AUTOMATIC PEOPLE COUNTING AND MATCHING

by

John A. Sallay

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Electrical Engineering

Approved:

Dr. Scott Budge Major Professor	Dr. Jacob Gunther Committee Member
Dr. Donald Cripps Committee Member	Dr. Byron R. Burnham Dean of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2009

Copyright © John A. Sallay 2009

All Rights Reserved

Abstract

Automatic People Counting and Matching

by

John A. Sallay, Master of Science

Utah State University, 2009

Major Professor: Dr. Scott Budge
Department: Electrical and Computer Engineering

The thesis explores software algorithm for implementing a people counting and matching system to be used on a bus. A special camera is used, known as a texel camera, that generates depth and color information for a scene. This added information greatly facilitates both the tasks of matching and counting.

Although people counting is a relatively mature field, there are several situations in which current technologies are not able to count correctly. Several of these difficult situations are tested with 82% counting accuracy.

The idea of matching people on a bus is also developed. The goal is not to identify a specific person on a bus, but to find the time that a specific person is on the bus, and what bus stops were used. There are several aspects of this matching problem that differentiate it from other classification tasks that have been researched. In this thesis, multiple measurements are used to classify a person and sequence estimation techniques explored. The techniques developed classify with 92% accuracy, even with a relatively large number of people on a bus.

(95 pages)

Acknowledgments

There have been many who have offered me great support as I have endeavored to do this thesis. I would like to thank all of the many people that walked under my camera frame. In particular the Fall 2009 ECE 1000 class was of great help in my data gathering effects. I would also like to thank Breton Prall and Kevin Neilsen for listening to my ideas and offering feedback. My major professor gave me inspiration to do the project and helped me along the way. Lastly, I thank my wife for allowing me the time to spend many long hours working to complete this thesis.

John Sallay

Contents

List of Tables

List of Figures

Acronyms

APC	Automatic People Counter
BCJR	Bahl, Cocke, Jelinek, and Raviv
FOV	Field of View
FPS	Frames per Second
HSI	Hue, Saturation, and Intensity
HTC	Handheld Texel Camera
IR	Infrared
KNN	K-Nearest Neighbor
LDA	Linear Discriminate Analysis
LIDAR	Light Detection and Ranging
QDA	Quadratic Discriminate Analysis

Chapter 1

Introduction

A transit system uses statistical information about its ridership in order to plan routes and schedule transportation. Several methods of obtaining this information exist, ranging from a driver counting the number of people who get on at each stop to approaches requiring far more sophisticated technology.

A system that counts the number of people entering and exiting a given area is known as an Automatic People Counter (APC). APCs have long been of interest to the transit industry. Many counters, such as those produced by Traf-sys [1] and Acorel [2], use an infrared (IR) beam to count people. In the past few years, video counters have become popular. APCs are employed in many situations but this thesis will focus on their uses in buses.

Though many solutions exist, there are still situations that consistently present difficulties to APCs. The greatest weakness is that of counting groups. An active IR sensor transmits a beam to a receiver; if the beam is broken, a count is incremented. Oftentimes people enter side-by-side or right behind one another. The active IR sensor does not have the ability to distinguish people in these situations. Video-based counters also suffer from the same problem. It can be hard to separate people from the scene background and from each other.

A people counter provides information about how many people entered and exited a bus at each stop that it makes. This associates each stop with a certain number of people. There is other information that could be useful to a transit authority. It would be useful to know at which stops specific people enter and exit. This would associate each person with two stops. A transit system could use this information to plan routes based on which stops are used in conjunction with each other, instead of making decisions based solely on

the number of people who utilize each stop. It is necessary to match an exiting person to a person who previously entered in order to obtain this information.

This thesis explores a solution to both of these problems. The focus will be on software algorithms, but it is important to understand the hardware used in this thesis work to capture the data needed for both counting and matching people. The camera used is known as a texel camera. A texel camera is the combination of a Light Detection and Ranging (LIDAR) camera with a color camera. A general overview of LIDAR and texel technology is given in secs. 1.1 and 1.2, respectively.

1.1 LIDAR Technology

Light Detection and Ranging (LIDAR) is a technology that provides the distance between the sensor and an object of interest. In the simplest of cases, a known pulse of light is transmitted. It reflects off of a surface and returns to the camera. The distance between the camera and the object can be easily found because the pulse travels at a constant speed, the speed of light. This is known as the time of flight principal.

A more complex kind of LIDAR emits a continuous wave of modulated light instead of a pulse. The time of flight cannot be found with the continuous wave; however, the phase difference between the emitted and reflected light can be used to produce the same information. The cyclic nature of phase introduces an ambiguity in the distance. There is no way for the sensor to distinguish a phase of $\pi/4$ from $9\pi/4$. In many cases, the maximum distance to be measured in known *a priori* and the modulation frequency of the emitted light can be set so that no ambiguity occurs over the region to be measured. This is the case in the application explored in this thesis.

Canesta Inc. is the manufacturer of the LIDAR used for this thesis. In their cameras, two sensors are used to collect phase information. The sensors are synchronized with the phase of the outgoing light, with only one sensor on at a time. There is an extra phase ambiguity with this sensor setup. Two measurements are made 90° apart so as to completely resolve this ambiguity [3, 4].

There exist two main classes of LIDAR imaging devices, scanning and flash. A scanning

LIDAR sweeps laser pulses through a scene gathering the depth information for one spatial location at a time. A flash LIDAR is similar to a CCD camera. It has an array of sensors that gather depth information at the same time. A flash LIDAR can capture images faster than a scanning LIDAR and does not have any mechanical parts; though, it needs to have a light source for every depth sensor in the array. The camera used throughout this thesis is a flash LIDAR.

1.2 Texel Technology

A texel camera is the combination of a LIDAR with an electro-optical (color) camera [5]. Depth and color information are collected simultaneously, which provides more information than is readily available in other video-based systems.

There exist other ways of producing similar information. One very popular method is stereo vision and triangulation. Two cameras are set up at known locations relative to each other. Correspondences are found in each image taken. Based upon the location of the pixels in the images and the location of the cameras, distance information can be gathered. Post processing has to be performed in order to fuse the information together. A texel camera provides this same information from direct measurements, which can increase accuracy and reduce the number of computations to be performed [6].

The LIDAR and color cameras used in this thesis are co-boresighted, which is to say, that although the two cameras have different optics and sensors, they receive the same light. The cameras are mounted 90° apart. A cold mirror is used to separate the light into two bands. It is positioned at a 45° angle in between the two cameras. The texel camera used is shown in fig. 1.1. The light in the visible spectrum is reflected into the lens of the color camera and the light in the infrared (IR) band passes through the cold mirror into the lens of the LIDAR camera. The response of the cold mirror used is shown in fig. 1.2. In this manner, both cameras receive different bands of light from the same locations in a scene, at the same time. Further information on texel cameras is given by Boldt [5].

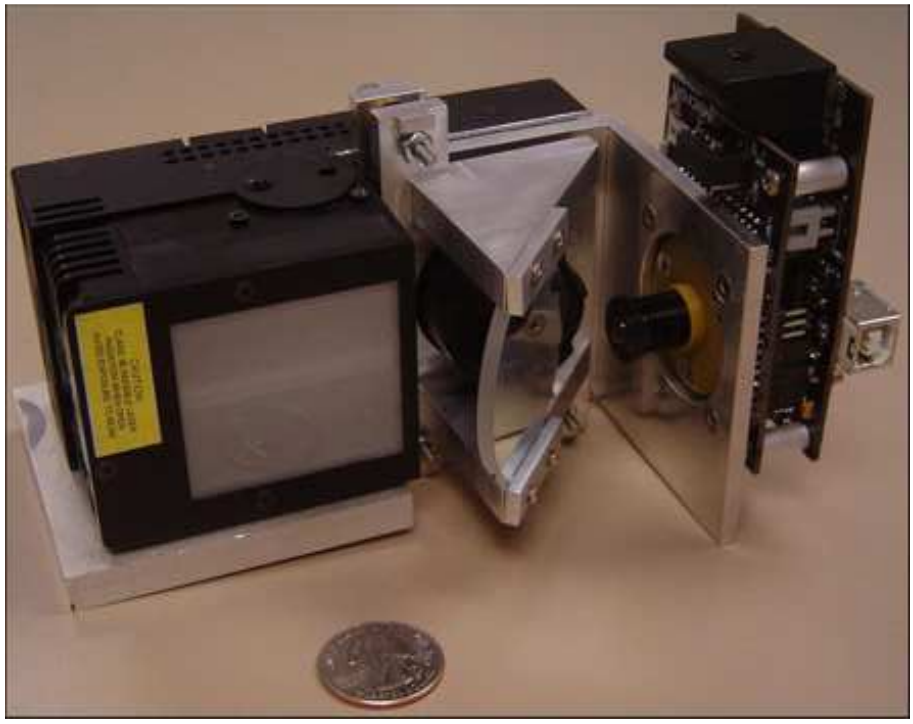


Fig. 1.1: Prototype texel camera.

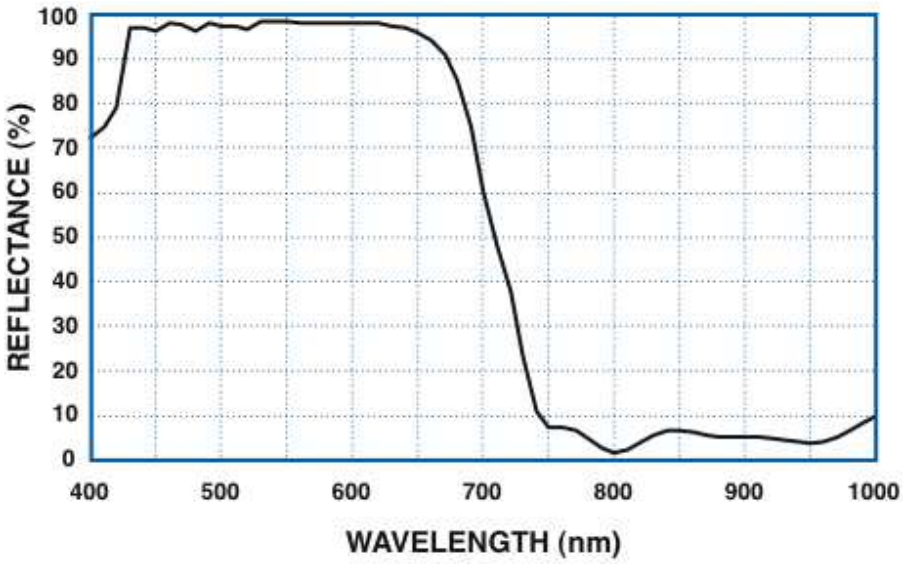


Fig. 1.2: Cold mirror transfer characteristics.

1.3 Anthropometry

Anthropometry is the study of human measurements. In some studies, hundreds of measurements are taken from different parts of the body [7]. In this thesis, only two of those measurements are used, height and head-to-shoulder distance. Data has been collected on tens of thousands of individuals over several years. Figure 1.3 shows a histogram of the ratio of height to head-to-shoulder distance on data taken from 8051 individuals, which was taken from a government database of anthropometric measurements [7]. This information will later be used to separate the image of a person's head from an image of their shoulders.

1.4 Previous Work in People Counting

The most common issues confronted when designing a people counter are: shadows, changes in background illumination, and occlusion [8]. A common approach to reduce occlusion is to mount the counting device above a doorway [8–10].

Göktürk and Tomasi use the same LIDAR camera employed in this thesis as a head tracker [6]. A stream of images is collected as a person enters. In every frame the person's head is manually found. Information from these frames is averaged together and used to track the head as the person exits. The algorithm performs relatively well, but requires a human being to manually sort through every frame in a training sequence.

The Multiple Hypothesis Tracking Algorithm (MHT) is commonly used for tracking objects [11–13]. For each measurement, a set of hypothesis are formed for the number of objects to be tracked in a scene. A tree is formed linking the time steps together, and a probability is assigned to each leaf. The leaf with the highest probability is chosen as the correct set of objects and motion. The algorithm naturally accounts for people entering, exiting, running into each other, and false alarms. Polat *et al.* use the algorithm to track body parts using motion and color [11].

Albiol mounted a gray scale camera above the doorway of a train [10]. Only three rows from each image are kept. All of the rows that are kept from a stop are stored together and processed at the same time. This significantly reduces the memory and computational requirements of the system.

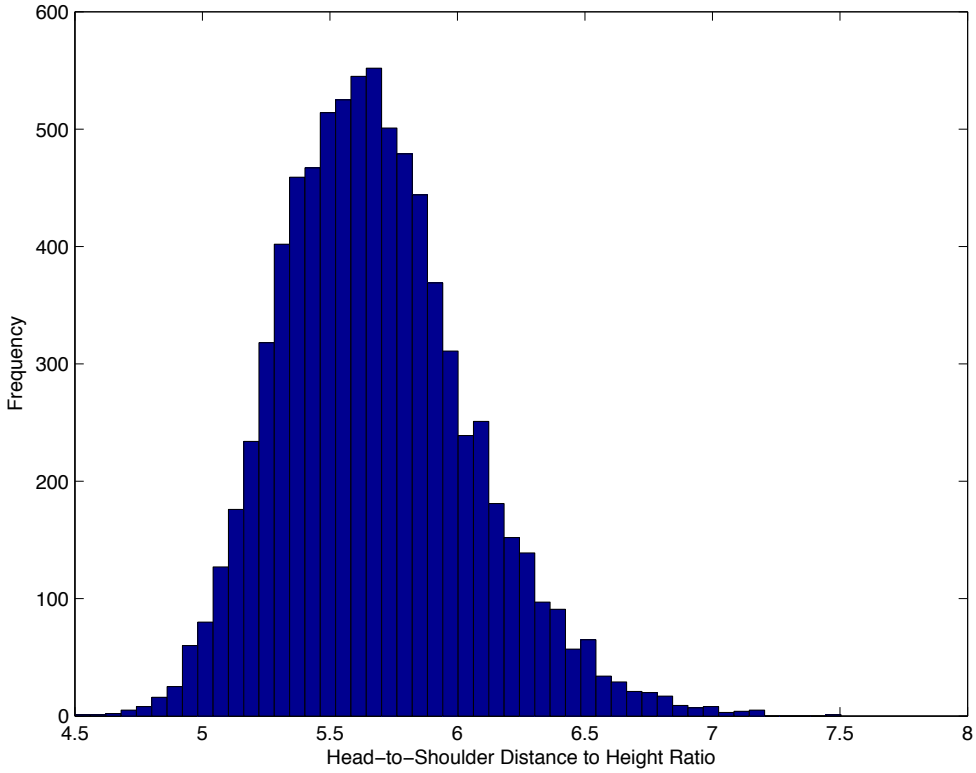


Fig. 1.3: Histogram of head-to-shoulder distance to height ratio.

Chen *et al.* tracked people via block motion prediction. Each image is divided into 16x16 pixel blocks. A motion vector is predicted for each block by comparing subsequent frames. Segmentation is performed via a region growing technique in all of the disjoint regions of the images. This method is able to segment people correctly as long as they form disjoint regions in the image.

Zhao and Nevatia employ a model-based approach [14]. Head, shoulders, and legs are modeled as ellipses in a given pose. A database of possible poses is stored. When a blob is found in the video stream, it is compared against all poses in the database. If a match is found, the blob is flagged as a person, and will be tracked until leaving the camera's field of view. A statistical model is used to separate people who are close together.

A graphical approach is used by Masoud and Papanikolopoulos [9]. Motion segmentation is performed via frame subtraction. The difference between two successive frames is viewed as motion. Features are gathered for each blob in a given frame, such as size and position. The features are compared from frame-to-frame and a correspondence graph is created. The graph shows the possible associations between regions in different frames. This is especially useful when people are close together. It may not be possible to distinguish two people who are close together in the first frame that they appear. After several frames they could break apart and would be easily distinguished. The algorithm can also distinguish people who start out separated but join together before exiting the camera's field of view.

1.5 Previous Work in People Matching

Much of the previous work in people matching has been focused on facial recognition. The human face has a wealth of features that can easily be tracked. Many methods, such as Fishers Linear Discriminant Analysis [15], have been developed to exploit these features.

Multiple measurements of each person are collected before a match is made. The information can be combined to produce a more accurate and robust classification. Bagui *et al.* use a modification of the nearest-neighbor rule on all of the measurements collected [16]. Shakhnarovich creates a distribution from the measurements and the Kullback-Leibler

divergence is used [17]. The Kullback-Leibler divergence is a measure of the similarity of probability distributions. Roy and Khattree modified Linear Discriminate Analysis (LDA) to be optimal for multiple measurements in an autoregressive system [18].

It is possible to formulate a trellis solution to the matching problem, but the complexity of the problem is to large to be solved. A common solution is to use a beam search [19, 20]. In the standard beam search algorithm the W best paths are found at each time step, where W is the width of the beam. The remaining paths are removed and only these W paths are propagated at the next time step.

1.6 Purpose of Thesis

The purpose of this thesis is to develop software algorithms that implement an automatic people counting and matching system using a handheld texel camera (HTC). The counter will be tested in various scenarios in which existing counters do not perform well, such as multiple people exiting at the same time.

The problem of matching people is unique in many ways. This thesis will show that it is possible to accurately match people in a bus environment. Different techniques will be developed and tested. Multiple measurements will be available to classify each person. Extensions will be made to some existing classification techniques to account for the multiple measurements.

1.7 Thesis Overview

This thesis describes the algorithms used and the tests performed on a people counting and matching system. The counting algorithm needs to be running constantly while people are entering and exiting a bus. The matching algorithm only needs to be performed once per bus stop. The matching algorithm relies on information gathered during the counting process, but the counting algorithm does not need any information from the matching algorithm. For this reason the algorithms will be treated separately.

Chapter 2 outlines the counting algorithm used. The algorithm will be broken down in several steps and each step will be explained separately. Data collection and a few special

hardware considerations are also explored. Chapter 3 explains what features were collected and how the best features were found. It then explores various methods of matching people using multiple measurements. Structures are presented that allow decisions to made using previous and future information. Chapter 4 outlines the tests performed on several systems and the results obtained. Chapter 5 provides a set of conclusions and presents some ideas for further research.

1.8 Summary of Contributions

A list of the contributions specific to this thesis are given below.

- Use of a texel camera in a people counting system.
- Testing of the counting system in a variety of situations in which current people counters do not perform well.
- Extensions of several classification techniques to include multiple measurements.
- Development of an ad hoc people matching algorithm.
- Development of an optimal people matching approach, and a suboptimal approximation to it.

The problem of classification of people also is unique. Each time a person enters a bus a classifier is created. Thus, the number of classes is constantly changing. Any classifier used will have to be constantly updated based upon the number of people currently under consideration.

Chapter 2

People Counting

This chapter explains the counting algorithms used. Figure 2.1 gives a high-level overview of the different processing steps. There are three major processes which are shown as dashed boxes in the figure. Sections 2.1 - 2.3 explore data collection and preprocessing. Segmentation is discussed in secs. 2.4 - 2.7. Sections 2.8 - 2.9 explain how region associations are made. Counting is a simple task once all of the other steps have been performed. Section 2.10 explains the counting procedure. The performance of the counter is analyzed in sec. 2.11. All of the input and output variables referenced in this chapter refer to fig. 2.1.

2.1 Data Collection and Special Hardware Considerations

The texel camera gathers three images at a time: depth, brightness, and color. The depth and brightness images come from the LIDAR camera while the color image comes from the color camera. An example of each kind of image is shown in fig. 2.2.

The depth and brightness images are pseudo-colored. In the brightness image, blue implies that the sensor collected little light and red implies that it collected more light. In the depth image, blue pixels are closest to the camera while red pixels are further away. The LIDAR camera has a sensor array of 64x64 pixels.

Minimum and maximum brightness thresholds are used. Any brightness value falling below or above these thresholds causes the corresponding value in the depth image to be set to 0. The pixels remain unchanged in the brightness image. All of the pixels that are thresholded out using the brightness filter appear blue, even though they may correspond to distances far away from the camera.

