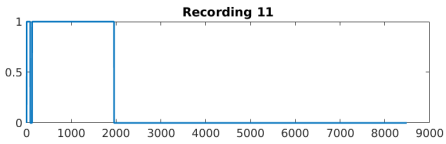
(a) A16/C8 Affinity Scores



(b) C26/A4 Affinity Scores



(c) A16 Tracked Frames



(d) C26 Tracked Frames

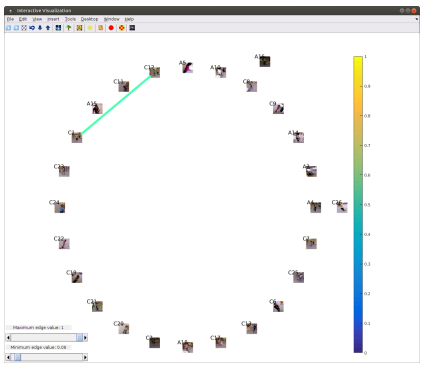
(e) Example Video Frame - A16

(f) Example Video Frame - C26

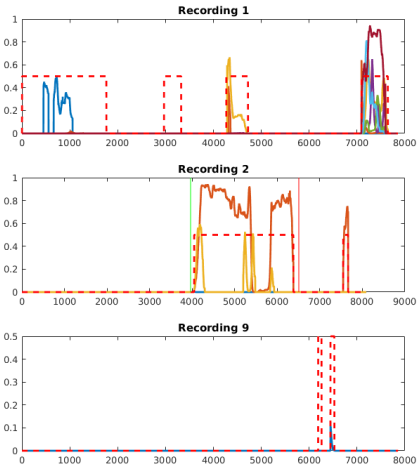
Figure 5.5: Exploring data about two room occupants who only have a social relationship to one other room occupant. Information includes a plot of the affinity scores, tracked frames, and then an example frame from a video during measured social affinity.

When performing the normalization, the relationships in Recording 1 end up getting marginalized since they are generally smaller and less sustained compared to the relationship with C12 in Recording 2. It is also worth noting that the interaction in Recording 2 is largely solely between C1 and C12; there is a small and sporadic contribution from A15, as they walk through the room, but the relationship is week.

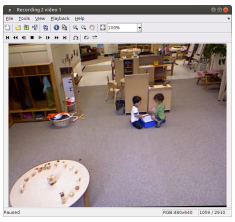
Since there's a sustained social affinity between two children, the interactive visualization tool can be used to pick out a video subsequence from that time. Figure 5.6(c)-(h) depicts 6 example frames from this interaction. This interaction begins when C1 enters from further back in the classroom, carrying a blue and white cooler, which contains a small pink teapot. C1 sits down on the floor, opens his cooler and brings out the



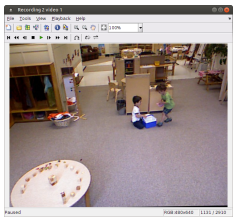
(a) Adjusted Social Graph for C1



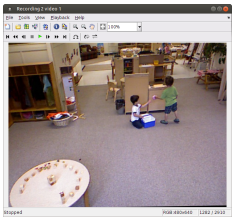
(b) Social Affinity Plots with Tracked Frames



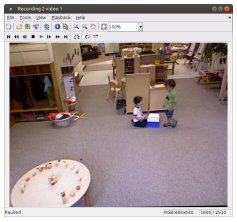
(c) Example Frame 1



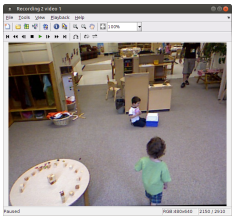
(d) Example Frame 2



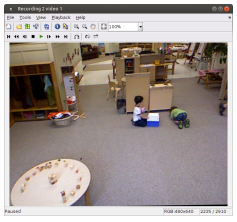
(e) Example Frame 3



(f) Example Frame 4



(g) Example Frame 5



(h) Example Frame 6

Figure 5.6: Exploring data about C1 in particular. (a) Depicts the social graph specific to C1, after applying a minimum threshold to graph edges. (b) Plots of social affinity scores for room occupants across different recordings plotted along with the frames in which C1 is tracked (dashed red line). (c)-(h) Example video frames showing C1 playing with C12, from a time window indicated with the red/green vertical bars in (b).

teapot. Then, C1 comes in and sits down, taking the teapot to look at. He examines the teapot, hands it back to C1, then runs around the area, comes back and rolls on the floor.

