Masthead Logo

Transportation Research Center Reports

Transportation Research Center for Livable Communities

8-31-2015

14-12 Capacity Analysis of Pedestrian Facilities Involving Individuals with Disabilities

Mohammad Sadra Sharifi
Utah State University

Keith Christensen
Utah State University

Anthony Chen
Utah State University

Follow this and additional works at: https://scholarworks.wmich.edu/transportation-reports

Part of the Disability Studies Commons, and the Transportation Engineering Commons

Footer Logo

TRCLC 14-12
August 31, 2015

Capacity analysis of pedestrian facilities involving individuals with disabilities

FINAL REPORT

Mohammad Sadra Sharifi, Keith Christensen, Anthony Chen.

Utah State University
Logan, UT 84322

Transportation Research Center
for Livable Communities

Western Michigan University | University of Texas at Arlington | Utah State University | Wayne State University | Tennessee State University

Technical Report
Documentation Page

1. Report No. TRCLC 14-12	2. Government Accession No. N/A	3. Recipient's Catalog No. N/A
4. Title and Subtitle Capacity analysis of pedestrian facilities involving individuals with disabilities		5. Report Date August 31, 2015
		6. Performing Organization Code N/A
7. Author(s) Mohammad Sadra Sharifi, Keith Christensen, Anthony Chen.		8. Performing Org. Report No. N/A
9. Performing Organization Name and Address Utah State University 4110 Old Main Hill Logan, UT 84322		10. Work Unit No. (TRAIS) N/A
		11. Contract No. TRCLC 14-12
12. Sponsoring Agency Name and Address Transportation Research Center for Livable Communities (TRCLC) 1903 W. Michigan Ave., Kalamazoo, MI 49008-5316		13. Type of Report & Period Covered Final Report 6/3/2013 - 8/1/2015
		14. Sponsoring Agency Code N/A
15. Supplementary Notes		

16. Abstract

Walking facilities are important infrastructures that must be designed to accommodate the behavior of pedestrians in order to be effective. Heterogeneity in pedestrian composition is one important factor generally overlooked in walking facility design guidelines. Particularly, individuals with disabilities are often ignored due to lack of available data on their pedestrian behaviors. A controlled, large-scaled walking experiment involving individuals with disabilities was conducted at Utah State University to observe individual pedestrian behaviors in various walking facilities; these facilities include passageway and bottleneck formations. The purpose of this report is twofold: (1) to model time headway between different individual types using a mixed distribution model, and (2) to estimate passageway and bottleneck capacities and to identify the impacts of involving individuals with disabilities on capacity estimations. Results showed that the proposed model had good performance for pedestrian time headway modeling. Analysis also revealed that visual impaired individuals and individuals using motorized wheelchairs had the minimum and maximum capacity reduction effect on passageways and bottlenecks. The findings are expected to improve the facility capacity estimations required and to meet a preferred level-of-service for heterogeneous populations.

17. Key Words Pedestrian walking behavior, individuals with disabilities, pedestrian facilities, capacity analysis		18. Distribution Statement No restrictions.		
19. Security Classification - report Unclassified	20. Security Classification - page Unclassified		21. No. of Pages 72	22. Price N/A

Disclaimer

The contents of this report reflect the views of the authors, who are solely responsible for the facts and the accuracy of the information presented herein. This publication is disseminated under the sponsorship of the U.S. Department of Transportation's University Transportation Centers Program, in the interest of information exchange. This report does not necessarily reflect the official views or policies of the U.S. government, or the Transportation Research Center for Livable Communities, who assume no liability for the contents or use thereof. This report does not represent standards, specifications, or regulations.

Acknowledgments

Table of Contents

List of Tables

List of Figures

Executive Summary

Walking facilities are important infrastructures that must be designed to accommodate the behavior of pedestrians in order to be effective. Heterogeneity in pedestrian composition is one important factor generally overlooked in walking facility design guidelines. Particularly, individuals with disabilities are often ignored due to lack of available data on their pedestrian behaviors. A controlled, large-scaled walking experiment involving individuals with disabilities was conducted at Utah State University to observe individual pedestrian behaviors in various walking facilities; these facilities include passageway and bottleneck formations. The purpose of this report is twofold: (1) to model time headway between different individual types using a mixed distribution model, and (2) to estimate passageway and bottleneck capacities and to identify the impacts of involving individuals with disabilities on capacity estimations. Results showed that the proposed model had good performance for pedestrian time headway modeling. Analysis also revealed that visual impaired individuals and individuals using motorized wheelchairs had the minimum and maximum capacity reduction effect on passageways and bottlenecks. The findings are expected to improve the facility capacity estimations required and to meet a preferred level-of-service for heterogeneous populations.

1. Introduction

The design of pedestrian infrastructure is an important process usually achieved by means of supply/demand analysis. Facility designers are needed to project pedestrian demands and to estimate the capacity required to meet a preferred level-of-service (LOS). Critical to this process is correctly estimating infrastructure supply levels or capacities. Presently, facility designers use guidelines found in the Highway Capacity Manual (HCM) (TRB, 2010) and the International Building Code (IBC) (ICC, 2012) based on pedestrian flow characteristics obtained through empirical research.