The color image is similar to what would be found in a normal digital camera. It

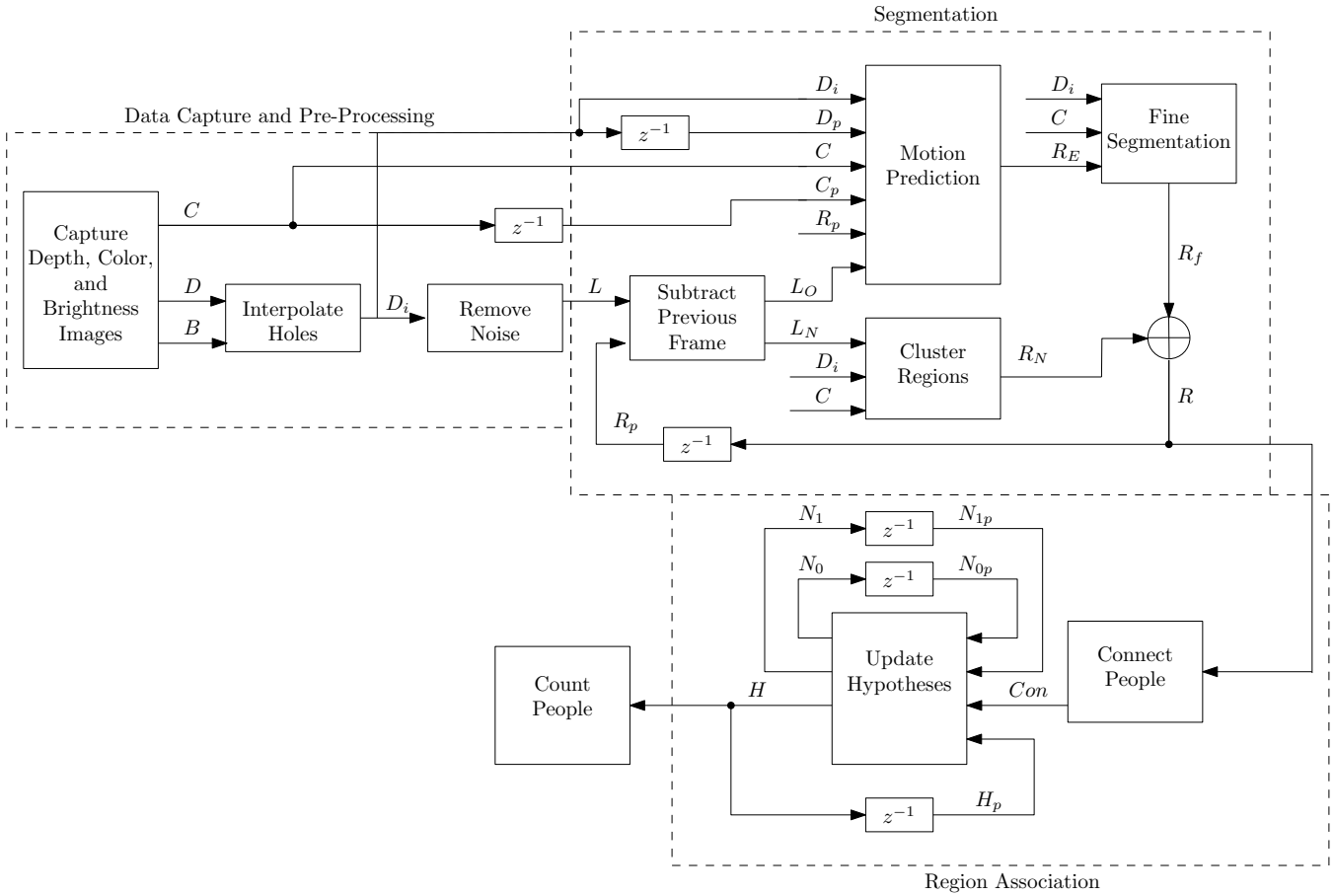
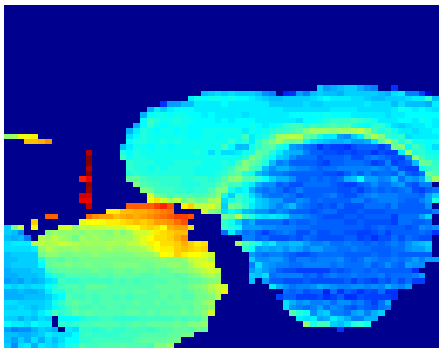
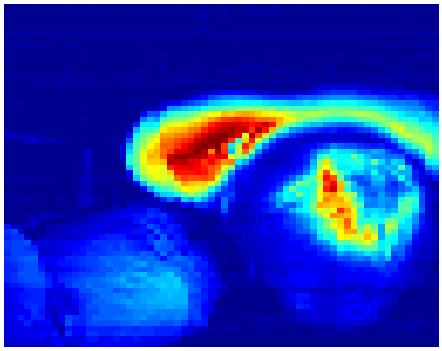


Fig. 2.1: Counting flow chart.



(a) Depth image



(b) Brightness image

(c) Color image

Fig. 2.2: Sample depth, brightness, and color images.

outputs three channels of color: red, green, and blue. The color camera has a sensor array of 1280x1024 pixels.

The two cameras capture data from the same scene; however, the field of view (FOV) of the two cameras is not the same. The color camera has a much wider FOV. The FOVs of the two cameras are matched together via a calibration process. The size of the resulting color image is one of the parameters of the calibration. For this application an image of 256x256 pixels was chosen. This provides color at a higher resolution than the depth image, but it not so large so as to be computationally prohibitive. Further information on the calibration process used is given by Boldt [5].

A portable camera frame was built to simulate the way the texel camera would be mounted in a bus. Measurements for the frame were taken from a 2007 Gillig ski bus. This style of bus does not have any steps at any of the entrances. The distance from the floor to the highest portion of the ceiling is 96 inches. The camera ideally would be mounted at this height in the bus, near the middle of the ceiling. If the camera were mounted near the door, then a person's depth would change as they stepped onto the bus, which would complicate the problem to be solved.

In a real-life scenario, the system would process the data as it was captured. It need not be processed in real time because there is time in between bus stops that can be used for processing. It does need to finish processing all of the data from a bus stop before the next stop occurs. For the purposes of this thesis the data is captured and processed off-line.

The flash LIDAR camera used was acquired several years ago and was one of the first of such cameras to be manufactured. There are some technical issues present that would not be present in a later generation texel camera.

The LIDAR emits 785 nm laser light [21]. Ideally, the cold mirror in the texel camera reflects only visible light into the color camera and transmits all of the light that the LIDAR emits back into the its own sensor array. However, many of the color images are affected by the LIDAR's emitted light. The color camera interprets this wavelength as red, and thus, many of the color images contain significantly more red than would ideally be present.

The amount of red introduced varies from image to image; thus, it cannot easily be removed. Sometimes, the red channel saturates, making it impossible to recover the correct color information. Color information is used to track and to match people. The addition of the light from the LIDAR makes it impossible to obtain consistent color measurements for a person.

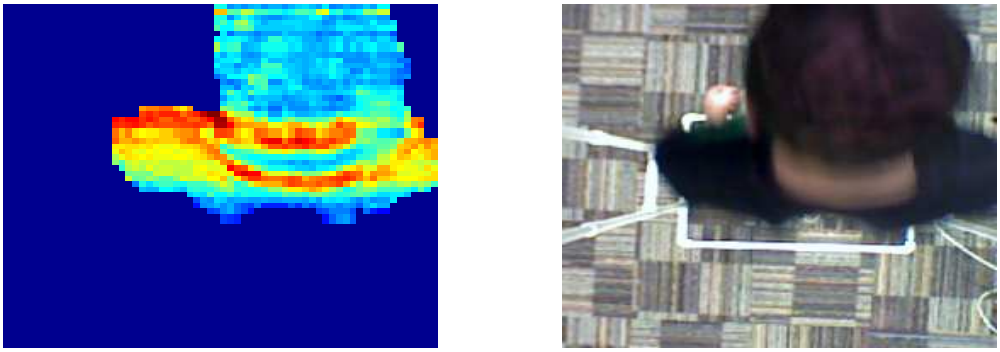
In an effort to compensate for this problem the red channel of the color image is not used. This decision was made because the algorithms were able to perform adequately without the red channel, and because the issue can be solved in hardware in the next generation texel camera.

The system also does not deal well with motion. A person standing still underneath the camera produces relatively consistent measurements. The standard deviation was found to be about 2.5 cm. However, a person moving underneath the camera has a much higher standard deviation, up to 30 cm. There is also a spatial correlation among measurement errors. The measurement error oftentimes effects entire regions of an image. The errors are not consistent and do not appear to be directly correlated with a person's speed. Figure 2.3 shows an example. The head is colored blue and there are two streaks that are red corresponding to the back. These colors are just as they should be. However, the shoulder blades appear to be at approximately the same distance from the camera as the head is.

If a person stands still underneath the camera, the depth information tends to be accurate and consistent for each frame. As soon as the person starts moving, effects such as those in fig. 2.3 occur. Walk error, the change in depth measurements due to the reflectivity of surfaces, may be partially responsible. Motion, however, also plays a significant role in the error.

The cause is not entirely understood, but it makes it nearly impossible to get consistent measurements. All of the current LIDAR cameras researched have some sort of motion blur reduction technology. It is assumed that a different camera will mitigate this problem.

Another issue is the maximum frame rate that can be obtained. The current camera and software cannot capture data faster than about 8 frames per second (fps). In many



(a) Depth image (b) Color image

Fig. 2.3: Motion error example.

motion-based applications, data is captured at a rate of 25 - 30 fps [8,10]. The capture rate on this camera is significantly slower than that used for similar applications.

The solution to the motion and capture rate problems is as follows. A person walks under the camera frame at about $\frac{1}{3}$ of their normal pace. This simulates a camera with a higher frame rate. Although this is not the ideal solution, it mitigates both of the aforementioned problems and provides a solution that is closer to what would be expected with a different camera.

The majority of the algorithm developed should function independent of the LIDAR camera used. Some parts of it address problems that are assumed to be specific to this camera. When this is the case it will be clearly stated at the beginning of the section.

2.2 Interpolate Holes

The current LIDAR camera is not able to gather accurate measurements in bright light. Saturation can occur, removing depth values that could be necessary for the rest of the algorithm. This part of the algorithm is assumed to be specific to the flash LIDAR used.

This is the only part of the algorithm where the brightness image is used. The camera has a brightness filter that sets to 0 any depth pixels whose brightness value falls below or

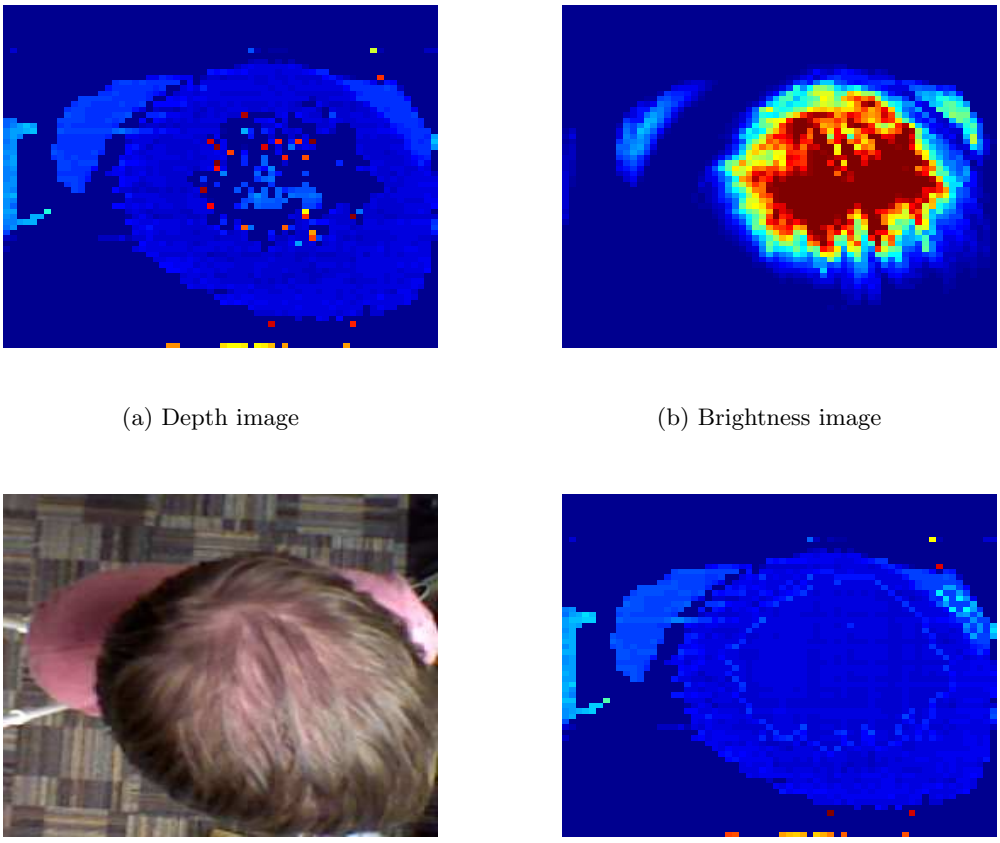
above a minimum or maximum threshold. This removes the least reliable data, but can cause some undesirable effects. Sometimes the majority of a person's head is eliminated due to brightness saturation. This causes several difficulties throughout the rest of the algorithm.

The solution chosen is to fill in holes via interpolation. An example of a hole is shown in fig. 2.4 (a), where a large portion of the depth pixels associated with the person's head have been set to 0. The interpolation is only performed using the captured depth and brightness images, D and B (see fig. 2.1). Only pixels whose brightness value falls above the maximum brightness threshold are changed. The depth values for these pixels was set to 0 during image capture. Interpolation is performed in the horizontal and vertical directions and the two results are averaged together. One line of the image is analyzed at a time. If the line contains saturated pixels, a non-saturated pixel is found on the left and on the right of (or above and below) the saturated region. Linear interpolation is performed to fill in the depth values in between the bounding pixels. Figure 2.4 (a) - (c) shows the depth, brightness, color images for a sample frame. Figure 2.4 (d) shows the output depth image after interpolation. The resulting depth image is called the interpolated depth image, D_i.

The solution is by no means optimal. It was designed so that it could be easily removed without disturbing the rest of the counting algorithm. It should be possible in the next texel camera built to eliminate the problem. The holes oftentimes occur in bright ambient light. One company, PMD Technologies, produces cameras with "Suppression of Background Illumination" technology. The camera sensors provide accurate measurements without saturation, even in direct sunlight [22].

2.3 Background and Noise Removal

Background and noise removal is a simple task using a texel camera. Only the depth image is considered at this stage in the processing. The first step is to median filter the interpolated depth image, D_i. This removes salt and pepper noise and smooths the image. Next, a depth threshold is applied to any pixel that falls outside a reasonable depth range. Upper and lower depth thresholds are set. Any pixel with a depth value falling above or



(a) Depth image

(b) Brightness image

(c) Color image

(d) Interpolated output image

Fig. 2.4: Interpolation example.

below these thresholds is set to 0.

The remaining pixels are grouped together using a recursive labeling technique [23, pp. 151-152]. The procedure is outlined in Algorithm 2.1. The image is separated in four-connected groups of pixels.

The size of each region in the resulting label image, L, is found, and a size threshold is used to eliminate small regions. The size of a region is dependent on how far away from the camera it is. An object will appear much larger in an image when it is near the camera than when it is far away. Thus, the mean depth value is incorporated into the size threshold as

$$minSize_i = \frac{T}{d_i}, \tag{2.1}$$

where T is the depth threshold, d_i is the average depth value of the ith region, and $minSize_i$ is the size threshold for the ith region. Any region smaller than the size threshold is considered background, and every pixel in the label image that corresponds to this region is set to 0. This prevents small objects from cluttering the scene. The depth image is not changed; however, L is used as a mask whenever the depth image is used.

An example of the background and noise removal process is shown in fig. 2.5. Note that the colors are scaled to produce the full range of colors from blue to red in both images.

2.4 Subtract Previous Frame

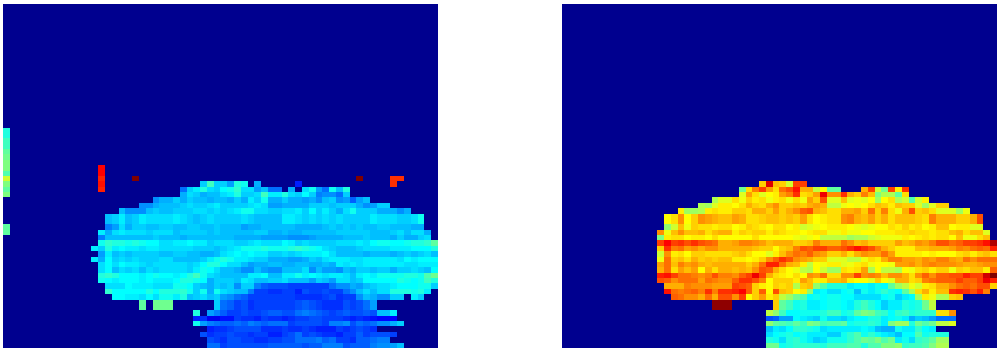
Previous frame subtraction is a simple technique for detecting motion in an image. When little motion occurs from one frame to the next, the difference between the two frames is small. When there is a significant amount of motion between frames, the difference is large. A person moving quickly through the frame can pass from one end of the frame to another in about 0.5 seconds. This corresponds to 10 - 15 frames (with the simulated faster capture rate), which is to say that a person can reasonably move through about $\frac{1}{10}$ of the frame at a time. Any motion greater than this would be considered significant.

The process used is essentially a simplified version of optical flow. The difference image

Algorithm 2.1 Recursive Region Labeling

Input:
Gray Scale Image, B,
Maximum Region Label, maxLabels
Output:
Label Image, L
Begin
Initialize nextColor = 1, EquivColors[maxLabels] = $\{0, 0, \dots\}$, notdone = true
For Each Pixel, (i, j)
If $B(i, j) \neq 0$
If $B(i-1, j) \neq 0$ and $B(i, j-1) = 0$
$L(i, j) = L(i-1, j)$ /* Same region as upper pixel */
End
If $B(i-1, j) = 0$ and $B(i, j-1) \neq 0$
$L(i, j) = L(i, j-1)$ /* Same region as left pixel */
End
If $B(i-1, j) \neq 0$ and $B(i, j-1) \neq 0$
$L(i, j) = L(i, j-1)$ /* Same region as left pixel */
/* Mark two regions as equivalent */
EquivColors[$max(L(i, j), L(i, j-1)] = min(L(i, j), L(i, j-1))$
End
If $B(i-1, j) = 0$ and $B(i, j-1) = 0$
$L(i, j)$ = nextColor /* Create new region */
nextColor = nextColor +1
End
End
End

While notdone = true
notdone = false
For Each Pixel, (i, j)
If EquivColors[$L(i, j)$] $\neq L(i, j)$
/* Assign pixels to equiv region */
$L(i, j)$ = EquivColors[$L(i, j)$]
notdone = true
End
End
End
End



(a) Depth image (b) Masked depth image

Fig. 2.5: Background and noise removal example.

is used to find when a person enters the scene. It is not used for motion tracking, as is often done in optical flow.

The label image, L, from sec. 2.3, and the previous region label image, R_p, are used to create difference images. The region label image is discussed in sec. 2.5.

In this algorithm, previous frame subtraction is used to detect when a person enters the frame. At a sufficient image capture rate, the amount of motion between frames will be small as a person moves through the camera's field of view. A large difference will occur when a new person enters the scene. The size threshold from sec. 2.3 is used on the differenced image. A small region will correspond to a moving person. If the size of the new region is below the size threshold, and it is bordering one of the regions from the previous frame, then all of the pixels in the new region are assigned to the previous region.

Two labels images are the output of this stage in the algorithm: the existing person label image, L_E, which contains all of the regions that were contained in the previous image, and the new person label image, L_N, which contains all of the new regions. Sample input and output images are shown in figs. 2.6 and 2.7. Figure 2.6 shows the frame difference as a person moves through a scene. Note that the difference image produces a rather small region, due only to the motion of the person in the scene. The size of the region falls below the size threshold and, as can be seen in fig. 2.6 (d), all of the pixels are assigned to the

region in the existing person label image. The new person label image is not shown because it does not contain any non zero values. Figure 2.7 shows the operation when a new person enters. The difference image is not shown because it is approximately the same as the new person label image. Previous and current depth images are shown instead of their respective label images. The depth images contain more detail than the label images, which can easily be derived from the depth images shown.