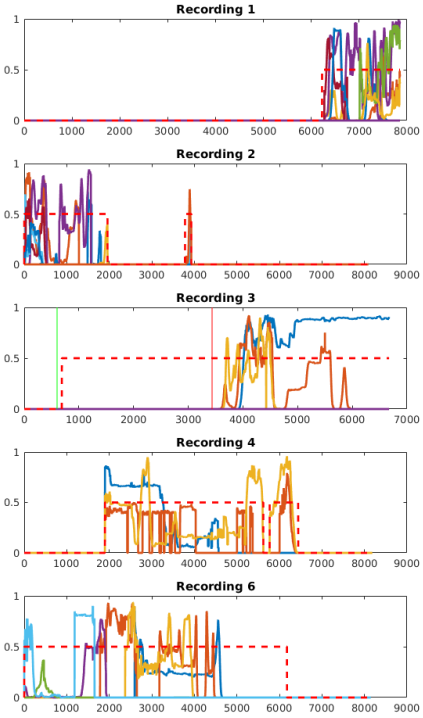
This sequence is an example of isolated interaction between two children and can give many clues about the relationship between these two children. This example indicates

the analytic power of this interactive visualization tool when it comes to discovering interesting video sequences in a larger corpus of data. Without such tools, all of these videos would have to be watched closely by a trained individual.

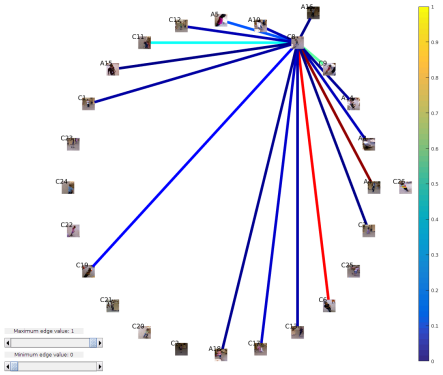
After examining occupants with only a single edge in the social graph, instead consider room occupants who have a high social affinity with other classroom occupants. Referring back to Figure 5.4(a)-(b) it can be seen that C8 has the two highest social relationships with other room occupants. What information can be learned about C8? First, from Table 5.1, it can be see that C8 has the most observations of any track (20,710 observed frames), followed next by A4 (with 18,800 frames) who shares the strongest social relationship with A4. The first thing that can be done to help analyze C8 is to filter out the graph edges so that only the edges from C8 are displayed, see Figure 5.7(b). This allows the social graph to be customized to C8 in particular, so that all edges connecting C8 can be see, even with a low minimum threshold, which would normally lead to a very large number of edges.

While adjusting the social graph to only show edges from C8 is helpful as a summary, information about the social affinity scores across each frame is lost. This information is useful, as it can indicate at what times C8 was playing with other room occupants, and also if C8 spent any time alone. More detailed information about the progression of social affinity scores is gained through the tracked frames affinity plot, depicted in Figure 5.7(a). Here, for each recording with observed frames, there is a plot of the affinity scores for each other room occupant who had a non-zero affinity score with C8. Additionally, the frames that C8 was tracked in are indicated by a dashed red line.

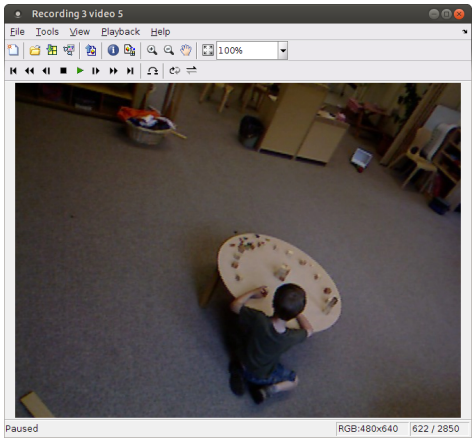
From the tracked frames affinity plot, it can be seen that there are several periods of time where C8 was interacting with a number of room occupants (especially recording 1 towards the end, and recording 2 towards the beginning). However, interestingly, C8 has a large window of time where they are tracked but not interacting with anyone else. In this particular set of recordings, this seems like abnormal behavior for C8, so that time frame might be of interest. Using the interactive visualization tool, the tracked frames affinity plot can be clicked on to extract video from a specific time window, with an example frame of this video depicted in Figure 5.7(c). This video depicts C8 walking into the classroom from the playground, and wandering over to sit down at the table and play with blocks.



(a) Social Affinity Plots with Tracked Frames



(b) C8 Social Graph



(c) Example Video Frame

Figure 5.7: Exploring data about C8 in particular. (a) Plots of social affinity scores for room occupants across different recordings plotted along with the frames in which C8 is tracked (dashed red line). (b) The social graph with only edges for C8 displayed. (c) An example video frame showing C8 playing alone, from a time window indicated with the red/green vertical bars in (b).