The existing person label image from the previous step, L_E, is used for motion prediction as described in sec. 2.5 and the new person label image, L_N, is clustered as described in sec. 2.7.

2.5 Motion Prediction

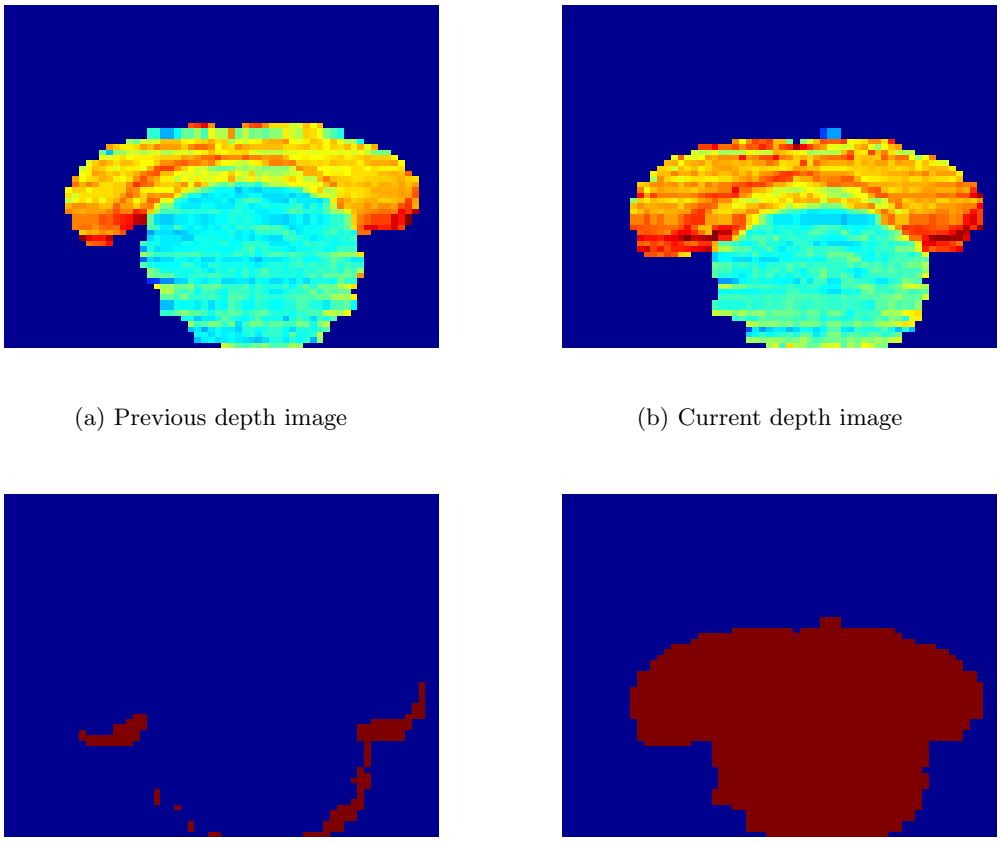
Block motion prediction is performed similar to that of Chen *et al.* [8]. Previous label, depth, and color images are used. The color image was originally used at its full resolution of 256x256 pixels; however, the computations required were found to be prohibitive. Thus, the color image is down-sampled to be the same size as the depth image, 64x64 pixels.

The previous label image is broken up into blocks of 8x8 pixels. A region label is assigned to each block by the majority label of the pixels inside the block. For each block in the previous image, a search window of 17x13 pixels is created in the new image. The block is shifted through every possible position in the search window and the best shift is found. The best shift is defined as the one the minimizes the shift error, $E(m, n)$, which is defined as

$$E(m,n) = \sum_{t=1}^{4} \sum_{i=1}^{8} \sum_{j=1}^{8} |I(i+m, j+n, t) - I_p(i,j,t)|, \tag{2.2}$$

where I is an image formed by stacking the depth image, D_i, on top of the color image, C. The image, I, is then masked by L_E. The image, I_p, is formed in the same manner using the previous depth, color images, and region label images, D_p, C_p, and R_p. The variable t indexes the layer of the stacked image.

The end result is that each region that was present in the previous frame is mapped



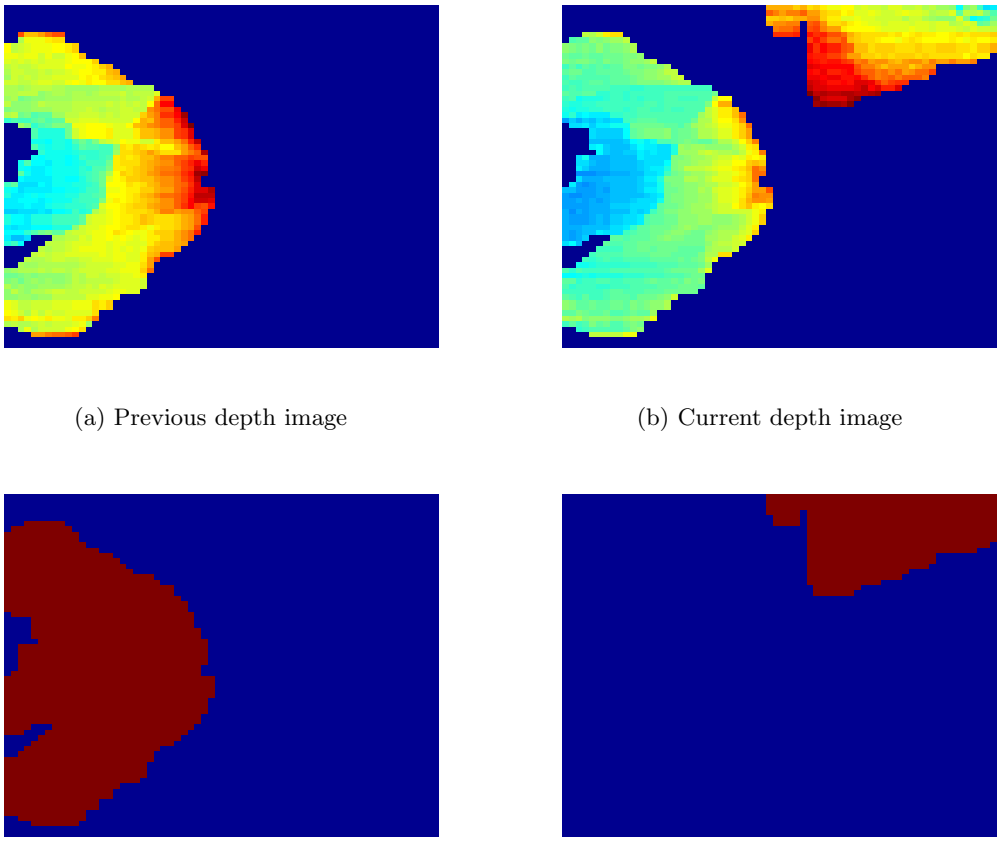
(a) Previous depth image

(b) Current depth image

(c) Previous frame subtraction

(d) Existing person label image

Fig. 2.6: Previous frame subtraction: one person.



(a) Previous depth image

(b) Current depth image

(c) Existing person label image

(d) New person label image

Fig. 2.7: Previous frame subtraction: two people.

to some location in the current frame. Figure 2.8 illustrates the idea of the block motion search.

It is important to note that head and shoulders are tracked separately. Not only are the head and shoulders different in height and color, they also move differently. Separating the two makes it much easier to track multiple people when they run into each other. It is also necessary for them to separated for the matching algorithms described in Chapter 3.

There may be pixels in the current frame that are not assigned to any region during this process. In order to assign the leftover pixels, the average depth and color values are found for each region. Each pixel that was not previously assigned to a region is now assigned to the region that it is closest to in depth and color.

Figure 2.9 shows example input depth images and the output label image. The image also shows vectors in the direction of motion. It can be seen that the person on the right is moving toward the bottom of the image, while the person on the left is standing still.

2.6 Fine Segment

Fine Segmentation is a process that cleans up some of the artifacts from motion segmentation. If more motion occurs than can be predicted by the search window, some regions of an image may have blotches that are in error. This may be an artifact of the low capture rate used. This operation may not be necessary in the next generation texel camera.

Figure 2.10 (a) shows an example image. The edge of the red head region was assigned

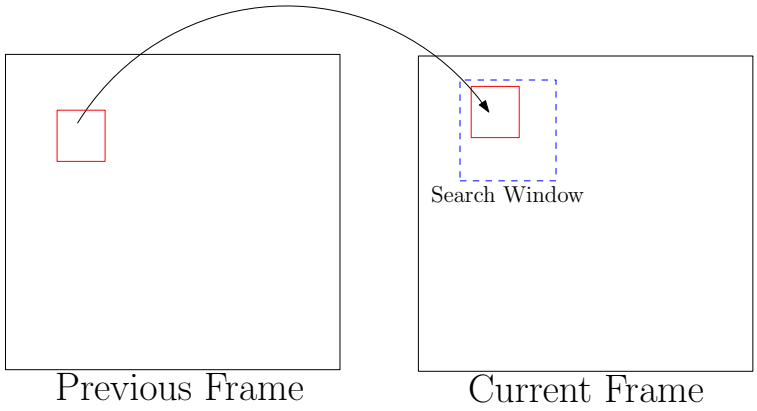
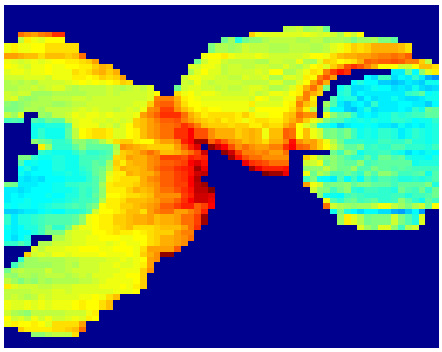
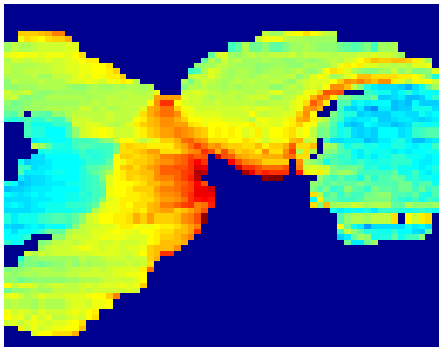


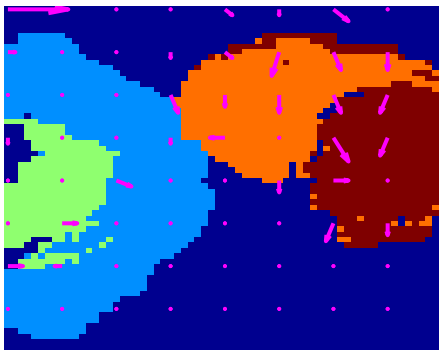
Fig. 2.8: Block motion prediction.



(a) Previous depth image



(b) Current depth image



(c) Output label image

Fig. 2.9: Motion prediction example.

to the orange shoulder region. There is also one pixel near the middle of the head that was assigned to the orange shoulder region.

The algorithm uses the average depth and color values for each region in R_E. Each pixel in the label image is assigned to the region that it is closest to in depth and color.

There may still be some pixels that are not correctly associated. For example, in fig. 2.10 (a) the small orange subregions in the center of the head and on the lower edge of the head are not changed by this process. The depth values of these pixels are such that they appear to be part of the shoulders. The algorithm removes any subregion that falls below a size threshold. This produces the finely segmented region label image, R_f. Figure 2.10 (b) shows the effect after processing. Three orange subregions were removed. The subregion that was on the right hand side of the image did correspond to shoulder pixels. However, the size of the subregion was too small, and it was removed. Some good data is lost, but bad data is removed as well.

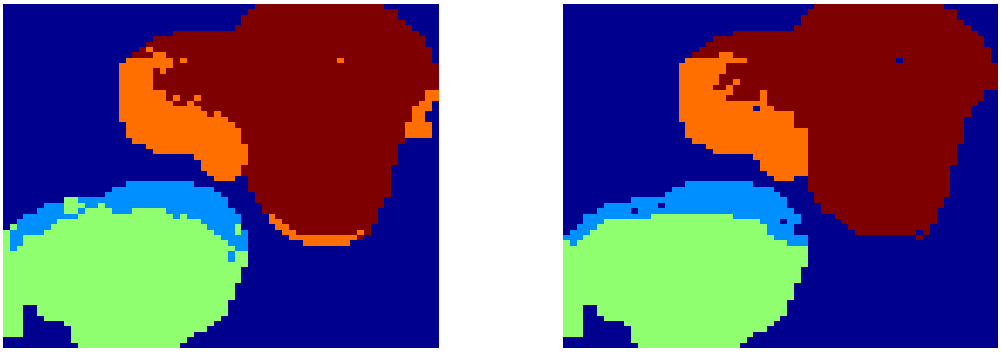
2.7 Cluster

Clustering is performed for two reasons. The first is to segment people when multiple people enter at the same time. The second is to segment the head and shoulders of the people who enter a scene. As explained in sec. 2.5, the head and shoulders are tracked separately.

There is no previous motion information for L_N that can be used to segment people when they first enter a scene, thus clustering techniques are employed. The algorithm used is k-means clustering. It is outlined in Algorithm 2.2 [24, pp. 695-700]. The algorithm divides the data into k regions of minimum variance.

The input images are turned into vectors with each one representing a different pixel location. Each vector contains the depth, color, and image coordinates of its respective pixel. The image coordinates are included so that the clustered regions are more likely to maintain spatial connectivity.

One common issue with k-means clustering is that the number of clusters must be known a priori. A common approach is start with $k = 2$ and increment k until a good



(a) Motion prediction output (b) Fine segmentation output

Fig. 2.10: Fine segmentation example.

Algorithm 2.2 K-Means Clustering

Input:

Set of Input Vectors, $\mathbf{v_i}$,
Number of Regions, k

Output:

Label Vector, $\mathbf{l}$

Begin

1. Choose k centroid locations
2. Assign each v_i to the region with the nearest centroid
3. Compute updated centroid location for each region
4. If termination condition is not met, return to (2)

End

solution is found [25, pp. 461-466]. A size threshold is used just as in other parts of the algorithm. Thus k is increased until a region falls below the size threshold. When k is too large, a person's shoulders may be split into two regions, not because the variance inside the region is large, but because the algorithm dictates that there must be k regions.

The output of this clustering is region label image, R_N. The label image, L_N, divides the image into groups by pixel connectivity. The region label image, R_N, divides the image into different regions based upon height, color, and connectivity. The different regions correspond to the heads and shoulders of the people in an image.

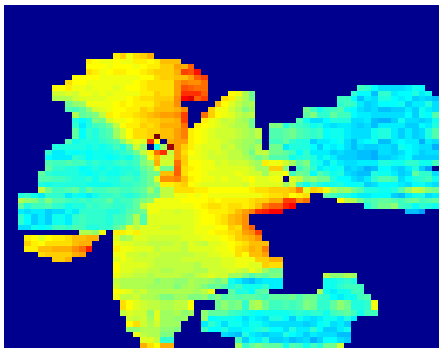
Figure 2.11 gives an example. There are three regions corresponding to the heads of the different people. In the center of the image, there are three shoulder regions that were separated. The depth image shows that all of the shoulder regions have similar depth values. The color of each region is different enough for the clustering to separate the shoulders.

2.8 Find Connections

All of the regions in an image are tracked separately. Which is to say that a person's head and shoulders are not associated with each other until a person exits. This allows the algorithm time to correct association errors that could occur. For example, two people could start off very close together, and their head regions may be confused. As they move throughout the image, it may become clear which head region belongs to which person. It is also common for a person's shoulders to enter the scene before the head. It is not possible to make a correct association until both regions have entered the camera's field of view.

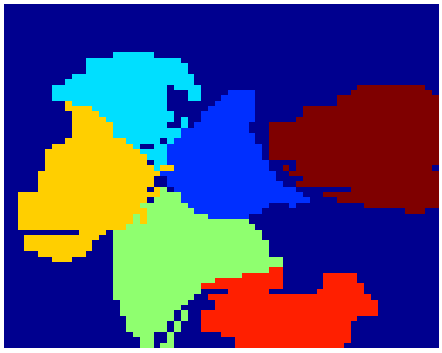
Height and connectivity are used to associate head to shoulders. In each frame, the average height is found for each region and the number of adjacent pixels is found for each pair of regions. In order for two regions to be considered a possible head and shoulder pair, the regions must be connected. The height difference between the regions also must fall within a reasonable range.

In sec. 1.3, anthropometric data was shown for the ratio of total height to head to shoulder distance. Figure 2.12 reproduces this information with the data normalized by the mean value of the ratio. This produces a set of data with a mean value of 1.



(a) Input depth image

(b) Input color image



(c) Clustered output label image

Fig. 2.11: Clustering example.

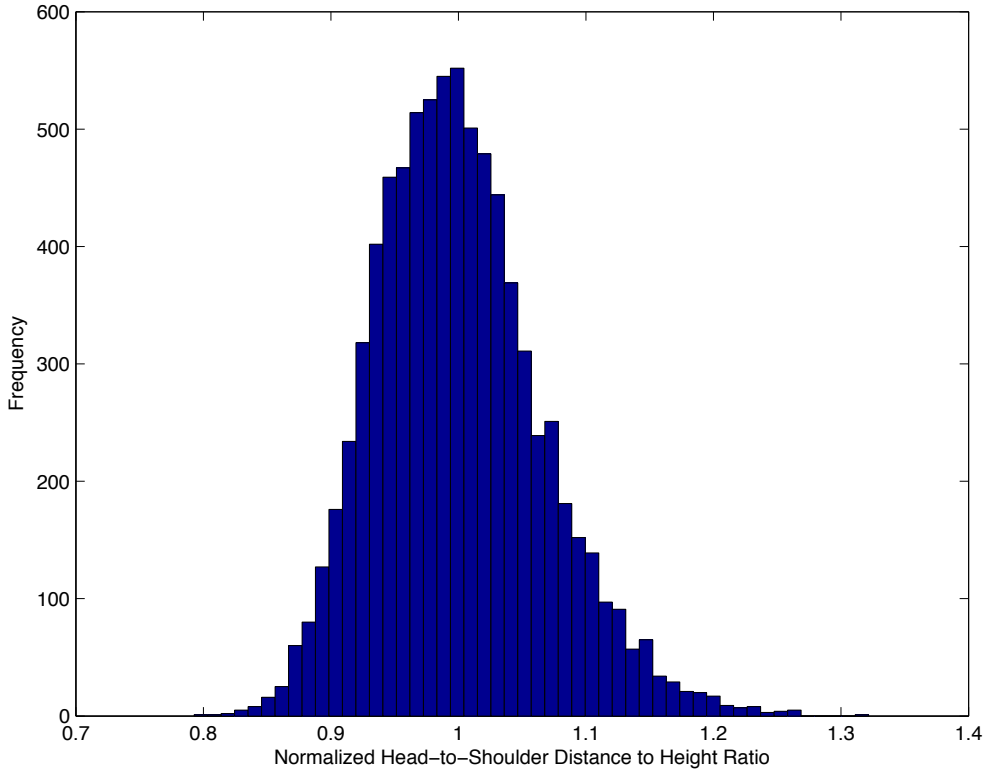


Fig. 2.12: Normalized histogram of head-to-shoulder distance to height ratio.

For each pair of regions in R, this normalized ratio, x, is found. A ad hoc fuzzy probability, $p(x)$, is assigned to each pair according to the mapping of fig. 2.13. The function is wider on the right side because the histogram spreads out further on the right hand side. The values on the x-axis, in which the value of $p(x)$ is 1, correspond to almost all of the density of the histogram. The cutoff points of $x = 0.5$ and $x = 2$ were found through experimentation.

This part of the algorithm produces an upper-triangular connectivity matrix, Con, where the (i, j) entry contains the fuzzy probability that regions i and j are connected.