While more analysis could be performed on this data, these examples illustrate the power of the interactive visualization tool. By manipulating the visualization, the data can be explored, and interesting periods within the recordings can be discovered. This can vary, depending on whether the user is interested in exploring room occupants who do not have strong relationships with many people, or room occupants who are alone for a significant period of time. The tool also allows exploring individuals who have

strong relationships with others, by visualizing how the social affinity scores vary as the occupant is tracked. Additionally, the data allows exploring videos between child-to-child relationships as well as child-to-adult relationships.

Through the use of this interactive visualization goals, the designed system is able to meet the initial goals. This visualization provides an overview of the social relationships within the classroom through the initial social graph, and these social relationships can be further examined in detail through various means. Additionally, the activity level vis-a-vis velocity along with other measures can be explored by right-clicking on the individuals in the graph. It is worth noting, however, that these measures are only an initial sampling. The system was designed to be modular, and incorporating additional measures would be possible.

Chapter 6

Conclusion & Future Work

6.1 Summary

This thesis outlines the proposal for a multi-sensor RGB-D system for monitoring the behavioral of children in a preschool classroom. The proposed system utilizes the 3D information generated by the sensors to detect and track occupants in a very complicated environment. The system was designed to be inexpensive, so that it could be quickly and easily installed in new locations. The system was also designed to be non-intrusive, so that it would not alter the natural classroom environment.

The proposed system was installed in the Shirley G. Moore Laboratory School, a research preschool classroom at the University of Minnesota. The system was used to take recordings of an actual preschool class, with absolutely no instructions given to the class or its instructors. Data recorded from this classroom was hand-labeled to generate groundtruth, and this groundtruth data was used to validate the efficacy of the proposed system. The results of this validation indicate that this system shows promise, and is able to handle the difficult environment it was developed for.

The system was also used to present interesting data on the activities of the occupants during a recorded period of time. An interactive tool was developed to explore this data, which could be used by psychiatrists to gain further insights into the development of children. The system was validated on over 22,619 frames of groundtruth data, with further experiments run on a wider dataset of 75,173 frames.

Since this data comes from a regular preschool classroom, there is no expectation

that any particular abnormal behavior will be found, however these experiments illustrate the power of such a system. The behavioral measures that the system currently allows were developed with guidance from the psychiatric community, and the system was designed to be extensible, so further measures could be added. The application of such a system to continuous data over a long period of time could help expose information that aids in the understanding of early childhood development and the early onset of mental disorders in a way that is infeasible without automation.

6.2 Contributions

The following are the main contributions of this thesis:

- Describing the development of a framework to fuse RGB-D data coming from multiple sensors into a global 3D point cloud using off-the-shelf consumer hardware.
- Developing a background subtraction approach that utilizes 3D information, which outperforms traditional RGB-based methods in experiments.
- Developing a method to utilize the global 3D point cloud for detecting occupants of the room, which involved a novel hierarchical approach.
- Developing a system for longterm tracking of detections (room occupants).
- Developing methods for using tracking data to analyze classroom behavior, such as the development of a proximity-based measure that is used to construction a social relationship graph.
- Developing an interactive visualization tool to allow experts to explore the resulting behavior data, enabling fundamental contributions to behavioral science.
- Developing methods to test and validate system performance, with and without hand-labeled groundtruth.

6.3 Future Work Directions

While this thesis presents a functional end-to-end system, there are many aspects of this system that can be explored and improved. The goal of the system is to allow for long-term analysis of a classroom and several components could be improved to further this goal. In particular, methodology needs to be further developed to aid the combination of different recordings to create a larger data corpus for behavioral analysis. While this work outlines approaches to re-identification, further work needs to be done to automate this process, a process that could generate its own dissertation. Another aspect, albeit less important, is enhancing the background subtraction method to dynamically adjust to the scene. While the static background model was sufficient for the experiments presented here, it could cause issues in certain circumstances. And finally, large-scale processing will require a more robust and automated method of data-stream temporal synchronization.

In addition to enhancements to the system, further mining of the collected data for the purposes of behavioral analysis would also be beneficial. The social graphs already computed provide a wealth of information, and while they can be explored manually, further automated analysis could significantly reduce workload. The following sections cover in more detail the different aspects of proposed research. Sections are presented based on the order they would appear in the processing pipeline, and each section discusses the importance and potential approaches.

6.3.1 Video Sequence Temporal Alignment

Currently, video sequences have a single temporal alignment that is performed manually. This is time consuming and represents a manual step that slows down the processing of data. In terms of stream-lining the data processing, it is important to be able to automatically find a common starting point for each recording. Without this, there is a significant amount of manual work required to align the videos.

The task of video alignment or synchronization is well researched, where most methods try to compute the similarity between two image sequences and maximize. For example, [110] presents a method for sequence alignment based on an extension of previous work in [111]. Here, the problem of image alignment between multiple sensors is

expanded to videos by computing normalized correlation on $7 \times 7 \times 7$ spatio-temporal cubes between sequences. Global similarity is computed the sum of normalized correlation across all the spatio-temporal cubes between two sequences and the transformation that maximizes this global similarity is found via Newton's method.

This method relies heavily upon the appearance across the video sequences, instead of focusing on the activities that are happening within the video sequences. Using descriptors that focus more on the activities being performed, such as optical flow, would allow the motion present in the scene to drive the alignment of video sequences. One such descriptor that has not been explored as much for video sequence registration is the log-polar histogram descriptor on self-similarity matrices (SSMs). This was explored in [112] for action recognition, but could also be used for video sequence alignment.

6.3.2 Adaptive Background Model

One issue not currently addressed in the current methodology is dynamically varying background. As it stands, the background model is learned *a priori* and is not updated during processing. This could lead to issue where static objects in the scene get moved and then get tracked. One avenue for further research would be to incorporate a dynamic update to the background model.

The risk in incorporating this dynamic update would be an increase in per-frame computation time. Additionally, issues arising from a static background model could potentially be dealt with in other ways. For instance, if a static object is moved and becomes tracked, the track could potentially be filtered out in post-processing.

A guiding principle for the background subtraction is that it must be fast. Speed considerations were one of the primary reasons why color information was dropped from the model: the computation involved added to the per-frame runtime, but did not improve performance appreciably. One simple approach for adding a dynamic update would be to utilize bit-shifting. Each voxel in the model could have an arbitrary bit length associated with it, depending on desired frame history. Starting from the least-significant bit, each bit would represent whether the past frames were occupied or empty. Voxel occupancy could then be considered as either an OR operation with all the bits in the frame history, or potentially a majority vote. At each frame, each occupied voxel would have the frame history bit-shifted with a 1 bit, and each unoccupied voxel would

be bit-shifted with a 0 bit.

6.3.3 Appearance Descriptors

While several appearance descriptors were discussed and compared in this work, there could be further analysis done on different appearance descriptors. Another avenue of approach would be to consider other 3D descriptors as well. Covariances-based descriptors have been explored as an option for point cloud descriptors [59], but there are also numerous other point-based descriptors as well: spin-images [113], FPFH [84], and SHOT [114] to name several. One of the downfalls of using descriptors based on point cloud data is that these descriptors are often more expensive to compute. It is also not straightforward to use these descriptors, as computing them on the global point cloud is going to be prone to errors: errors in the calibration of the point clouds are compounded by distortion in the depth image of the sensors to create inconsistencies in the global cloud registration. These inconsistencies will lead to issues with point cloud descriptors computed on the global cloud (although the descriptors could be computed on the per-sensor clouds).

6.3.4 Person Reidentification

While this work presents some experiments on using various descriptors for person reidentification, these descriptors perform best when several ranked options are supplied. This means they could work well for automating a manual approach, but they would not work well for a purely automated method. In order to further automate this system, further exploration of the person reidentification problem is warranted. The best approach for this would be to look at methods that combine a multitude of different descriptors in clever ways, such as the symmetry-driven accumulation of local features (SDALF) [115] or the later SDALF+C which incorporates covariance descriptors in SDALF [116].

6.3.5 Further Behavior Analysis

The social graphs computed as described in this work represent a currently-untapped wealth of information for automated analysis. Examining the relationships of room occupants over time can yield important information, as described in the background

chapter. The social graphs can provide information on the kinds of relationships room occupants have, such as if a child has stronger ties to adults than children, or if a child does not have any strong ties to anyone (*i.e.*, they are ostracized). With social graphs computed over a long enough timeframe, cliques could start to form, and the wealth of recent research on social graphs could become applicable. While this work allows a user to manually explore the social graph and associated data, many automated techniques could also be applied to these graphs, even some of the previously mentioned techniques could be applied like the clustering using either min-cut [81] or DBSCAN [85].

Instantaneous proximity information can also be important as well. For instance, if a child is often too close to other children. Very close proximity can represent an invasion of personal space, and can be associated with impulsiveness, a potential sign related to ADHD or Autism. Room location (or micro-geography as referred to in [21]), could also be important. Children returning to a specific spot in the classroom repeatedly, or spending an larger proportion of their time there. A tool could be created to tag specific room locations, so that the number of times an occupant returns to that location, or the duration spent at that location, could be measured.