2.9 Update Hypothesis

Possible head-to-shoulder associations are made for each frame and stored in Con. A hypothesis is also created for each pair of regions. The hypotheses are stored in an upper-triangular matrix, H. They are found by combining the information in Con with information from the previous frames. This information is thresholded to produce the final hypothesis: any value greater than 0.5 is considered a head-to-shoulder match.

Originally, a weighted average combination was postulated. The weight of the new information would be determined by the number of frames the regions had been present in the image. For example, if a region had been in the field of view for nine frames, then the weight corresponding to the previous hypothesis would be 0.9 and the weight corresponding

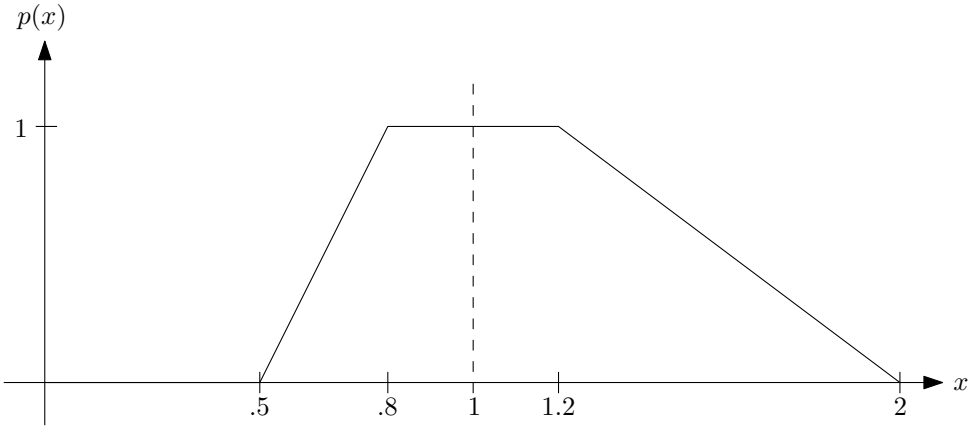


Fig. 2.13: Fuzzy probability mapping.

to the entry of Con would be 0.1. The problem with this is that the value is largely determined by the first few frames. The weight of each successive update becomes smaller and smaller. If an incorrect association is made, it becomes increasingly difficult to change the hypothesis with each frame.

An ad hoc method is proposed that is based on a few observations from the test data.

- Values of Con that are close to 0.5 cause most of the errors.
- The predicted associations are usually correct.
- It is common to have one or two frames for each person where an incorrect association occurs.
- If the original guess for two regions is in error, there may be any number of frames before information needed to make a correct association is available.

A threshold, t, is used to classify each entry in Con as good or poor. Any association with a value within the threshold of 0 or 1 is considered good. Any other value is considered poor.

The threshold was chosen to be $t = 0.2$. That is to say that a good match would have a fuzzy probability between $[0, 0.2]$ or $[0.8, 1]$. Two upper-triangular matrices, N_0 and N_1, store the number of good 0's and good 1's for each pair of regions. These matrices are updated using their previous values, N_{0p} and N_{1p}, and the thresholded values of Con.

The predicted associations either support or refute the previous hypothesis, which is stored in H_p. If the association supports the hypothesis, then the update for the (i, j) entry is the mean of $Con(i, j)$ and $H_p(i, j)$. If it refutes the hypothesis, $H_p(i, j)$ is decremented (or incremented) toward 0.5. The amount of the decrement (or increment) is determined by the number of good 1's and 0's. If the current hypothesis states that the two regions are connected, then only the number of good 0's is used. The decrement is a fixed number, chosen to be $\delta = 0.1$, times the number of good 0's.

Lastly, When an updated hypothesis changes from a 1 to a 0 the number of good 1's is set to 0 and vice versa. This prevents the decrement or increment from becoming too large.

The process is summarized as Algorithm 2.3.

Association errors are relatively rare. The method performs well and is able to correct errors on the data collected. One of the reasons that a method such as this is necessary is because of the low capture rate of the system. It is possible that a less ad hoc algorithm could be used, if more frames were available per person.

2.10 Count

Counting is a trivial task once all of the other steps of the algorithm have been performed. When a region leaves the camera's field of view, hard decisions are made concerning region associations. The system waits until all of the regions associated with a person have left the scene to proceed. Once all of the regions are no longer in the field of view a count is made.

The direction of travel determines whether the count of people on the bus is incremented on decremented. In the experiments one direction was arbitrarily chosen to be the "on" direction.

2.11 Performance

The counting algorithm was tested in several situations. The tests are grouped by the number of people in the field of view. There were 140 tests done with one person in the camera's field of view and 50 tests with multiple people. Situations were tested that are not easily counted by existing counters, such as collisions of people and multiple people entering at the same time. As described in sec. 2.1, people walked through the camera slowly in order to simulate a faster camera and to reduce adverse motion effects.

An error is considered to be any miscount of people. This could be when a person enters or exits but is not counted, or when a person is double counted. It is possible for errors to cancel each other out in the final count; however, in these tests each person was counted individually and the errors were totaled for each person.

The counter achieved 100% accuracy in the cases with only one person in the field of view. It acheived 88% accuracy with multiple people in the field of view. There was

Algorithm 2.3 Update Hypotheses

Input:

Connectivity Matrix, Con,
Matrix of Previous Number of Good 0's, N_{0p}
Matrix of Previous Number of Good 1's, N_{1p}
Matrix of Previous Hypotheses, H_p
Threshold for Selecting Good 0's and 1's, t
Decrement Step Size, δ

Output:

Updated Hypotheses, H

Begin

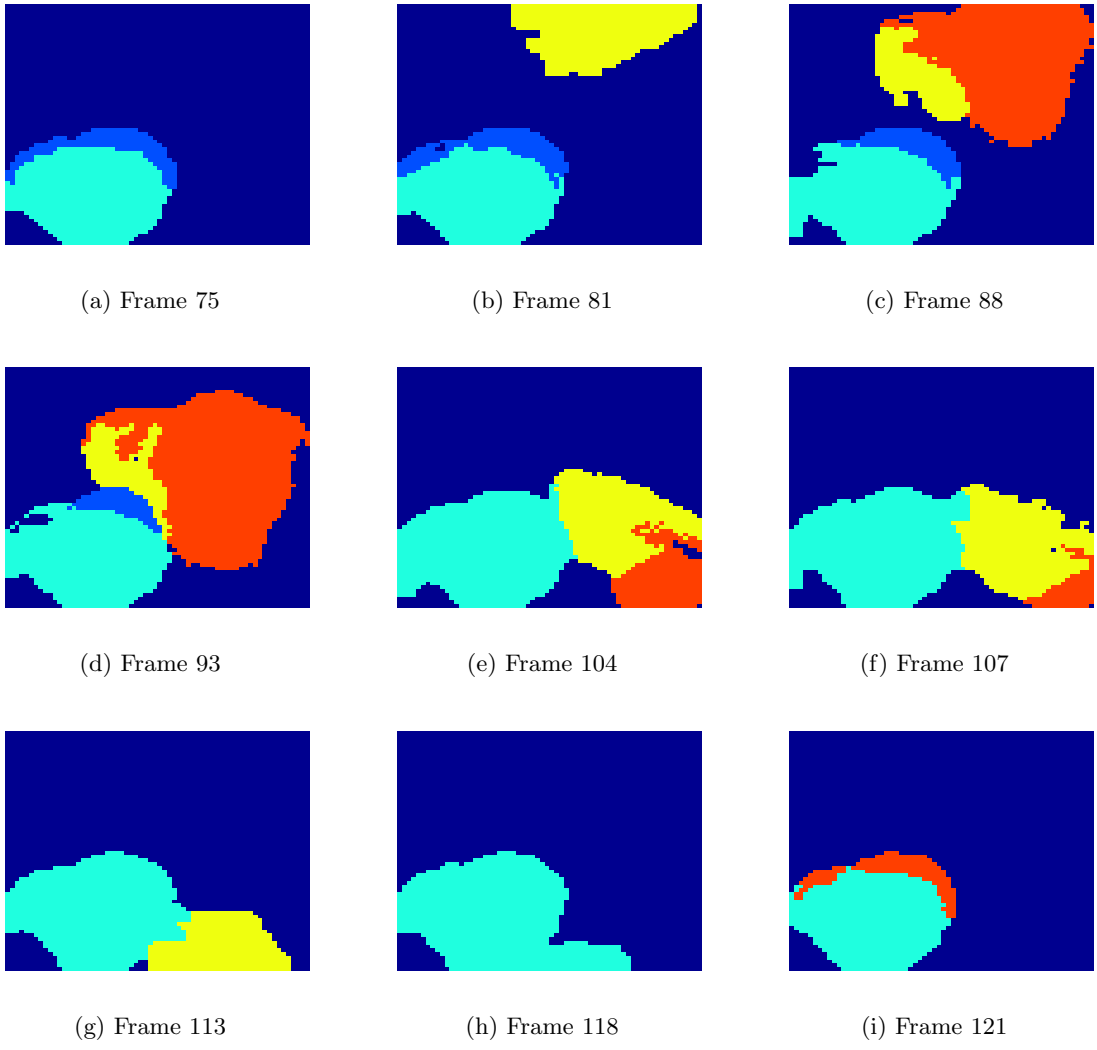
For Each valid (i, j)
If $Con(i, j) < t$ /* A Good 0 */
$N_0(i, j) = N_{0p}(i, j) + 1$
Else If $Con(i, j) < 1 - t$ /* A Good 1 */
$N_1(i, j) = N_{1p}(i, j) + 1$
Else
$N_0(i, j) = N_{0p}(i, j)$
$N_1(i, j) = N_{1p}(i, j)$
End

If ($Con(i, j) > .5$ and $H_p(i, j) > .5$) or ($Con(i, j) < .5$ and $H_p(i, j) < .5$)
$H(i, j) = \frac{1}{2}(Con(i, j) + H_p(i, j))$
Else If $Con(i, j) < .5$ and $H_p(i, j) > .5$
$dec = \delta N_0(i, j)$
$H(i, j) = H_p(i, j) - dec$
If $H(i, j) < .5$
$N_1(i, j) = 0$ /* Reset Good 1's */
End
Else
$inc = \delta N_1(i, j)$
$H(i, j) = H_p(i, j) + inc$
If $H(i, j) > .5$
$N_0(i, j) = 0$ /* Reset Good 0's */
End
End
End
End

one missed count and five double counts. The double counts primarily occured when two people bumped into each other. The person who was bumped may jerk significantly, which produces significant motion artifacts. Most of the double counts could be easily corrected in a new camera that handles motion better.

Figure 2.14 gives an example of a scene with two people. One person is standing still while another is passing through the scene. A scene such as this oftentimes occurs on a full bus.

The stationary person in the lower left part of fig. 2.14 (a) appears to have a head and shoulder shown. Figure 2.15 shows the color image corresponding to Frame 75. The image is actually of a person's shoulder. There are some returns from the person's arm and chest. The blue region corresponds to these pixels. In Frame 104 that region vanishes. The person passing through the scene is covering up some of the arm and chest. The region becomes too small and is removed from further consideration. In Frame 118, the region corresponding to the exiting person becomes too small and is removed, even though there are still a few pixels in the scene that correspond to the leaving person. At this point the counter is incremented. By Frame 121, the person has completely left the field of view, and the arm and chest region becomes large enough to be considered a separate region again. The arm region is colored the same as the head of the previous person. The region labels associated with a person can be reused as soon as that person exits. This reuse of region labels does not correspond to a problem in the algorithm.



(a) Frame 75 (b) Frame 81 (c) Frame 88

(d) Frame 93 (e) Frame 104 (f) Frame 107

(g) Frame 113 (h) Frame 118 (i) Frame 121

Fig. 2.14: Counting example.

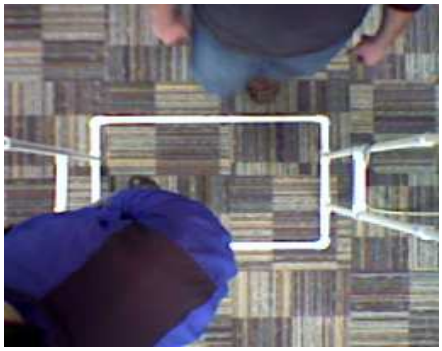


Fig. 2.15: Frame 75 color image.

Chapter 3

Matching People

This chapter explores the matching algorithms used. Sections 3.1 and 3.2 cover the features that are tracked for each person. Sections 3.3 and 3.4 describe possible techniques that can be used to try to match a specific person. Section 3.5 discusses sequence estimation. Several techniques are developed and analyzed.

3.1 Feature Generation and Co-occurrence Matrices

In order to correctly associate people, identifying information needs to be collected. Features are collected for every region in a frame. Several features easily come to mind, such as height and color, as was described in Chapter 2. It is possible to come up with dozens of features to describe each person. The natural questions to be asked are: "What features should be used?" And "How can one tell which features are best?" This section, as well as sec. 3.2, answer these two questions.

Each person is divided into head and shoulder regions in each frame. This provides a natural division of features. All of the features that are calculated for a head are also calculated for the shoulders. A feature vector is created by stacking all of the features associated with a person into a vector. In the following discussion, features will be referred to as belonging to a region. This region could be either a head or a shoulder. When a person exits, regions are associated as that person's head and shoulder. At that time the distinction will be made and the feature vectors created, using data from both the head and shoulder regions.

The features collected can be broken down into three categories: depth features, color features, and texture features. The depth features used are head and shoulder height. The color features used are found in the HSI (Hue, Saturation, and Intensity) color system.

The reason for this is that the intensity of a scene can vary greatly from when a person enters to when they exit. For example, a person may enter a bus in direct sunlight and then leave in a shadow. The hue and saturation should not change significantly if the lighting changes. However, the intensity values will vary greatly. Thus, only hue and saturation are tracked for each region. The height and color values stored as features are the respective mean values for the region.

The last set of features deal with image texture. Texture is only collected on the color image. The depth measurements from a person should be approximately smooth with a small curvature. Height texture does not make sense because a region's height should be smooth. Thus, a measurement of texture in the depth image would be a better characterization of the measurement noise in the LIDAR than height variations from person to person. Color textures, on the other hand, can vary significantly between people. A person wearing a hat should have significantly different texture in the head region than a person with wavy brown hair.

There exists several different measurements of texture in an image. Tuceryan and Jain present several possible measurements [26]. Table 3.1 summarizes the texture features explained. Each of the texture features are found on both hue and saturation. There are six in total which provide for 12 features using the two color channels. When this is added to the original height and color features it brings the total number of features to 15 per region, or 30 per person.

It is important to note that all of these features, except for the standard deviation, are found using a co-occurrence matrix, P_d, and not on the original image.

A co-occurrence matrix contains pixel frequency counts, like a one-dimensional histogram, but it also contains spatial information. A co-occurrence matrix tells how many times a certain pixel combination occurs at a given separation. The (i, j) output value of the co-occurrence matrix is found by summing the number of times that a pixel with value j is a certain distance away from a pixel with value i, at a given orientation.

An example 4x4 matrix is

Table 3.1: Summary of texture-based features.

Feature	**Formula**
Standard Deviation	$(\frac{1}{N-1}\sum_{(i,j)\in R_k}(I(i,j)-\hat{\mu}_k)^2)^{\frac{1}{2}}$
Energy	$\sum_i \sum_j (P_d(i,j))^2$
Entropy	$-\sum_i \sum_j P_d(i,j)\log(P_d(i,j))$
Contrast	$\sum_i \sum_j (i-j)^2 P_d(i,j)$
Homogeneity	$\sum_i \sum_j \frac{P_d(i,j)}{1+\|i-j\|}$
Correlation	$\frac{1}{\sigma_x\sigma_y}\sum_i \sum_j (i-\mu_x)(j-\mu_y)P_d(i,j)$

$$I = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 3 & 1 & 2 \\ 1 & 1 & 2 & 2 \\ 3 & 1 & 2 & 0 \end{bmatrix}. \tag{3.1}$$

The separation and orientation of pixels to be considered must be selected before calculating the co-occurrence matrix. In this example, the separation is one pixel and the orientation is to the right. The output co-occurrence matrix is

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 4 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}. \tag{3.2}$$

The (i,j) entry of G is interpreted as, there are $G(i,j)$ pixels in I that have a pixel value of i, and whose right neighbor has a pixel value of j. For example, $G(2,0) = 1$. This value can be easily verified by checking for the (i,j) values in I, $I(4,3) = 2$ and $I(4,4) = 0$. This is only occurrence of this set of pixels in the image, with this orientation. ($I(3,4) = 2$ and $I(4,4) = 0$, although these correspond to the correct values of (i,j), the second pixel is below the first, not to the right.) A different co-occurrence matrix could be found by

changing either the separation to more than one pixel or the orientation. It is also important to note that wrap around does not occur. In this example, $I(1,4)$ does not have a right neighbor; thus, no value can be added to the co-occurrence matrix at this point.

The co-occurrence matrix provides a lot of information about texture. Smooth images will have a nearly diagonal co-occurrence matrix, while highly textured images will have the co-occurrence values spread out. The texture measures in Table 3.1 all provide a different method of quantifying how spread out this matrix is.

In the example given, only pixel values between 0 and 3 were considered. In an image with 256 gray levels, a 256x256 co-occurrence matrix would be produced. Oftentimes, small variations in pixel values are not of great importance. If this is the case, then the image can be quantized to 8 or 16 gray levels and a much smaller matrix will be produced. The co-occurrence matrices used to find the texture related features in this thesis are produced from input images quantized to 8 gray levels.

3.2 Feature Selection

A large number of features are collected for each person. The task now is to decide which of these features are best for classification purposes. This can be difficult to do because the features collected may be correlated with each other. Two features may not be able to classify well by themselves, but together, they may achieve very good classification. On the other hand, a feature may be able to classify well, as long as it is not paired with a certain other feature. Some features may have no effect on classification. It is desirable to only calculate and use the best features for classification.

The optimal solution is to try every possible set of features with a given classification technique, and choose the one that classifies correctly most often. With every set of features a simulation must be performed, which can take a significant amount of time. In most cases the number of combinations and the time required is far to large to perform an exhaustive search.

A simple suboptimal approach the effect of adding one feature at a time [27, pp. 157-162]. Start with an initially empty set of features to consider. Analyze each feature, one

at a time, by performing a simulation. Choose to keep the feature that is most accurate in the simulation. Repeat the process, analyzing the best feature from before with one other feature added. The process is repeated either a set number of times or until accuracy ceases to improve from adding another feature. This is known as sequential forward selection.

A similar sequential backward selection algorithm exits. The process starts by considering all of the available features at the same time. One feature is removed, and a simulation is performed. This is repeated on all features, and the worst feature is removed from consideration. The process is repeated until either of the termination conditions described above is met.

Both of the previous algorithms can produce good results, but they can easily get stuck in a local minimum. Once a feature is added (or removed) it may not be later removed from (or added back into) the set of features. There exits a forward-backward algorithm that alleviates this problem. The algorithm is summarized as Algorithm 3.1. The algorithm is not guaranteed to find a global minimum, but, in general, it is able to produce a better result than either the forward or backward algorithms by themselves.

Several classification techniques are described in sec. 3.3. The forward-backward algorithm was performed on each of the different techniques. A different set of best features was found for each classification technique. Linear Discriminate Analysis (LDA) performed the best of all of the techniques tested. Table 3.2 shows the features that were selected using the forward-backward algorithm for LDA. There are nine features that are collected for each region, or 18 features per person. Note that the same features are used for both head and shoulders.

3.3 Classification Techniques

The classification performed falls into the category of supervised learning. When a person enters a bus, a set of feature vectors is collected. These features are used to make a classifier for the person. When a person leaves a bus, another set of feature vectors is collected. This set is compared to all of the classifiers that currently exist and the best match is found. This section explores different criteria for finding a best match.

Algorithm 3.1 Forward-Backward Feature Selection.

Input:
Vector of Features Indices, $\mathbf{f}$,
Output:
Vector of Best Features Indices , $\mathbf{f_b}$
Begin
Initialize $\mathbf{f_b} = \{\emptyset\}$, classPercent[] $= \{0, 0, \cdots\}$
For $i = 1$ to numFeatures (1)
If $i \notin \mathbf{f_b}$
Add i to $\mathbf{f_b}$ /* Test addition of each feature */
classPercent[i] = RunSimulation($\mathbf{f_b}$)
Remove i from $\mathbf{f_b}$
Else
Remove i from $\mathbf{f_b}$ /* Test removal of each feature */
classPercent[i] = RunSimulation($\mathbf{f_b}$)
Add i to $\mathbf{f_b}$
End
End

$j = \arg\max(\text{classPercent}[i])$
If $j \notin \mathbf{f_b}$
Add j to $\mathbf{f_b}$ /* Add best feature */
Else
Remove j from $\mathbf{f_b}$ /* Remove best feature */
End
If Termination Condition is met
Exit
Else
Goto (1)
End
End

Table 3.2: List of features used for classification.

Height	
Hue	Saturation
Hue Standard Dev.	Saturation Standard Dev.
Hue Contrast	Saturation Contrast
Hue Homogeneity	Saturation Homogeneity

A wide variety of classification techniques exists. This section explores three: Linear Discriminate Analysis (LDA), Quadratic Discriminate Analysis (QDA), and K-Nearest Neighbor (KNN).

The problem to be explored is different than those usually explored using these techniques. Normally, one measurement is classified at a time. In this particular case, several measurements are collected as a person walks underneath the camera, and then the measurements are combined together. The aforementioned classification techniques have been extended to account for classification using multiple measurements known to be from the same class.

3.3.1 Linear Discriminate Analysis

In LDA, it is assumed that all measurements are independent draws from a set of Gaussian distributions with different means, $\boldsymbol{\mu}_i$. In the context of the problem at hand, each class corresponds to a distinct individual on a bus. It is further assumed that all of the Gaussian distributions have the same covariance matrix, R. When only one measurement is considered, the problem becomes that of estimating

$$\hat{k} = \arg\max_k P(G = k|\mathbf{X} = \mathbf{x}), \tag{3.3}$$

where k is the class considered and $\mathbf{x}$ is a measurement. For this thesis, the measurements are the features vectors of the exiting people.

Using Bayes Rule,

$$P(G = k|\mathbf{X} = \mathbf{x}) = \frac{P(\mathbf{X} = \mathbf{x}|G = k)P(G = k)}{P(\mathbf{X} = \mathbf{x})}, \tag{3.4}$$

where

$$P(\mathbf{X}=\mathbf{x}|G=k)=\frac{1}{C\cdot|\hat{R}|^{\frac{1}{2}}}e^{-\frac{1}{2}(\mathbf{x}_i-\hat{\boldsymbol{\mu}}_k)\hat{R}^{-1}(\mathbf{x}_i-\hat{\boldsymbol{\mu}}_k)^T}, \tag{3.5}$$

$$\hat{\boldsymbol{\mu}}_k=\frac{1}{V_k}\sum_{g_i=k}\mathbf{x}_i, \tag{3.6}$$

$$\hat{R}=\frac{1}{V-K}\sum_{k=1}^{K}\sum_{g_i=k}(\mathbf{x}_i-\hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i-\hat{\boldsymbol{\mu}}_k)^T, \tag{3.7}$$

and

$$P(G=k)=\hat{\pi}_k=\frac{V_k}{V}. \tag{3.8}$$

In the previous equations, V is the total number of vectors in the training sequence, V_k is the total number in the sequence corresponding to class k, K is the total number of classes, and $\sum_{g_i=k}$ means the sum on all of the training vectors from class k.

The probability $P(\mathbf{X}=\mathbf{x})$ may be ignored because it is common to all classes and, thus, has no effect on the maximization problem. In order to reduce the complexity of the calculations, $\log(P(\mathbf{X}=\mathbf{x}|G=k)P(G=k))$ is evaluated. If terms common to all k are ignored,

$$\delta_k^l(\mathbf{x})=log\hat{\pi}_k-\frac{1}{2}(\mathbf{x}-\hat{\boldsymbol{\mu}}_k)^T\hat{R}^{-1}(\mathbf{x}-\hat{\boldsymbol{\mu}}_k). \tag{3.9}$$

If all of the classes are equally likely, then the $\log\hat{\pi}_k$ term may also be ignored. The $-\frac{1}{2}$ scale factor can be removed, which changes the problem from that of maximization to minimization. This gives as a final result

$$\delta_k^l(\mathbf{x})=(\mathbf{x}-\hat{\boldsymbol{\mu}}_k)^T\hat{R}^{-1}(\mathbf{x}-\hat{\boldsymbol{\mu}}_k), \tag{3.10}$$

which is the Mahalanobis distance between $\mathbf{x}$ and $\hat{\boldsymbol{\mu}}_k$. In other words, the most likely class is the one for which this Mahalanobis distance is minimized.

3.3.2 Linear Discriminate Analysis with Multiple Measurements

There are several feature vectors collected as a person exits a bus, one from each frame the person is present in. A better classification can be performed by using all of these measurements for classification.

LDA can easily be extended to exploit multiple measurements. In the case of $n = 2$ measurements, repeated application of Bayes Rule yields

$$P(G = k|\mathbf{X}_1 = \mathbf{x}_1, \mathbf{X}_2 = \mathbf{x}_2) = \frac{P(\mathbf{X}_1 = \mathbf{x}_1|G = k, \mathbf{X}_2 = \mathbf{x}_2)P(G = k|\mathbf{X}_2 = \mathbf{x}_2)}{P(\mathbf{X}_1 = \mathbf{x}_1|\mathbf{X}_2 = \mathbf{x}_2)}, \tag{3.11}$$

$$= \frac{P(\mathbf{X}_1 = \mathbf{x}_1|G = k, \mathbf{X}_2 = \mathbf{x}_2)P(\mathbf{X}_2 = \mathbf{x}_2|G = k)P(G = k)}{P(\mathbf{X}_1 = \mathbf{x}_1|\mathbf{X}_2 = \mathbf{x}_2)P(\mathbf{X}_2 = \mathbf{x}_2)}. \tag{3.12}$$

If the measurements are independent, this simplifies to

$$P(G = k|\mathbf{X}_1 = \mathbf{x}_1, \mathbf{X}_2 = \mathbf{x}_2) = \frac{P(\mathbf{X}_1 = \mathbf{x}_1|G = k)P(\mathbf{X}_2 = \mathbf{x}_2|G = k)P(G = k)}{P(\mathbf{X}_1 = \mathbf{x}_1)P(\mathbf{X}_2 = \mathbf{x}_2)}. \tag{3.13}$$

This process can be extended to m independent measurements as

$$P(G = k|\mathbf{X}_1 = \mathbf{x}_1, \cdots, \mathbf{X}_m = \mathbf{x}_m) = P(G = k)\prod_{i=1}^{m}\frac{P(\mathbf{X}_i = \mathbf{x}_i|G = k)}{P(\mathbf{X}_i = \mathbf{x}_i)}. \tag{3.14}$$

As in LDA with one measurement, computation can be simplified by maximizing the log of (3.14) and removing common terms. This produces

$$\delta_k^l(x) = log\hat{\pi}_k - \frac{1}{2}\sum_{i=1}^{m}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^T\hat{R}^{-1}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k), \tag{3.15}$$

where x is the set of measurements, $\{\mathbf{x}_{1i}, \mathbf{x}_{2i}, \cdots, \mathbf{x}_{mi}\}$.

If all of the classes are equally likely, this can again be simplified as

$$\delta_k^l(x) = \sum_{i=1}^{m}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^T\hat{R}^{-1}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k). \tag{3.16}$$

The most likely class is now the one that minimizes this sum of Mahalanobis distances.

3.3.3 Quadratic Discriminate Analysis

QDA makes the same assumptions as LDA, except that it does not assume that each of the classes have a common covariance matrix. Thus, a covariance matrix needs to be computed and stored for each class as given by

$$\hat{R}_k = \frac{1}{V_k - 1} \sum_{g_i=k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^T. \tag{3.17}$$

This produces similar discriminate functions to those described in sec. 3.3.1. The $\frac{1}{|\hat{R}_k|^{\frac{1}{2}}}$ term from the original distribution can no longer be ignored. In the case of one measurement this becomes

$$\delta_k^q(\mathbf{x}) = log\hat{\pi}_k - \frac{1}{2}\log|\hat{R}_k| - \frac{1}{2}(\mathbf{x} - \hat{\boldsymbol{\mu}}_k)^T \hat{R}_k^{-1}(\mathbf{x} - \hat{\boldsymbol{\mu}}_k). \tag{3.18}$$

When multiple measurements are available the discriminate function becomes

$$\delta_k^q(x) = log\hat{\pi}_k - \frac{m}{2}\log|\hat{R}_k| - \frac{1}{2}\sum_{i=1}^{m}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^T \hat{R}_k^{-1}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k). \tag{3.19}$$

3.3.4 K-Nearest Neighbor

In KNN, feature vectors are classified according to the vectors from the training set that they are closest to. In the case of counting people on a bus, features are stored for each person as they enter the bus. Voronoi, or decision regions, are created for each person. An exiting vector falls into one decision region, and is assigned the same label as that region.

The K in K-Nearest Neighbor is the number of points to be considered in creating the Voronoi regions. If some of the K points belong to different classes, then the region corresponds to the class that the majority of the K points belong to. As an example, assume $K = 5$. At a given point, three of the nearest training vectors may belong to class 1, while the remaining two nearest training vectors belong to class 2. The vector is assigned to class 1 because there were more neighbors in class 1 than in class 2.

In practice, the Voronoi regions do not need to be calculated explicitly. All of the

training vectors are stored. The distance between an exiting vector and all of the training vectors is found. The classes of the vectors corresponding to the K smallest distances are found. The exiting vector is assigned to the majority class of the K nearest neighbors.

KNN is easily extended to multiple measurements. The K nearest neighbors are found for each of the exiting vectors, $\mathbf{x}_i$. For m measurements, $K \cdot m$ neighbors are found. All of the vectors are assigned to the majority class of these $K \cdot m$ nearest neighbors.

3.4 Performance of Classification Techniques

A simple test was performed in order to compare the classification techniques described in the previous section. Data was collected for 70 different people as they entered and exited from a simulated bus environment. The marginal accuracy of a classifier is the percentage of correct associations of people when one person is removed from the bus. It is more difficult to match people correctly if 100 people are on the bus as opposed to 2 or 3. Thus, the marginal accuracy is associated with a given number of people on the bus.

In all of the comparisons, the marginal accuracy was computed with 70 people on the bus. In an actual bus, the direction of travel determines whether a person is entering or exiting. The decision, however, is arbitrary in the simulated bus environment. There are two sets of feature vectors collected for each of the 70 people. For each person, one of those sets is selected at random to be the entering vectors, and the other to be the exiting vectors. The total number of possible selections is

$$P = 2^{70} \left(\begin{array}{c} 70 \\ 1 \end{array} \right) = 2^{70} \cdot 70. \tag{3.20}$$

The number of possibilities, P, is far to large to test every possible combination of people. Monte Carlo analysis was used to estimate the accuracy of each classifier. The experiment of removing one person from the bus was repeated 7,000 times and the number of correct associations was counted. Table 3.3 summarizes the performance of each technique.

Table 3.3: Marginal accuracy of classifiers with 70 classes.

Technique	**Accuracy**
LDA	89.8%
QDA	56.1%
KNN	54.5%

It is assumed that the head and shoulder features are independent. This produces a block diagonal covariance matrix that will be used in LDA and QDA,

$$R = \begin{bmatrix} R_H & 0 \\ 0 & R_S \end{bmatrix}. \tag{3.21}$$

The inverse of R can be found by finding the inverse of each block. This can easily be verified by direct substitution,

$$I = RR^{-1} = \begin{bmatrix} R_H & 0 \\ 0 & R_S \end{bmatrix} \begin{bmatrix} R_H^{-1} & 0 \\ 0 & R_S^{-1} \end{bmatrix} = \begin{bmatrix} R_H R_H^{-1} & 0 \\ 0 & R_S R_S^{-1} \end{bmatrix} = I. \tag{3.22}$$

LDA performed significantly better than the other classifiers. The reason for this is the sparsity of data. In QDA each person has 18 features that are tracked in two 9x9 correlation matrices. There is a minimum of nine measurements needed in order for the matrix to be invertible. In many cases, there are fewer than nine measurements available for a person. A regularizer is used, but it causes significant distortion when there are few measurements. The correlation matrices are not very accurate for some of the people in the set.

In KNN a similar problem a occurs. Due to the sparsity of data, a measurement may be relatively close to the true mean, but close to only a few of the measurements from the correct person to be associated with.

In LDA, the overall covariance matrix is constructed from all of the training measurements. The main cause of variation from the mean value of the features collected is measurement noise. This noise is independent of the person being matched. LDA is able to produce a more stable covariance estimate than QDA by combining all of the data available. LDA will be used for the remainder of the experiments in Chapter 4.

Other classification techniques could have been tested. For example, there are many distribution matching techniques. These would suffer from the same problems as QDA and KNN. The sparsity of the data would limit the accuracy of the such techniques.

3.5 Sequence Estimation

The marginal accuracy is useful, but it does not indicate the accuracy in an actual bus environment. One person exited the bus at a time and an incorrect decision had no effect on any decisions made at a later time. This is not the case for the matching problem to be explored. Decisions can have a great effect on all of the later decisions to be made.

The problem now turns into one of sequence estimation. The difficult question is: "How to incorporate previous information into future decisions." The remainder of this chapter explores a few methods of incorporating this information.

3.5.1 A Simple Technique

A simple approach is to make a hard decision as soon as a person exits the bus. The person associated with the exiting person is removed from further consideration. In this manner, everyone on the bus will eventually be assigned to an exiting person.

The hard decisions are the only information that is incorporated into the later decisions. Any decision reduces the number of people to be considered at a later time. If the decision is correct, this increases the probability of a correct decision at a later time. However, if the decision is incorrect, the error will propagate through to other decisions. One error can cascade into several.

3.5.2 An Ad Hoc Approach

The simple approach of sec. 3.5.1 leaves much room for improvement. The effect of one error can be catastrophic. It may be possible to exploit some of the properties of the data in order to reduce the cascading effects of an error. This section outlines a few heuristics that make use of previous data in order to make decisions.

Ideally, in the sequence estimation problem there would be a one-to-one mapping of exiting people to people who entered the bus. This idea is shown in fig. 3.1. The letters on the left hand side represent people on the bus. The numbers on the right hand side represent the ordering of people exiting the bus. A different set of symbols is used because the order of the exiting people is not known *a priori*. The order in which the people leave is read from top to bottom. Thus, person 1 exits before person 3.

This one-to-one mapping is ideal, but there are situations in which it cannot occur. The counting algorithm does not achieve 100% accuracy. Thus, it is possible to have a person get off who was never counted as being on the the bus and vice versa. If a one-to-one mapping is assumed, the first case leads to a minimum of two classification errors.

As an example, person x exits and is classified as person y. When person y exits, another error must occur. If person y is the last person to leave the bus, then the error will not propagate any further, because there is no one left on the bus for the error to propagate to.

This problem can continue for far more than two errors. Figure 3.2 shows an example in which three errors occur. In this case the right hand side has letters corresponding to order in which the people exited. Person e is on the right hand side but not the left. This

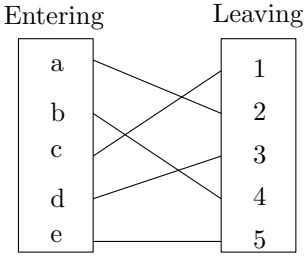


Fig. 3.1: Example one-to-one mapping.

indicates that person e entered but was not counted. The question mark next to person d indicates that there is no one to whom person d can be matched.

Error propagation can also occur when the count of people on the bus is correct. In this case, the minimum number of errors is two. For example, person x exits and is classified as person y. Only two errors occur if person y is later classified as person x. If person y is not classified as person x, more errors will occur. An example is illustrated in fig. 3.3. Person b is mistakenly associated with person e. Person e is associated with person b, thus only two errors occur. If person e had been associated with any other person, more errors would have occurred.

In an attempt to mitigate these problems, a one-to-many mapping of entering people to exiting people is allowed. That is to say that a person is allowed to exit the bus multiple times. If a classification error occurs, it does not necessarily propagate to other decisions; however, it also causes other errors to occur. In an extreme case, a single person could be classified as every exiting person. Only one of these decisions is correct, all of the others constitute errors.

Figure 3.4 shows the best case scenario of the example in fig. 3.2 using a one-to-many mapping. Only one incorrect association occurs in this case, person b to person e.

Figure 3.5 shows the best case scenario of the example in fig. 3.3. Only one classification error occurs; however, person e is left on the bus, which could cause another error to occur at a later point in time.

The ad hoc approach assumes that counting and classification errors will occur. It attempts to mitigate their effects via thresholding. The remainder of this section explains

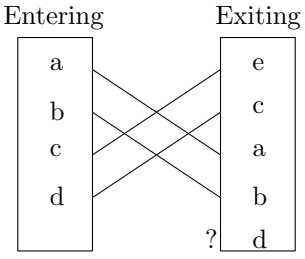


Fig. 3.2: Example with counting errors.

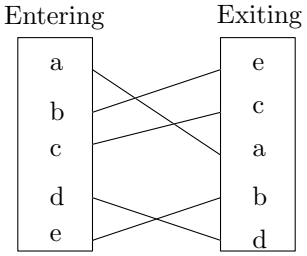


Fig. 3.3: Error with two people.

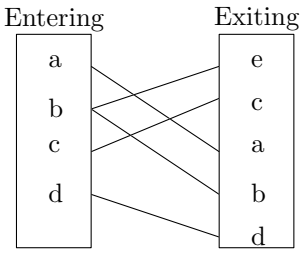


Fig. 3.4: One-to-many mapping example 1.

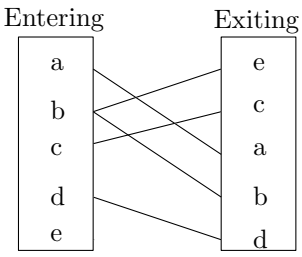


Fig. 3.5: One-to-many mapping example 2.

the three thresholds used. The discriminate function used for classification in this thesis is (3.16). However, any discriminate function may be used.

The value of the discriminate function is oftentimes much smaller for one person than for every other person considered. This approach assumes that it is appropriate in this situation to make a hard decision without any further processing. A difference threshold, t_{d1}, is set and the values of the discriminate function are sorted. If the difference between the smallest and the second smallest values of the discriminate function is above this threshold, a hard decision is made.

There are also situations in which the best match may not be a very good match. For example a person may enter the bus wearing a hat. The hat is removed while the person is on the bus. The features collected for the head, when that person exits, will be very different from the features collected when that person entered. A large measurement error will occur which increases the likelihood of a classification error.

A minimum value threshold, t_m, is used to mitigate this problem. If the minimum value of the discriminate function used is below this threshold, the match is considered good, a hard decision is made, and the person is removed from further consideration. If the minimum value is above the threshold, a hard decision can still be made, but the person will not be removed from further consideration.

The last heuristic deals with situations is which the value of the discriminate function is similar for multiple people, creating an ambiguous decision. Errors often occur in this situation. Another difference threshold, t_{d2}, is added to the minimum discriminate value obtained, producing $t_w = t_{d2} + d_{min}$. All of the people for whom the value of the discriminate function falls below t_w are put into a waiting pool. Decisions are not made until more information is known. Whenever a hard decision is made, the waiting pool is checked. If the person for whom the hard decision was made is in the waiting pool, they are removed. At this point, it may be possible to make further decisions, since the source of the ambiguity may have been resolved. This decision will cascade through the waiting pool until no further decisions can be made.

It may not be possible to eliminate everyone from the waiting pool in this manner. For example, everyone may have exited the bus but there may still be 10 people left in the waiting pool. A simple approach is taken, as shown in Algorithm 3.2.