There is also a possibility for incorporating video-based or point-cloud-based activity recognition. Previous work [117] has looked at the identify stereotypic behaviors in children from static camera video. Stereotypies occur in the normal population but are more prevalent in those diagnosed with Autism or Rhett's syndrome, and are something that warrants more psychiatric research. Activity recognition could also be applied to other disorders, such as OCD, where repeated behaviors or a fixation on a particular location could be related to a compulsion.

Many of these ideas can be combined to create an automated "highlights reel" from video recorded (versus the manual approach to creating highlighted videos presented in this work). While making automatic assessments can be error prone, and having a trained psychiatrist watch 8 hours of video is time-consuming, this system could ingest 8 hours of video and distill it down to 15 minutes of useful information to show to a psychiatrist. The system could look for very specific behaviors, such as cases of persistent close proximity, or behavioral tics like stereotypies, or persistent cases of lone children, and return short video sequences of these behaviors.

References

[1] World Health Organization. The world health report 2004: Changing history, annex table 3: Burden of disease in DALYs by cause, sex, and mortality stratum in WHO regions, estimates for 2002. *Geneva: WHO*, 2004.

[2] World Health Organization. Global status report on non-communicable diseases. *Geneva: WHO*, 2010.

[3] T. Insel. National institute of mental health strategic plan. `http://www.nimh.nih.gov/about/strategic-planning-reports/index.shtml`, Aug. 2015.

[4] G. Esposito and P. Venuti. Analysis of toddlers' gait after six months of independent walking to identify autism: A preliminary study 1. *Perceptual and motor skills*, 106(1):259–269, 2008.

[5] G. Esposito, P. Venuti, F. Apicella, and F. Muratori. Analysis of unsupported gait in toddlers with autism. *Brain and Development*, 33(5):367–373, 2011.

[6] E. Walker, T. Savoie, and D. Davis. Neuromotor precursors of schizophrenia. *Schizophrenia Bulletin*, 20(3):441–451, 1994.

[7] P. M. Torrens and W. A. Griffin. Exploring the micro-social geography of children's interactions in preschool: A long-term observational study and analysis using geographic information technologies. *Environment and Behavior*, 2012.

[8] A. Legendre and D. Munchenbach. Two-to-three-year-old children's interactions with peers in child-care centres: Effects of spatial distance to caregivers. *Infant Behavior and Development*, 34:111–125, 2011.

[9] A. Ali and E. Dagless. Vehicle and pedestrian detection and tracking. In *IEE Colloquium on Image Analysis for Transport Applications*, pages 5–1. IET, 1990.

[10] J. D. Coie, N. F. Watt, S. G. West, J. D. Hawkins, J. R. Asarnow, H. J. Markman, S. L. Ramey, M. B. Shure, and B. Long. The science of prevention: A conceptual framework and some directions for a national research program. *American Psychologist*, 48(10):1013, 1993.

[11] I. A. C. Council. IACC report to the congressional appropriations committee on the state of autism research. `https://iacc.hhs.gov/reports/2004/report-to-the-congressional-appropriations-committee-april.shtml#research-matrix`, Apr. 2003.

[12] R. J. Landa and L. G. Kalb. Long-term outcomes of toddlers with autism spectrum disorders exposed to short-term intervention. *Pediatrics*, 130:S186–S190, 2012.

[13] S. J. Rogers, A. Estes, C. Lord, L. Vismara, J. Winter, A. Fitzpatrick, M. Guo, and G. Dawson. Effects of a brief early start denver model (ESDM)–based parent intervention on toddlers at risk for autism spectrum disorders: A randomized controlled trial. *Journal of the American Academy of Child & Adolescent Psychiatry*, 51(10):1052–1065, 2012.

[14] G. Dawson, S. Rogers, J. Munson, M. Smith, J. Winter, J. Greenson, A. Donaldson, and J. Varley. Randomized, controlled trial of an intervention for toddlers with autism: the early start denver model. *Pediatrics*, 125(1):e17–e23, 2010.

[15] T. H. McGlashan and J. O. Johannessen. Early detection and intervention with schizophrenia: rationale. *Schizophrenia Bulletin*, 22(2):201–222, 1996.

[16] L. Zwaigenbaum, S. Bryson, and N. Garon. Early identification of autism spectrum disorders. *Behavioural Brain Research*, 251:133–146, 2013.

[17] J. Schiffman, E. Walker, M. Ekstrom, F. Schulsinger, H. Sorensen, and S. Mednick. Childhood videotaped social and neuromotor precursors of schizophrenia: a prospective investigation. *American Journal of Psychiatry*, 2015.