Algorithm 3.2 Waiting Pool Reduction

Input:
Vector of Possible Entering People, **p**,
Vector of Possible Exiting People, **e**,
Vector of Distances for Each Association **d**,
Output:
Matrix of Associations , A
Begin
Initialize dist1=[],dist2=[]
While p is not empty
For Each Distinct Person in **p**, i
$\mathbf{d}^i = \{k : \mathbf{p}_k = i\}$ /* Add all instances of i in **p** */
Add $\max(\mathbf{d}^i) - \min(\mathbf{d}^i)$ to dist1
End
For Each Distinct Person in **e**, j
$\mathbf{d}^j = \{k : \mathbf{e}_k = j\}$ /* Add all instances of j in **p** */
Add $\max(\mathbf{d}^j) - \min(\mathbf{d}^j)$ to dist2
End
$w1 = \arg\max_i$ dist1 /* Find the max of */
$w2 = \arg\max_j$ dist2 /* dist1 and dist2 */
If max(dist1) > max(dist2)
Remove Association of person $w1$ with largest d from Consideration
Else
Remove Association of person $w2$ with largest d from Consideration
End
Recursively Add any Decisions that can be Made to A
End
End

Each entry in the waiting pool is comprised of a possible entering person, p_i, a possible exiting person, e_i, and a distance, d_i. These three values are known collectively as an edge. All of the edges in the waiting pool can be grouped into three vectors: a vector of entering people, **p**, a vector of exiting people, **e**, and a vector of distances, **d**. A given person may appear several times in **p** or **e**.

When a person exits, there may be ambiguity of multiple people who could have left.

The ambiguity can also be viewed in another light. For each person on the bus, there may be several stops at which that specific person may have exited. The waiting pool reduction algorithm attempts to resolve both ambiguities.

All of the distances associated with a given person, person i, in $\mathbf{p}$ are put in the vector $\mathbf{d}^i$. This represents the distances for all of the stops that person i may have exited at. The difference between the maximum and minimum values of $\mathbf{d}^i$ is computed for each distinct person in $\mathbf{p}$. If the difference is small, a significant ambiguity exists. However, if the difference is large, the ambiguity can be resolved more easily. This difference is stored in the dist1 vector for each distinct person in $\mathbf{p}$.

The same process is repeated for every distinct person in $\mathbf{e}$, and the differences are stored in the dist2 vector. In this case, all of the values in $\mathbf{d}^j$ represent the distances associated with all of the people who may have been the jth person to exit the bus.

Only one edge is removed at each iteration of the algorithm. The edge associated with the maximum value of dist1 and dist2 is removed. At this point it may be possible to make hard decisions. If any exiting person now has only person associated with it, then a hard decision is made. This person is removed from further consideration. All edges in the waiting pool involving this person are removed. It is possible that other hard decisions can be made due to this propagation. These decisions are propagated through until no further decisions can be made.

The algorithm outputs a matrix of associations, A. The matrix has two rows. Each column represents an association of an entering person to an exiting person.

3.5.3 An Optimal Approach

The ad hoc approach uses thresholds to improve the matching accuracy. An improvement does occur, but the approach is not optimal with respect to any criterion. The section explores an optimal method of performing sequence estimation in the maximum likelihood sense.

The probability of a given sequence of n sets of feature vectors is

$$P(X_1 = x_1, X_2 = x_2, \cdots, X_n = x_n | G_1 = k_1, G_2 = k_2, \cdots, G_n = k_n), \tag{3.23}$$

where $(G_i = k_i)$ is the event that the ith person to leave the bus is person k_i, and $(X_i = x_i)$ is the event that the ith observation is x_i. The observation x_i is the set of feature vectors collected as the ith person exits a bus, $x_i = \{\mathbf{x}_{1i}, \mathbf{x}_{2i}, \cdots, \mathbf{x}_{mi}\}$. In sec. 3.3.2 only one subscript was used, but two are now needed. The second subscript indicates the class from which the m measurements came.

In order to simplify equations below, the events $(X_i = x_i)$ and $(G_i = k_i)$ will be referred to as x_i and k_i for the remainder of this discussion.

The problem can now be expressed as

$$\arg\max_{\mathbf{k}} P(x_1, x_2, \cdots, x_n | k_1, k_2, \cdots, k_n), \tag{3.24}$$

where $\mathbf{k}$ is a vector whose entries are $k_1, k_2, \cdots, k_n$.

Each observation, x_i, only depends on the class, k_i, from which it came. That is to say that each x_i is conditional independent given k_i,

$$P(x_1, x_2, \cdots, x_i | k_1, k_2, \cdots, k_n) = \prod_{i=1}^{n} P(x_i | k_i). \tag{3.25}$$

LDA is used for classification, thus each x_i is assumed to be a set of observations from a Gaussian random variable with mean, $\hat{\boldsymbol{\mu}}_{k_i}$, and covariance, $\hat{R}$. The log of (3.25) can be used to simplify computations,

$$\log \prod_{i=1}^{n} P(x_i | k_i) = \sum_{i=1}^{n} \log P(x_i | k_i), \tag{3.26}$$

where

$$\log P(x_i|k_i) = -\log C - \frac{1}{2}\log|\hat{R}| - \frac{1}{2}\sum_{j=1}^{m}(\mathbf{x}_{ji} - \hat{\boldsymbol{\mu}}_{k_i})^T \hat{R}^{-1}(\mathbf{x}_{ji} - \hat{\boldsymbol{\mu}}_{k_i}). \tag{3.27}$$

The first two terms are common for all i. If they are ignored as well as $-\frac{1}{2}$ scale factor, the end result is $\delta^l_{k_i}(x_i)$ from (3.16). The new problem is to find

$$\arg\min_{\mathbf{k}} \sum_{i=1}^{n} \delta^l_{k_i}(x_i). \tag{3.28}$$

An optimal solution would be to evaluate this sum for every possible value of $\mathbf{k}$ and choose the set that minimizes the sum. This would be prohibitive, even for a relatively small number of people on a bus. The number of possible orderings for N people is $N!$.

The Viterbi algorithm reduces the number of orderings to be tested without sacrificing optimality [28]. It uses a trellis to search through all of the possible paths. In order to use a trellis, the notion of state must be defined. In this thesis, the state is defined as the people on the bus.

Figure 3.6 shows an example trellis with four people on a bus. Each black dot represents a different state, which is labeled with a set in curly braces. Each edge in the trellis is labeled with an exiting person. The S_i labels at the bottom of the graph represent the different stops. In this example, one person exits at each stop.

The first state on the left hand side of fig. 3.6 is $\{a, b, c, d\}$, which is to say that all four people are on the bus. One person exits at S_1. There are four possible new states. Each of these are produced by removing one person from the initial state, $\{a, b, c, d\}$. The usefullness of the Viterbi algorithm can be seen at S_2. There are two paths coming into state $\{c, d\}$. These paths correspond to person a leaving, followed by person b and vice versa. Each of these paths has a cost associated with it by (3.28). One of the costs will be smaller than the other.

There are four possible exiting orders that pass through state $\{c, d\}$ at S_2, $[a, b, c, d]$, $[b, a, c, d]$, $[a, b, d, c]$, $[b, a, d, c]$. Only the first two possibilities will be analyzed. That is to

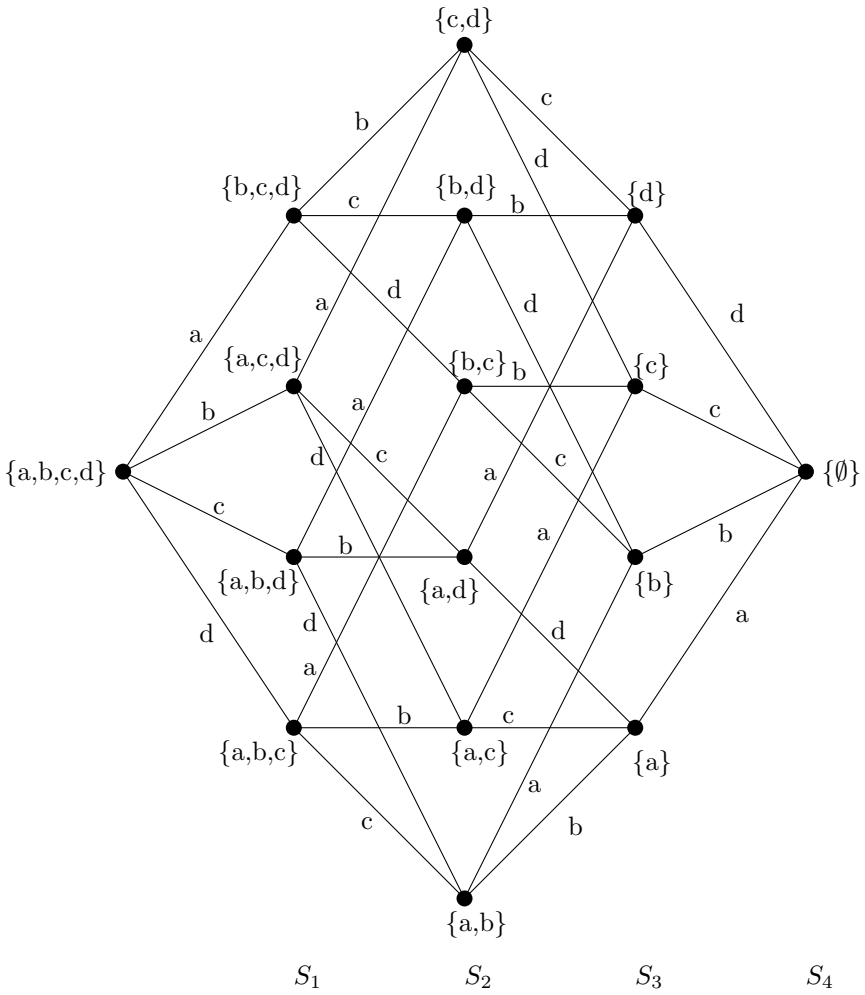


Fig. 3.6: A simple trellis with four people.

say that person c left at S_3 and person d left at S_4. The cost of each part of the sum in (3.28) is independent of who has already left the bus, which in this case produces

$$\sum_{i=1}^{4} \delta^l_{k_i}(x_i) = \delta^l_{k_1}(x_1) + \delta^l_{k_2}(x_2) + \delta^l_c(x_3) + \delta^l_d(x_4), \tag{3.29}$$

where k_1 and k_2 are a and b or vice versa. The last two terms in the sum do not change with the choice of k_1 and k_2. It is only necessary to minimize these first two terms whose value is known at S_2. It is not necessary to wait until S_4 to make a hard decision. This is true for any path that passes through state $\{c, d\}$ at S_2. As an example, if the path cost of $k_1 = a$ and $k_2 = b$ is smaller than that of $k_1 = b$ and $k_2 = a$, then the path corresponding to $k_1 = b$ and $k_2 = a$ can be removed from the trellis.

In terms of the trellis, if multiple paths pass through the same state, the path with a larger cost can be pruned and never considered again. In this manner, the trellis significantly reduces the total number of possible sequences to be explored, without loosing optimality.

The trellis can also be used to accommodate people entering a bus. Figure 3.7 gives an example where there are originally three people on the bus. Two people leave, and then another person enters. There are no labels on the edges corresponding to person d entering the bus. The edge labels only correspond to exiting people.

3.5.4 A Beam Search Approximation

The Viterbi algorithm significantly reduces the number of paths that need to be explored in order to arrive at an optimal solution in the maximum likelihood sense. However, the number of paths can still be prohibitive. The number of states at a given time step depends on the number of people currently on the bus and the number of people that will be considered as a possible match. As a simple example, assume N people are on the bus and one person exits at each stop. The number of states after k stops is

$$\begin{pmatrix} N \\ k \end{pmatrix}. \tag{3.30}$$

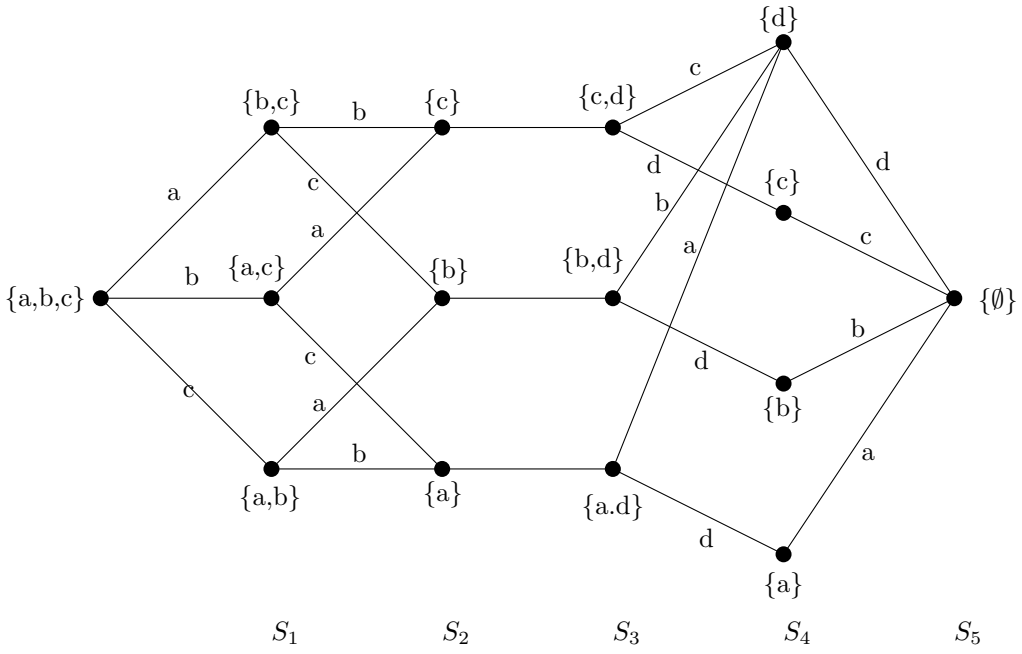


Fig. 3.7: Trellis representing an entering person.

The number of states to be considered can grow quite large, even for relatively small N. It is not feasible to store nor to compute paths for such a large number of states. A common approach used in such a situation is a beam search [19, 20]. A beam width, W, is selected. At each stage of the trellis, only the W best paths are stored. At the next stage, extensions are made from these paths, the path costs are sorted, and then the number of extensions is pruned to the W paths of smallest cost. Optimality is sacrificed in order to decrease memory and computational requirements.

The number of states to be stored at each time step is constant. The number of extensions changes for each time step depending on several factors. It is given by

$$K_{t+1}^{-} = \min\left(K_t^{+} N_t, \begin{pmatrix} N_t^p \\ N_t - 1 \end{pmatrix}, \left[K_t^{+} N_t^e + \begin{pmatrix} N_t^p - N_t^e \\ N_t - N_t^e - 1 \end{pmatrix}\right]\right), \tag{3.31}$$

where K_t^{-} is the number of extensions before pruning at stop t, K_t^{+} is the number of extensions after pruning, N_t is the current count of people on the bus, N_t^p is the current

number of people under consideration (those for whom a hard decision has not been made), and N_t^e is the number of people who enter the bus at stop t.

No entering situations will be explored throughout the remainder of this thesis. Thus, only the first two terms of (3.31) need to be considered.

It is not necessary to make all K_{t+1}^- extensions. At stop t, all of the paths in the trellis are in sorted order. There are N_t people on the bus at this time. For each person, there is a cost associated with the event of that specific person exiting at time t. These costs are also sorted. Using sorted data reduces the number of extensions that need to be stored.

It is not necessary to store the full state at each time step either. The necessary information can be found by only storing the person corresponding to the edge labels as in fig. 3.6. The set of exiting people can be found by back tracing through the trellis.

A buffer, B_t, is created which can store W edge labels. The buffer B_{t-1} can be used to find the W best states from the previous stage of the trellis. A cost vector, C_p, is also used. This vector contains the path costs associated with each of the paths that can be derived from back tracing through $B_{t-1}, B_{t-2}, \cdots, B_1$. The new path costs will be stored in C. The path cost of an extension of person j at time t can be calculated as

$$c_j = C_p[i] + \delta_{k_t}^l(x_t), \tag{3.32}$$

where $C_p[i]$ is the cost of the ith path at time $t-1$, and $\delta_{k_t}^l(x_t)$ is the value of the discriminate function for class k_t at time t.

If $W < K_t^+$, then the W best extensions are made from $B_{t-1}[0]$, where $B_{t-1}[0]$ is the state with minimum path cost from the previous stage of the trellis. If $W > K_t^+$, then all possible extensions of $B_{t-1}[0]$ are made. The trellis requires a one-to-one mapping of entering people to exiting people, a person cannot exit the bus at multiple stops. Any extension of $B_{t-1}[0]$ that would require a person to exit the bus twice, will not be made. Thus, it is possible that fewer than W or K_t^+ extensions will be made at this stage of the algorithm. The cost vector C is updated with each extension.

The extensions of $B_{t-1}[0]$ naturally appear in sorted order, due to the input data being

pre-sorted. If $W < K_t^+$, then there is no need to create any further extensions of $B_{t-1}[0]$ once W extensions have been placed into the buffer. Note that any extension that would require a person to exit the bus twice is not stored in the buffer, and thus is not counted as one of the W extensions. It is also possible for two paths to correspond to the same set of people exiting but in different orders. The Viterbi algorithm prunes the path of greater cost. The algorithm checks each extension before it is added to B_t. Thus, no two paths in B_t will ever correspond to the same set of people exiting.

The process is now repeated, creating extensions of $B_{t-1}[1]$. If B_t contains less than W states, the extensions are created as before. If B_t contains W states, an insertion sort is used. An extension of $B_{t-1}[1]$ is created, with path cost c_j. If $c_j < C[W-1]$, it is first checked to make sure that it does not correspond to the same set of exiting people as any other stored path. If it does not correspond to any other stored path, an insertion sort is used to put the set of extensions in order by cost. The highest cost extension in B will be dropped as the new extension is added. If it does correspond to another stored path, the path with higher cost is removed. An insertion sort is now used to put the lower cost path in the correct location. If $c_j > C[W-1]$, then the path is not added. The costs of the input data are sorted, thus $c_j < c_{j+i}, i > 0$. If $c_j > C[W-1]$, then $c_{j+i} > C[W-1], i > 0$. There is no need to test any further extensions of $B_{t-1}[1]$.

This process is repeated, creating extensions of all of the elements of B_{t-1}. The elements of B_t are in sorted order after all of the extensions have been made. It is thus ready to be used to find the extensions at stop $t+1$.

Any path from B_{t-1} that does not have an extension in B_t is removed. It is not possible for this path to be used at a later stage in the beam search. It is possible for all but one of the paths associated with B_i to be eliminated. If there is only one valid path at time i, a hard decision can be made.

3.5.5 An Ad Hoc Beam Search

The beam search proposed requires a one-to-one mapping of entering people to exiting people. Section 3.5.2 gave a few examples of instances in which a one-to-one mapping is

not ideal. Also, the accuracy of a beam search is a function of the beam width, W. A wide beam will produce a better approximation of the optimal trellis, but it also increases the amount of computations to be performed. The beam search relies heavily on the order of the exiting people as well. An attempt is made to combine the beam search with the thresholds of the ad hoc technique. It may be possible to reduce the complexity of the beam search without sacrificing accuracy.

The thresholds, t_{d1} and t_m, are used in the same manner as was described in sec. 3.5.2. In the standard beam search, a set of N_p^t people are considered for possible extensions at time t. The third threshold, t_w, is used to limit the number of extensions made. Only the people for whom the associated cost is below t_w are used to create extensions.

An allowance for a many-to-one mapping is also made. Back tracing is no longer performed when an extension is added to the trellis, which allows a person to exit at multiple stops.

Chapter 4

Performance of Matching Algorithms

The previous chapter outlined several sequence estimation techniques. All of those techniques, except for Viterbi algorithm using a full trellis, are tested in this chapter. The marginal accuracy of LDA is also included for comparison.

4.1 Experimental Design

Data has been collected for 70 distinct individuals entering and exiting a simulated bus. In a real-life scenario the direction in which a person travels determines whether a person is entering or exiting. However, in the simulated bus environment outlined in Chapter 2, this distinction between entering and exiting is not necessary. There are two sets of feature vectors collected for each person, and the decision as to which corresponds to the person entering or exiting can be made arbitrarily. In the test outlined in this section this association is done randomly. That is to say that for some people in an experiment, the first set of feature vectors will correspond to those people entering and for the rest of the people in the experiment, the second set of feature vectors will correspond to those people entering. This increases the randomness of the experiments performed. It is important to note that this does not double the number of people in the experiment, it randomly selects which data will be used for training and which data will be used for testing.