[18] J. Elicker, K. M. Ruprecht, and T. Anderson. Observing infants' and toddlers' relationships and interactions in group care. In *Lived Spaces of Infant-Toddler Education and Care*, pages 131–145. Springer, 2014.

[19] J. Bowlby. *A Secure Base: Clinical Applications of Attachment Theory*. Taylor & Francis, 2005.

[20] C. Garvey. *Play*. Harvard University Press, 1990.

[21] P. M. Torrens and W. A. Griffin. Exploring the micro-social geography of children's interactions in preschool: A long-term observational study and analysis using geographic information technologies. *Environment and Behavior*, 45(5):584–614, 2013.

[22] J. M. Rehg, G. D. Abowd, A. Rozga, M. Romero, M. A. Clements, S. Sclaroff, I. Essa, O. Y. Ousley, Y. Li, C. Kim, et al. Decoding children's social behavior. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3414–3421. IEEE, 2013.

[23] J. Hashemi, M. Tepper, T. V. Spina, A. Esler, V. Morellas, N. Papanikolopoulos, H. Egger, G. Dawson, and G. Sapiro. Computer vision tools for low-cost and noninvasive measurement of autism-related behaviors in infants. *Autism Research and Treatment*, 2014.

[24] I. Laptev. On space-time interest points. *International Journal of Computer Vision (IJCV)*, 64(2–3):107–123, 2005.

[25] S. S. Rajagopalan, A. Dhall, and R. Goecke. Self-stimulatory behaviours in the wild for autism diagnosis. In *IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 755–761. IEEE, 2013.

[26] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf. Survey of pedestrian detection for advanced driver assistance systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(7):1239–1258, 2010.

[27] M. Enzweiler and D. M. Gavrila. Monocular pedestrian detection: Survey and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(12):2179–2195, 2009.

[28] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893. IEEE, 2005.

[29] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(9):1627–1645, 2010.

[30] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5. `http://people.cs.uchicago.edu/~rbg/latent-release5/`.

[31] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision (IJCV)*, 88(2):303–338, 2010.

[32] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[33] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[34] S. Blackman and R. Popoli. *Design and analysis of modern tracking systems.* Boston, MA: Artech House, 1999.

[35] L. Haritaoglu, D. Harwood, and L. Davis. W4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(8):809–830, August 2000.

[36] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(7):780–785, 1997.

[37] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Online multiperson tracking-by-detection from a single, uncalibrated camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(9):1820–1833, 2011.

[38] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial & Applied Mathematics*, 5(1):32–38, 1957.

[39] O. Tuzel, F. Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classification. In *Lecture Notes in Computer Science*, volume 3952, pages 589–600. Springer, 2006.

[40] F. Porikli, O. Tuzel, and P. Meer. Covariance tracking using model update based on lie algebra. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 728–735, Jun. 2006.

[41] Z. Khan and I.-H. Gu. Tracking visual and infrared objects using joint Riemannian manifold appearance and affine shape modeling. In *IEEE Conference on Computer Vision Workshops (CVPRW)*, pages 1847–1854, Nov. 2011.

[42] M. Chen, S. K. Pang, T. J. Cham, and A. Goh. Visual tracking with generative template model based on Riemannian manifold of covariances. In *Proceedings of the 14th International Conference on Information Fusion*, pages 1–8, Jul. 2011.

[43] Y. Wu, J. Cheng, J. Wang, H. Lu, J. Wang, H. Ling, E. Blasch, and L. Bai. Real-time probabilistic covariance tracking with efficient model update. *IEEE Transactions on Image Processing*, PP(99):1, 2012.

[44] K. Sankaranarayanan and J. Davis. Object association across PTZ cameras using logistic MIL. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3433–3440, Jun. 2011.

[45] Y. Zhang and S. Li. Gabor-LBP based region covariance descriptor for person re-identification. In *Sixth International Conference on Image and Graphics*, pages 368–371, Aug. 2011.

[46] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38, Dec. 2006.

[47] Y. Bar-Shalom, P. K. Willett, and X. Tian. *Tracking and data fusion.* YBS publishing, 2011.

[48] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer. Multi-camera multi-person tracking for EasyLiving. In *IEEE International Workshop on Visual Surveillance*, pages 3–10. IEEE, 2000.

[49] D. Focken and R. Stiefelhagen. Towards vision-based 3-d people tracking in a smart room. In *IEEE International Conference on Multimodal Interfaces*, pages 400–405. IEEE, 2002.

[50] A. Mittal and L. S. Davis. M_2Tracker: a multi-view approach to segmenting and tracking people in a cluttered scene. *International Journal of Computer Vision (IJCV)*, 51(3):189–203, 2003.

[51] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(2):267–282, 2008.

[52] M. Munaro and E. Menegatti. Fast RGB-D people tracking for service robots. *Autonomous Robots*, 37(3):227–242, 2014.

[53] C. Madden, E. D. Cheng, and M. Piccardi. Tracking people across disjoint camera views by an illumination-tolerant appearance representation. *Machine Vision and Applications*, 18(3-4):233–247, 2007.

[54] M. Farenzena, L. Bazzani, A. Perina, V. Murino, and M. Cristani. Person re-identification by symmetry-driven accumulation of local features. In *IEEE Conference Computer Vision and Pattern Recognition (CVPR)*, pages 2360–2367. IEEE, 2010.

[55] B. Prosser, W.-S. Zheng, S. Gong, and T. Xiang. Person re-identification by support vector ranking. In *Proceedings of the British Machine Vision Conference*, pages 21.1–21.11, 2010.

[56] D. Gray and H. Tao. Viewpoint invariant pedestrian recognition with an ensemble of localized features. In *European Conference on Computer Vision (ECCV)*, pages 262–275. Springer, 2008.

[57] L. Bazzani, M. Cristani, A. Perina, and V. Murino. Multiple-shot person re-identification by chromatic and epitomic analyses. *Pattern Recognition Letters*, 33(7):898–903, 2012.

[58] O. Tuzel, F. Porikli, and P. Meer. Pedestrian detection via classification on Riemannian manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(10):1713–1727, Oct. 2008.

[59] D. A. Fehr. *Covariance based point cloud descriptors for object detection and classification*. PhD thesis, University of Minnesota, 2013.

[60] A. Cherian, S. Sra, A. Banerjee, and N. Papanikolopoulos. Jensen-bregman logdet divergence with application to efficient similarity search for covariance matrices. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(9):2161–2174, 2013.

[61] R. Vezzani, D. Baltieri, and R. Cucchiara. People reidentification in surveillance and forensics: A survey. *ACM Computing Surveys (CSUR)*, 46(2):29, 2013.

[62] A. Bedagkar-Gala and S. K. Shah. A survey of approaches and trends in person re-identification. *Image and Vision Computing*, 32(4):270–286, 2014.

[63] S. Gong, M. Cristani, S. Yan, and C. C. Loy. *Person re-identification*, volume 1. Springer, 2014.

[64] N. Walczak, J. Fasching, W. Toczyski, R. Sivalingam, N. Bird, K. Cullen, V. Morellas, B. Murphy, G. Sapiro, and N. Papanikolopoulos. A nonintrusive system for behavioral analysis of children using multiple RGB+Depth sensors. In *IEEE Workshop on Applications of Computer Vision*, Jan. 2012.

[65] A. Maimone and H. Fuchs. Reducing interference between multiple structured light depth sensors using motion. In *Virtual Reality Workshops (VR)*, pages 51–54. IEEE, 2012.

[66] A. Teichman, S. Miller, and S. Thrun. Unsupervised intrinsic calibration of depth sensors via slam. In *Robotics: Science and Systems*, volume 248, 2013.

[67] D. Herrera, J. Kannala, and J. Heikkilä. Joint depth and color camera calibration with distortion correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(10):2058–2064, 2012.

[68] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge Univ Press, 2nd edition, 2000.

[69] T. Svoboda, D. Martinec, and T. Pajdla. A convenient multi-camera self-calibration for virtual environments. *PRESENCE: Teleoperators and Virtual Environments*, 14(4):407–422, August 2005.

[70] J. Fasching, N. Walczak, R. Sivalingam, K. Cullen, B. Murphy, G. Sapiro, V. Morellas, and N. Papanikolopoulos. Detecting risk-markers in children in a preschool classroom. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1010–1016. IEEE, 2012.

[71] J. Yao and J. Odobez. Multi-layer background subtraction based on color and texture. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, Jun. 2007.

[72] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58:11:1–11:37, Jun. 2011.

[73] Z. Lin, M. Chen, and Y. Ma. The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. Technical Report UILU-ENG-09-2214, UIUC, Oct. 2010.

[74] C. Qiu and N. Vaswani. ReProCS: A missing link between recursive robust pca and recursive sparse recovery in large but correlated noise. *arXiv preprint arXiv:1106.3286*, 2011.

[75] N. Walczak, J. Fasching, W. D. Toczyski, V. Morellas, G. Sapiro, and N. Papanikolopoulos. Locating occupants in preschool classrooms using a multiple rgb-d sensor system. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2166–2172. IEEE, 2013.