It is very difficult to quantify the accuracy of the proposed matching algorithms. The number of errors is dependent on the order in which people exit and the total number of people on the bus. In order to communicate this information as accurately as possible, Monte Carlo analysis is used. In Monte Carlo analysis, an experiment is repeated a large number of times, each time randomly selecting the data to be used. This analysis is done for differing numbers of people on the bus, ranging from 1 to 70. In all cases, a graph will

be shown that displays the accuracy as a function of the number of people on the bus.

In the experiments to be performed, a bus size, N, is chosen. N people are randomly selected from the 70 available people. One of the two sets of feature vectors available for each person is chosen at random to be the training data and a classifier is constructed. In sec. 3.4, LDA performed better than the other classifiers, and thus will be used in all of the experiments performed. The mean is found for each person and an overall covariance matrix is found.

One person is removed from the bus at a time, and the discriminate function value is found for each possible association. People are removed from the bus until the bus is empty. The number of errors, E, is counted and the error percentage can be easily computed as $\frac{E}{N}$. There exist several possible error measures for this system. For the purposes of this thesis an error is an incorrect association of an exit event with an entrance event.

Each experiment is repeated so that the total number of exit events is approximately constant. The total number was chosen to be 7,000. For example, when two people are on the bus, the experiment described above will be performed 3,500 times, and when 70 people are on the bus the experiment is performed 100 times. In this manner the accuracy percentages are calculated using approximately the number of samples in each case.

4.2 Marginal Accuracy

The marginal accuracy of several classifiers was compared in sec. 3.4. Although, the test performed to find the marginal accuracy is different than the tests performed in this chapter, it is included for purposes of comparison.

Figure 4.1 shows the marginal accuracy for differing numbers of people on the bus using LDA, without sequence estimation.

4.3 A Simple Approach

Perhaps the easiest classification test is to always choose the best match at every step. This person is then removed from further consideration. Figure 4.2 shows the accuracy percentage as a function of the number of people on the bus.

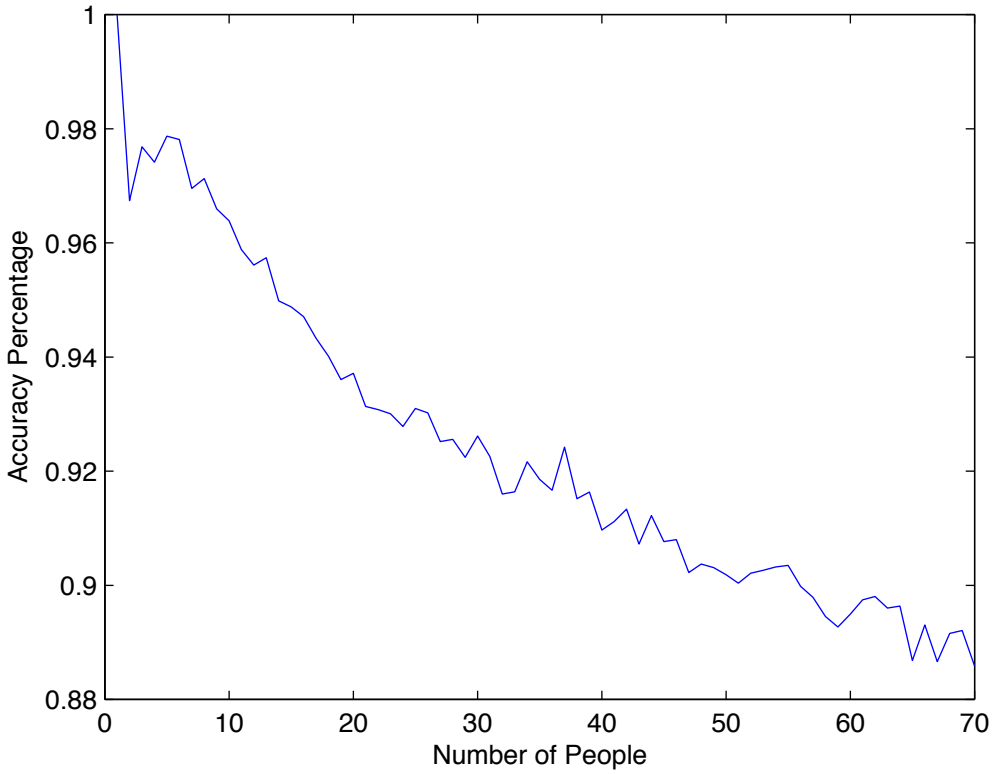


Fig. 4.1: Marginal accuracy.

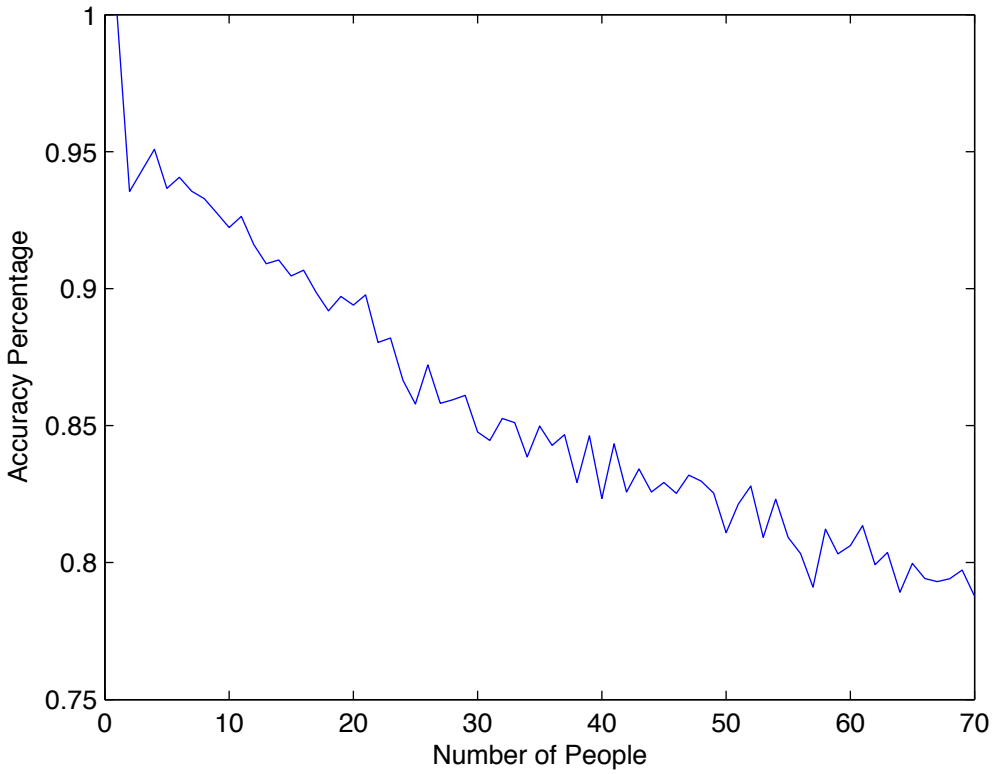


Fig. 4.2: Accuracy of a simple classification technique.

The advantage of this method is simplicity. It takes very little time and memory to compute. It is, however, the least accurate of all of the methods explored. If the best match is in error, there is no way of correcting it at a later time.

4.4 An Ad Hoc Approach

The ad hoc approach exploits several observations that can be made from the data. It relies on three thresholds to make classification decisions. The approach is described in greater detail in sec. 3.5.2.

The thresholds used were found through experimentation. If the difference in the values of the discriminate function for the best match and the second best match is above 35, then a hard decision is made. If the value of the discriminate function of a hard decision is below 150, then that person is removed from further consideration. Lastly, only people for whom the value of the discriminate function is within 35 of the smallest value are considered as possible matches.

Figure 4.3 shows the accuracy for various numbers of people on the bus. The graph also shows the accuracy of the simple approach from the previous section. The ad hoc solution significantly outperforms the previous method for all N. The difference is especially pronounced for large N.

The ad hoc method requires only slightly more memory and computations than the original method. It also only removes a person from consideration if the match is below a threshold. This removes the constraint of a one-to-one mapping of entering and exiting people. This can be viewed as both an advantage and a disadvantage. If there is a large difference between the set of feature vectors collected when a person enters and when the same person exits, an error will oftentimes occur. If a match is incorrect, then this threshold prevents that error from propagating. On the other hand, it is now possible for the same person to exit at multiple stops. If the counting is accurate, then there are people who may potentially never get off the bus.

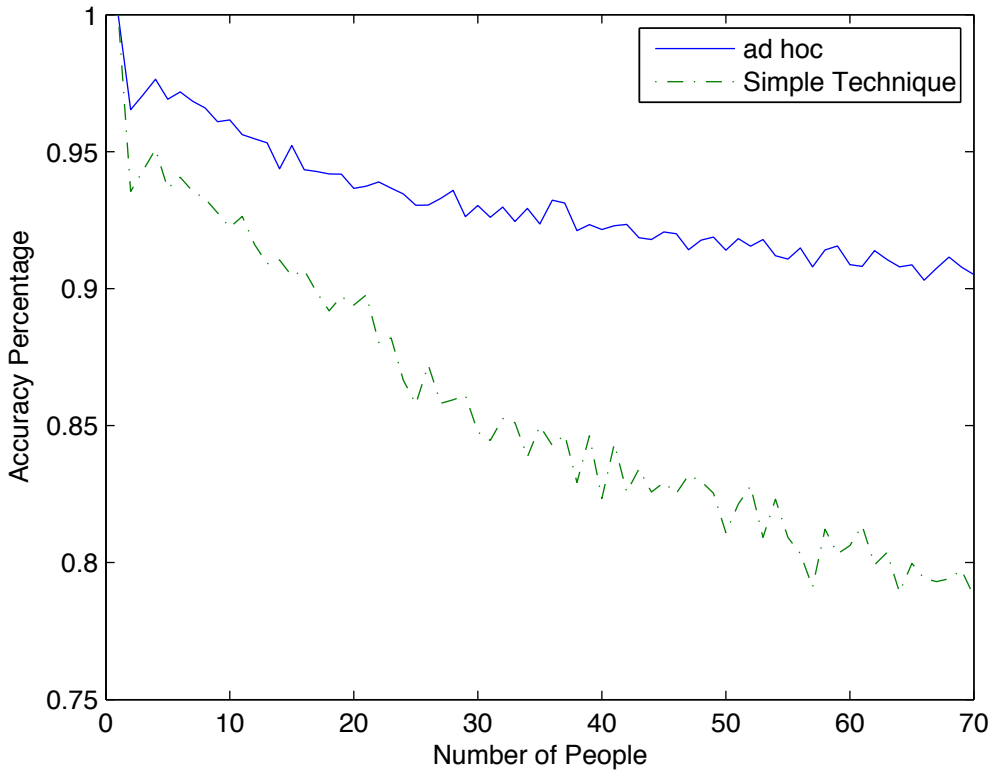


Fig. 4.3: Accuracy of ad hoc technique.

4.5 A Trellis-Based Beam Search Approach

The trellis-based approach is described in more detail in sec. 3.5.3. The full trellis is far to large to store for even small N; thus, a beam search is used where only the W best paths are kept at a given time. There is a trade-off between beam-width and the amount of memory and computations required. Large beam widths are more accurate but the amount of time necessary to use them may be prohibitive.

Beam widths of 1,10,100, and 1000 were tested. Figure 4.4 shows the accuracy at each of the widths. It should be noted that a beam width of 1 simply chooses the best match at each time step. This is exactly the same procedure as the simple test performed in sec. 4.3. The accuracy increases with the beam width. This is to be expected because a larger beam width is a better approximation of the full trellis. A beam width of 1000 is a significant improvement over a beam width of 1, with an increase in accuracy of almost 15% with 70 people on the bus.

It can also be seen that for small numbers of people on the bus, an increase in the beam width has little effect on the accuracy. This occurs because the full trellis can be stored for very small N. Thus, increasing the beam width beyond the maximum width of the trellis has no effect on the outcome.

If the best match at a given stop is not the correct match, it is possible, in some cases, to correct this. This is, however, highly dependent on the ordering of the data. As an example, number the people according to the order in which they exit. If person 2 is matched to person 1, this error may be corrected at the next time step when person 2 exits. If person 70 is matched to person 1, however, the error cannot be corrected until person 70 exits, 69 time steps later.

In the trellis, any path that does not have any successor branches at the current time step is pruned. When, at a given time step, there is only one path that has not been pruned a hard decision is made. The trellis delays those decisions and the beam width controls how long the delay is. For a small beam width, the delay is also small which reduces the error correction capabilities of the trellis.

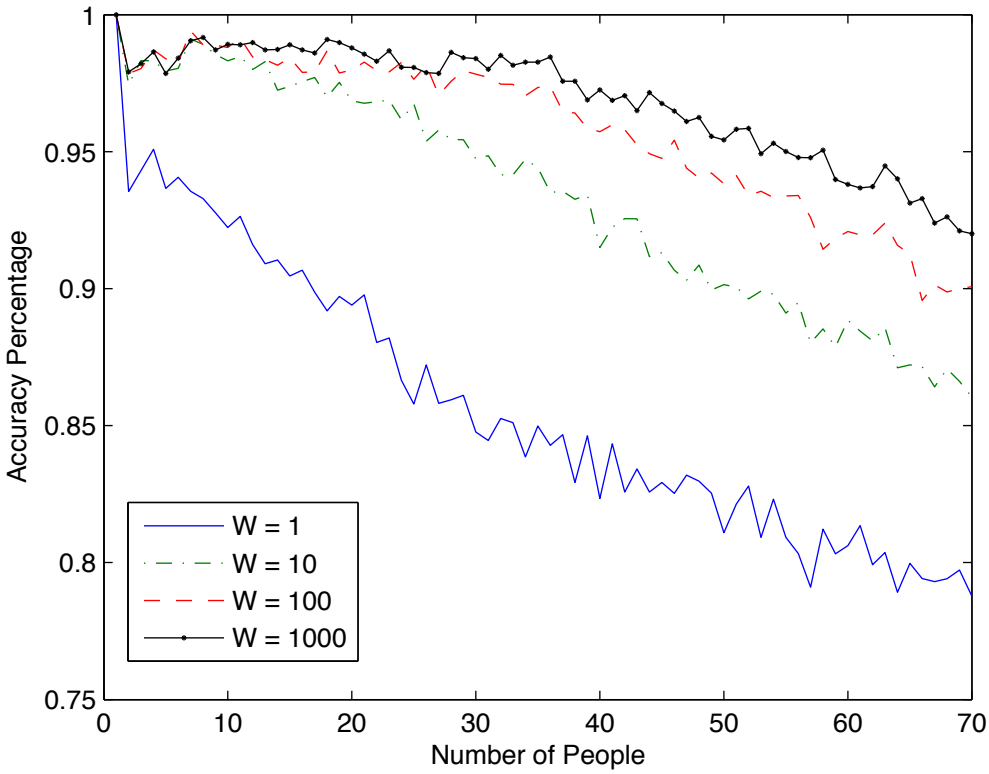


Fig. 4.4: Accuracy using a beam search.

Figure 4.5 shoes the accuracy for beam widths of 10, 100, and 1000 as well as the accuracy of the ad hoc method. The beam search is clearly an improvement over the ad hoc method for a sufficiently wide wide beam. The ad hoc method performs much worse than the beam search for small N. It does, however, decrease in accuracy more slowly than the beam searches. For large N, the ad hoc method is only outperformed by a beam width of 1000.

The beam search requires significantly more memory and computations than the other proposed methods. However, the amount required is still reasonable. There is very little overhead, and each state in the trellis requires only 8 bytes of storage in the current implementation. Even with a beam width of 1000, each time step requires approximately 8 kB of storage. Which is to say that the program would only need a maximum of about 1 MB of memory for 125 time steps of the algorithm with a beam width of 1000. The computations are lengthy but may not be prohibitive. Using a beam width of 1000, it takes about 0.5 seconds to process each event, with 70 people on the bus on a machine with dual 2.67 GHz processors and 8 GB of RAM. The computations for the matching algorithm can be done in between the stops that a bus makes. In most cases this would still allow time to process all of the data before the next bus stop occurs. However, if a large number of people exit the bus at the same time, there may not be sufficient time to process all of the information.

4.6 A Combined Approach

An attempt is made to combine the idea of a beam search with the thresholds of the ad hoc technique. It may be possible to reduce the complexity of the beam search without sacrificing accuracy. For the purposes of comparison the same threshold values are used in this approach as in the ad hoc approach.

Figure 4.6 compares a beam search of width 2 with and without the ad hoc thresholds. It also shows the accuracy of the ad hoc method for comparison. The beam search with thresholds does not perform as well as the ad hoc method, but it is quite a bit better than the beam search without thresholds for large N; however, for small N, $N < 15$, the beam search outperforms both ad hoc methods.

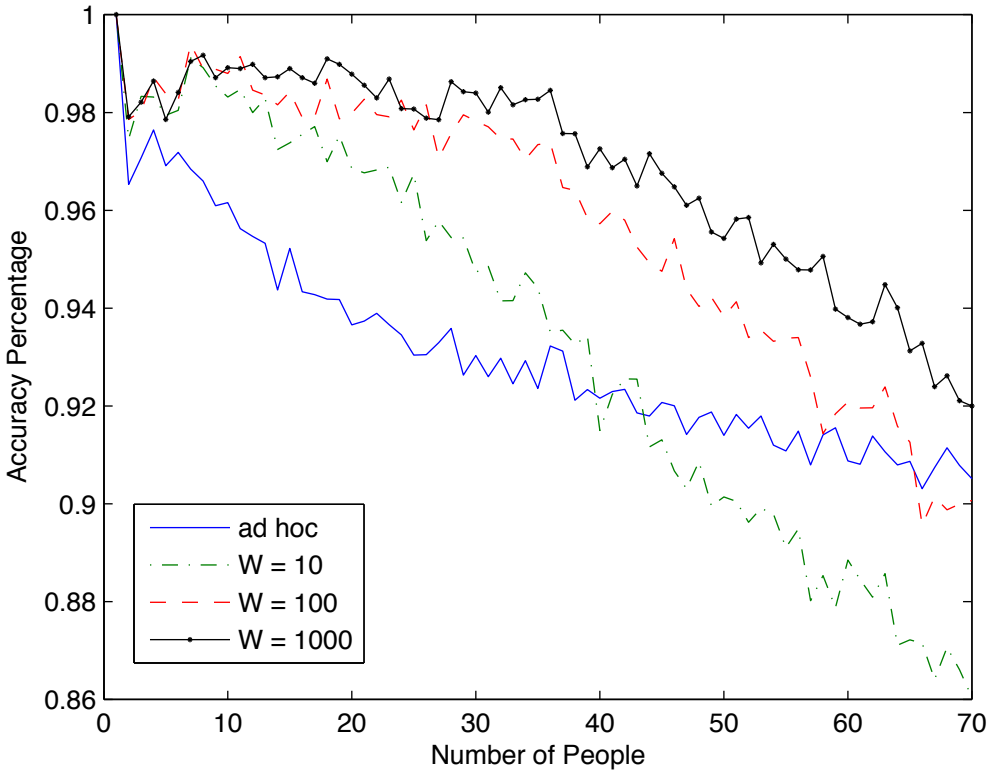


Fig. 4.5: Comparison of ad hoc and beam search methods.