[76] R. Sivalingam, A. Cherian, J. Fasching, N. Walczak, N. Bird, V. Morellas, B. Murphy, K. Cullen, K. Lim, G. Sapiro, and N. Papanikolopoulos. A multi-sensor visual tracking system for behavior monitoring of at-risk children. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1345–1350, May 2012.

[77] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer Series in Statistics, 2001.

[78] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[79] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(8):888–905, 2000.

[80] D. Wagner and F. Wagner. Between min cut and graph bisection. *Mathematical Foundations of Computer Science*, pages 744–750, 1993.

[81] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)*, 59(2):167–181, 2004.

[82] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnörr. Real-time optic flow computation with variational methods. In *Computer Analysis of Images and Patterns*, pages 222–229. Springer, 2003.

[83] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter. Voxel cloud connectivity segmentation-supervoxels for point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2027–2034. IEEE, 2013.

[84] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (FPFH) for 3D registration. In *IEEE International Conference on Robotics and Automation*, pages 3212–3217. IEEE, 2009.

[85] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *International Conference on Knowledge Discovery and Data Mining*, volume 96, pages 226–231. AAAI, 1996.

[86] S. T. Birchfield and S. Rangarajan. Spatiograms versus histograms for region-based tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1158–1163. IEEE, 2005.

[87] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(3):480–492, 2012.

[88] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Log-euclidean metrics for fast and simple calculus on diffusion tensors. *Magnetic Resonance in Medicine*, 56(2):411–421, 2006.

[89] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

[90] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 15(12):3736–3745, 2006.

[91] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *IEEE Transactions on Image Processing*, 17(1):53–69, 2008.

[92] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. R. Bach. Supervised dictionary learning. In *Advances in Neural Information Processing Systems*, pages 1033–1040, 2009.

[93] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1794–1801. IEEE, 2009.

[94] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

[95] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[96] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, Mar. 2010.

[97] T. Guha and R. Ward. Learning sparse representations for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(8):1576 –1588, Aug 2012.

[98] I. Ramirez, P. Sprechmann, and G. Sapiro. Classification and clustering via dictionary learning with structured incoherence and shared features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3501–3508. IEEE, 2010.

[99] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": Interactive foreground extraction using iterated graph cuts. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 309–314. ACM, 2004.

[100] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html.

[101] B. Keni and S. Rainer. Evaluating multiple object tracking performance: the CLEAR MOT metrics. *EURASIP Journal on Image and Video Processing*, 2008.

[102] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics,*, 9(1):62 –66, Jan. 1979.

[103] J. Yao and J. Odobez. Fast human detection from videos using covariance features. *8th European Conference on Computer Vision Visual Surveillance Workshop (ECCV-VS)*, Oct. 2008.

[104] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933.

[105] L. Van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[106] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[107] T. Pang-Ning, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Pearson, 2006.

[108] H. Karcher. Riemannian center of mass and mollifier smoothing. *Communications on Pure and Applied Mathematics*, 30(5):509–541, 1977.

[109] M.-C. Chang, N. Krahnstoever, and W. Ge. Probabilistic group-level motion analysis and scenario recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pages 747–754, Nov. 2011.

[110] Y. Ukrainitz and M. Irani. Aligning sequences and actions by maximizing space-time correlations. In *European Conference on Computer Vision (ECCV)*, pages 538–550. Springer, 2006.

[111] M. Irani and P. Anandan. Robust multi-sensor image alignment. In *IEEE International Conference on Computer Vision (ICCV)*, pages 959–966. IEEE, 1998.

[112] I. Junejo, E. Dexter, I. Laptev, and P. Pérez. View-independent action recognition from temporal self-similarities. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(1), 2011.

[113] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[114] F. Tombari, S. Salti, and L. D. Stefano. A combined texture-shape descriptor for enhanced 3d feature matching. In *EEE International Conference on Image Processing (ICIP)*, pages 809–812. IEEE, 2011.

[115] L. Bazzani, M. Cristani, and V. Murino. Symmetry-driven accumulation of local features for human characterization and re-identification. *Computer Vision and Image Understanding*, 117(2):130–144, 2013.

[116] S. J. Poletti, V. Murino, and M. Cristani. SDALF+C: Augmenting the SDALF descriptor by relation-based information for multi-shot re-identification. In *Iberoamerican Congress on Pattern Recognition*, pages 423–430. Springer, 2013.

[117] J. Fasching, N. Walczak, V. Morellas, and N. Papanikolopoulos. Classification of motor stereotypies in video. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.