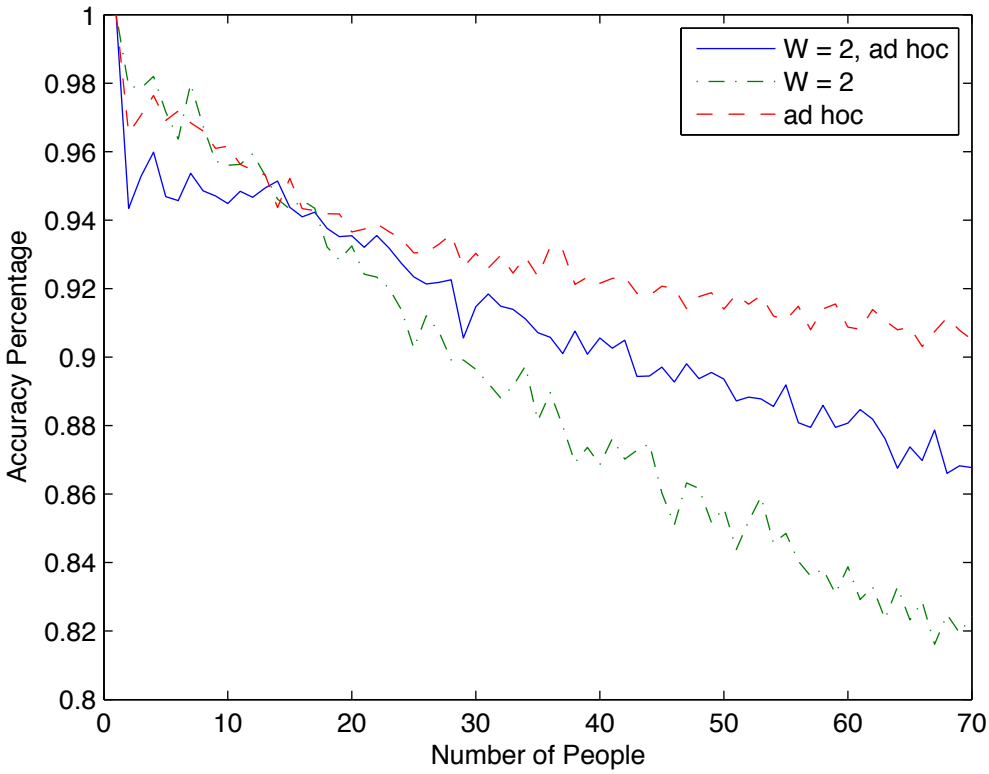


Fig. 4.6: Accuracy of combined technique with beam width of 2.

Figure 4.7 compares a beam search of width 10 with and without the thresholds. As before the accuracy of the ad hoc method is also shown. In this case, the ad hoc methods outperform the beam search for $N > 40$. The benefit of using the thresholds for large N is not near as pronounced as before, although there is still improvement. The ad hoc beam search performs significantly worse than the original beam search for small N.

Figure 4.8 shows the same comparison using a beam width of 100. In this case there is very little difference in the accuracy for all three curves with 70 people on the bus. The crossover point appear to be at $N = 69$. The clear winner out of the three is the original beam search.

All of the widths tested are compared to the ad hoc method in fig. 4.9. The accuracy increases with the width of the beam. The ad hoc method outperforms all of the modified beam searches for small N. All of the beam widths, excluding $W = 2$, perform about as well as the ad hoc method for large N.

The ad hoc beam search does not appear to be very valuable. It sacrifices accuracy for small N for better performance for large N. For a sufficiently wide beam, a improvement is made over the ad hoc technique, but not over the original beam search.

There are several heuristics that could be used to modify both techniques. Only a few were tested here. It is possible that a different set of heuristics could outperform the proposed methods.

4.7 Summary of Test Results

The best method tested is the beam search with a width of 1000. It performs as well or better than every other proposed method for all values of N tested. The beam search is an approximation of an optimal technique in the maximum likelihood sense. The other beam searches perform well, with the performance increasing with the beam width.

The ad hoc method proposed outperforms most of the beam searches for sufficiently large N. It performs very poorly compared to the beam search for small N. It does, however, require fewer computations than a beam search.

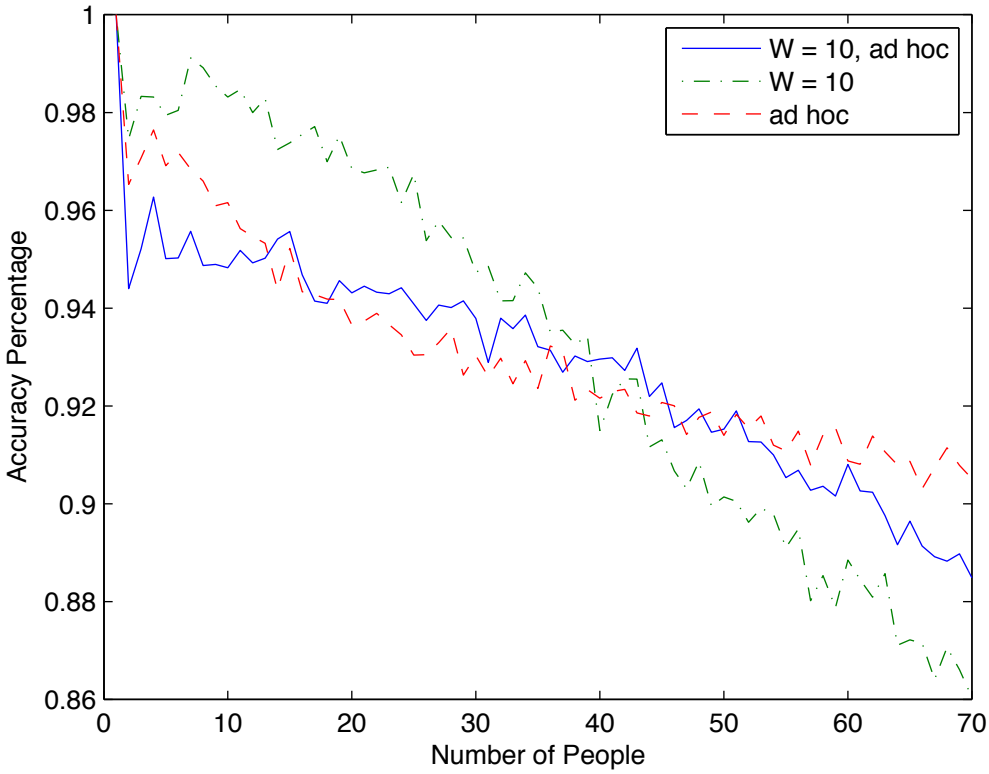


Fig. 4.7: Accuracy of combined technique with beam width of 10.

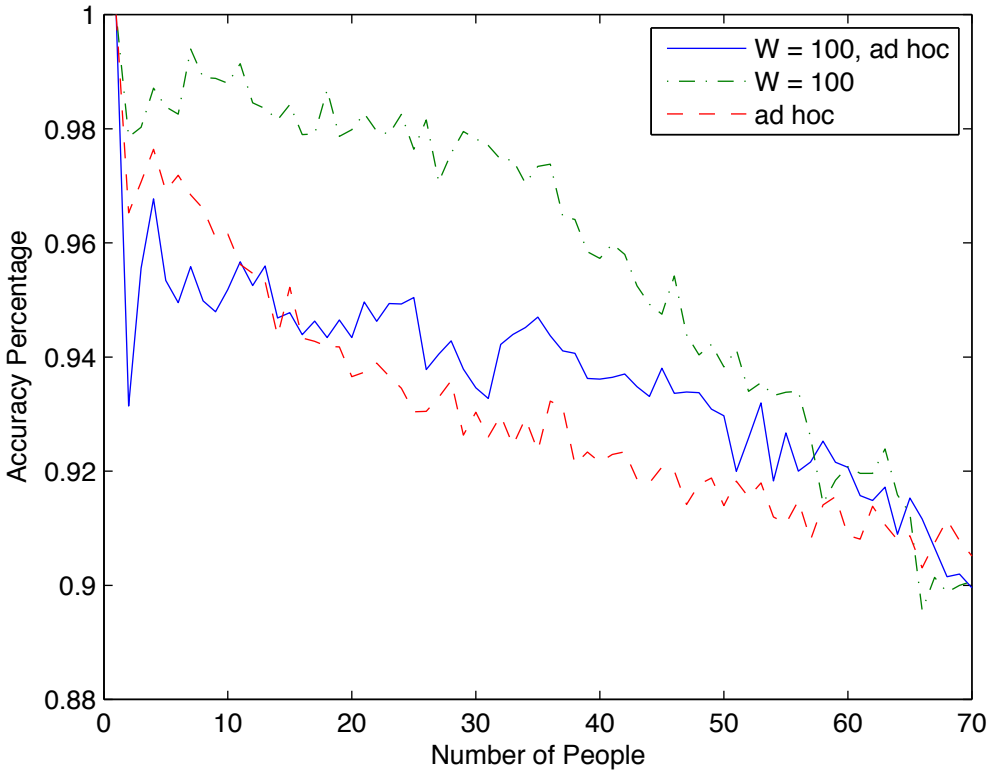


Fig. 4.8: Accuracy of combined technique with beam width of 100.

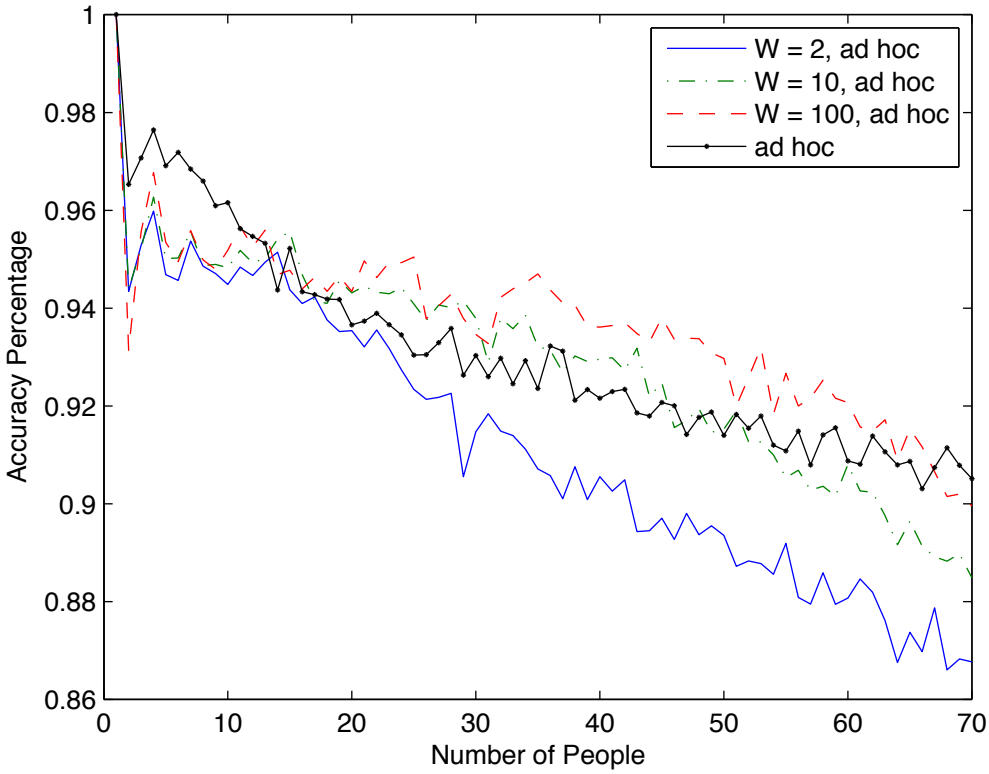


Fig. 4.9: Accuracy of combined technique with various beam widths.

An attempt was made to merge the two methods together. The performance of the beam search with heuristics was increased for large N, but decreased for small N. None of the modified beam searches was able to classify much better than the original ad hoc technique.

4.8 Extensions to a More Realistic Scenario

The techniques were tested in a relatively limited manner. The scenario tested is not very realistic. In an actual bus environment several people may get on or off at a stop. Although, a formal investigation of this topic will not be done here, some observations can be made.

It can be assumed that a person will never get on and off at the same stop. Thus, all of the feature vectors for the exiting people can be processed before those of the entering people. Adding in the information from the entering people first would only increase the probability of error.

When a person enters a bus, the total number of states at the current time step does not change. It does, however, increase the number of states at the next time step. This can be seen in fig. 3.7. Special care will need to be taken to analyze the matching performance in such a situation.

Chapter 5

Conclusions

A people counting and matching system was developed in this thesis. The system utilizes a texel camera, which provides depth information which facilitates the tasks of counting and matching. An algorithm for counting was given and the performance was tested in several situations. The concept of using multiple measurements to classify a person as well as methods of matching a set of people were explored. Tests were performed on various sequence estimation techniques and the results were analyzed. This chapter summarizes the results of the thesis and provides some ideas for future research.

5.1 Summary of Contributions

Chapter 2 outlines the counting algorithm. Depth information from the texel camera is used to separate background and foreground. Block motion prediction is used to track people as they move through the camera's field of view. Head and shoulders are tracked separately for each person. Predictions of possible head-to-shoulder associations are made in each frame. When a person exits the scene, a hard decision is made as to which regions were associated with that person. The system was tested in various difficult situations and was able to count very accurately.

Chapter 3 explores the matching problem. Possible features to be used in classification are explored. A few suboptimal methods of selecting the best set of features for classification are given. An ad hoc method is developed for sequence estimation. An optimal trellis approach and a suboptimal beam search approximation are developed as well.

The techniques developed in the previous chapter are tested, and the results are presented in Chapter 4. In the case of the beam search several beam widths were tested. A hybrid approach is also tested. The performance of the various techniques is analyzed and

compared.

5.2 Ideas for Further Research

The proposed people counting and matching system shows promising results. People counting is a relatively mature field with a wealth of research. The people counter is able to accurately count in many situations that are difficult for other existing counters. The task of matching people is unique. This thesis has show that such a matching system could perform reliably in a bus environment; however, there are still many aspects of the matching problem that have yet to be explored.

The people counter was tested in various situations that are difficult for most people counters. These situations were created in a simulated environment. Further testing needs to be done in actual bus scenarios. There may be situations that occur that were not simulated and other dynamics that could not be predicted.

In this thesis, only one scenario was explored for people matching: one person leaves the bus at a time until the bus is empty. This is the equivalent of having one person exit at each stop and no one enter the bus. Allowing people to enter, and for more than one person to exit at a time, adds new dimension to the problem.

The performance of the matching system can also be tested in more difficult circumstances. A troop of boy scouts may enter a bus. In which case, several of the features collected may not be of any value to distinguish the people.

The effect of missed counts can also be investigated. There may be people on the bus who are never counted as exiting, and people who get off who were never counted as having entered. The beam search proposed requires a one-to-one mapping of entering to exiting people, which assumes that the counting is always accurate. The effects of these missed counts needs to be analyzed.

A matching error was defined as an incorrect association of an entering person with an exiting person. It is very difficult to quantify the performance of the system and this measure may not be the best. It is also important to ask questions such as, “How do the errors effect the statistics for each stop?” and “Where do the errors occur?” The

information that a transit system will use in decision making needs to be known in order to decide on an error measure.

A trellis-based sequence estimation approach was utilized. A beam search was used to reduce the complexity of the problem. In a beam search, the order in which people exit has a large effect on the accuracy. An ad hoc method was proposed which reduces the dependency on ordering. In general, it did not perform as well as the beam search. It may be possible to add other heuristics to the beam search algorithm that will perform better.

A different kind of algorithm could also be used. The algorithm used is based on Viterbi algorithm, which estimates the most likely sequence of events. The Bahl, Cocke, Jelinek, and Raviv (BCJR) algorithm chooses the most likely event to occur at a given time given all of the previous and future events [29]. The incorporation of future events would remove the dependency on the order in which people exit. Like the original trellis approach, the number of states and computations needed to perform this algorithm would be prohibitive. A simplification, similar to the beam search, would need to be made.

Lastly, this thesis only considered the matching problem on a bus. The system could be extended for use in a train or a mall. There could be a much larger number of people to consider in these cases. It would be interesting to see the accuracy of the system when $N = 200$ or $N = 500$.

References

[1] TrafSYS, "People counters," [http://www.trafsys.com/people-counters.aspx], 2009.

[2] ACOREL, "ACOREL Onboard Counter," [http://www.acorel.com/en/ACOREL%20Onboard%20Counter.pdf], 2009.

[3] Canesta, "Canesta 101: Introduction to 3d vision in cmos," [http://www.canesta.com/assets/pdf/technicalpapers/Canesta101.pdf], 2008.

[4] S. Göktürk, H. Yalcin, and C. Bamji, "A time-of-flight depth sensor - system description, issues and solutions," in *Computer Vision and Pattern Recognition Workshop*, pp. 35–35, June 2004.

[5] B. Boldt, "Point cloud matching with a handheld texel camera," Master's thesis, Utah State University, Logan, 2007.

[6] S. B. Göktürk and C. Tomasi, "3d head tracking based on recognition and interpolation using a time-of- flight depth sensor," in *Computer Vision and Pattern Recognition*, 2004.

[7] "Anthropometric data sets," [http://www.dtic.mil/dticasd/anthro_ds.html], 2009.

[8] C.-H. Chen, Y.-C. Chang, T.-Y. Chen, and D.-J. Wang, "People counting system for getting in/out of a bus based on video processing," in *Eighth International Conference on Intellignet Systems Design and Application, IEEE*, pp. 565–569, 2008.

[9] O. Masoud and N. P. Papanikolopoulos, "A novel method for tracking and counting pedestrians in real-time using a single camera," *IEEE Transactions on Vehicular Technology*, vol. 50, no. 5, pp. 1267–1278, Sept. 2001.

[10] A. Albiol, "Real-time high density people counter using morphological tools," *IEEE Transactions on Intelligent Transportation Systems*, vol. 2, no. 4, pp. 204–218, Dec. 2001.

[11] E. Polat, M. Yeasin, and R. Sharma, "Tracking body parts of multiple people: a new approach," in *IEEE Workshop on Multi-Object Tracking*, pp. 35–42, 2001.

[12] I. Cox and S. Hingorani, "An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol. 18, no. 2, pp. 138–150, Feb. 1996.

[13] D. Reid, "An algorithm for tracking multiple targets," *Automatic Control, IEEE Transactions*, vol. 24, no. 6, pp. 843–854, Dec. 1979.

[14] T. Zhao and R. Nevatia, "Bayesian human segmentation in crowded situations," in *Computer Vision and Pattern Recognition, IEEE*, pp. 459–466, 2003.

[15] M. Fukumi, S. Karungaru, and Y. Mitsukura, "Feature generation by simple-flda for pattern recognition," *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference*, vol. 2, pp. 730–734, Nov. 2005.

[16] S. Bagui, S. Bagui, A. Chatterjee, and K. Mehra, "Classification with multiple independent measurements under a separate sampling scheme," *Statistical Methodology*, vol. 3, no. 3, pp. 234 – 251, 2006.

[17] G. Shakhnarovich, J. W. Fisher, III, and T. Darrell, "Face recognition from long-term observations," in *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III*. London: Springer-Verlag, pp. 851 – 868, 2002.

[18] A. Roy and R. Khattree, "Discrimination and classification with repeated measures data under different covariance structures," *Communications in Statistics - Simulation and Computation*, vol. 34, no. 3, pp. 167–178, 2005.

[19] P. Gupta, D. Doermann, and D. DeMenthon, "Beam search for feature selection in automatic svm defect classification," in *Pattern Recognition, 2002. Proceedings, 16th International Conference*, vol. 2, pp. 212–215, 2002.

[20] H. Ney and S. Ortmanns, "Dynamic programming search for continuous speech recognition," *Signal Processing Magazine, IEEE*, vol. 16, no. 5, pp. 64–83, Sept. 1999.

[21] *Electronic Perception SDK Reference Manual*, Canesta Inc., 2004.

[22] T. Mller, H. Kraft, J. Frey, M. Albrecht, and R. Lange, "Robust 3d measurement with pmd sensors," in *Proceedings of the First Range Imaging Research Day at ETH Zurich*, 2005.

[23] D. H. Ballard and C. M. Brown, *Computer Vision*. Englewood Cliffs, NJ: Prentice Hall, 1982.

[24] T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms for Signal Processing*. Upper Saddle River: Prentice Hall, 2000.

[25] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning; data mining, inference, and prediction*. New York: Springer, 2001.

[26] M. Tuceryan and A. K. Jain, "Texture analysis," in *The Handbook of Pattern Recognition and Computer Vision*, 2nd ed. Hackensack, NJ: World Scientific Publishing Co., ch. 2, pp. 207–248, 1998.

[27] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. San Diego: Academic Press, 1999.

[28] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *Information Theory, IEEE Transactions*, vol. 13, no. 2, pp. 260–269, Apr. 1967.

[29] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (corresp.)," *Information Theory, IEEE Transactions*, vol. 20, no. 2, pp. 284–287, Mar. 1974.