However, the walking design guidelines have assumed typical homogenous population characteristics. Heterogeneity in pedestrian composition is an important factor because different types of pedestrians act in different ways. While individuals with disabilities constitute a significant portion of the population in the United States, accounting for 12.6% (or about 30.2 million) of the working age population and 16.7% (or about 51.5 million) of the total population (U.S. Census Bureau, 2010), pedestrian facilities do not account for diversity in physical abilities. Current walking design guidelines limit the disabled, non-conforming subset of the population to facility capacities for homogeneous populations. To account for the needs of individuals with disabilities, the Americans with Disabilities Act (ADA, 1990) and Americans with Disabilities Act Accessibility Guidelines (ADAAG, 2002) provide rules and guidelines for the design of pedestrian facilities. These codes are based only on physical properties and do not consider the heterogeneity in pedestrian flow characteristics. On the other hand, existing researches studied limited number of geometric walking designs and there is a significant lack of investigation focused on analyzing the capacities of different walking facilities encountering homogenous and heterogeneous population scenarios. Therefore, it is worthwhile to address and analyze the relationships of basic traffic flow variables for different walking environments.

To overcome the limitations, a controlled large-scaled walking experiment involving individuals with disabilities was conducted at Utah State University (USU) to explore the impacts of walking behaviors on capacity of various walking facilities. This report presents a mixed time headway distribution model to study the interaction behaviors of different individual types in uni-directional pedestrian flow. Moreover, behavior effects on capacity of passageway

and bottleneck were investigated. The remainder of this report is organized as follows. The next section briefly describes the experimental design process and the data collection method. Afterwards, methodology including pedestrian time headway definition and mixed time headway distribution is discussed. The research outcomes including trajectory, time headway modeling, and capacity analysis are presented in subsequent sections. Finally, concluding remarks are presented. The findings are expected to improve estimation of facility capacity, which is a required process to meet a preferred level-of-service for heterogeneous populations. Furthermore, the results of the research may be used to develop well-characterized individual-based theories.

2. Literature Review

2.1. Confirmatory review on existing regulations and guidelines

Planners generally use existing regulations and guidelines for designing and assessing public pedestrian facilities. The Highway Capacity Manual (HCM) (TRB, 2010), the International Building Code (IBC) (ICC, 2012), and the Americans with Disabilities Act Accessibility Guidelines (ADAAG, 2002) are three reference manuals generally used in the United States to design and evaluate capacities of different outdoor walking facilities (i.e. sidewalks with different geometrics) and indoor walking facilities (i.e. sizing building components). This section provides a review on these design guidelines to identify the properties of different references.

2.1.1. Highway Capacity Manual (HCM)

The Highway Capacity Manual (HCM), published by Transportation Research Board (TRB), is extensively used for designing and assessing transportation facilities in the United States. While HCM has been viewed as a reference document in engineering analysis process, it doesn't constitute a legal standard for transportation facility design. Originally published in 1950, this guideline was the first manual to define and quantify the concept of capacity for different transportation facilities. (TRB, 2010). This measure assists planners, designers, and operators in evaluating the adequacy of transportation facility supplies to meet the predicted demand. In early versions, only methodologies to evaluate capacity of roadway elements (i.e.

freeway, highway, streets, etc.) were provided. However, the fourth edition was extended to enable the evaluation of different pedestrian facilities including walkways, pedestrian queuing areas (i.e. elevators, transit platforms), shared off-street paths, pedestrian crosswalks, and pedestrian facilities along urban streets. The following macroscopic traffic flow definitions were used in the HCM for pedestrian capacity analysis (TRB, 2010):

- **Pedestrian flow rate**: Pedestrian flow rate is the number of pedestrians passing a line across the width of a walkway perpendicular to the pedestrian path per unit of time. Pedestrian flow rate can be determined for unit of effective width expressed as pedestrian per minute per meter (P/min/m).
- **Pedestrian density**: Pedestrian density is defined as the average number of pedestrians per unit of area within a walkway expressed as pedestrians per square meter (P/m^2).
- **Pedestrian space**: Pedestrian space is the inverse of density and it determines the average area provided for each pedestrian in a walkway. Space unit is expressed as square meters per pedestrians (m^2/P).

The proposed capacity analysis methods in HCM guidelines are mainly based on the relationships among macroscopic traffic flow variables (i.e. flow, density, space). These relationships can be presented using fundamental traffic flow diagrams. HCM adopts several basic research efforts on these diagrams for capacity analysis purposes. Fundamental diagrams presented in the guideline are generally obtained from basic empirical studies by Fruin (1987), Older (1968), Oeding (1963), Navin and Wheeler (1969), and Pushkarev and Zupan (1975). Figure 1 shows relationships between pedestrian flow and space for different populations, extracted from different empirical studies.
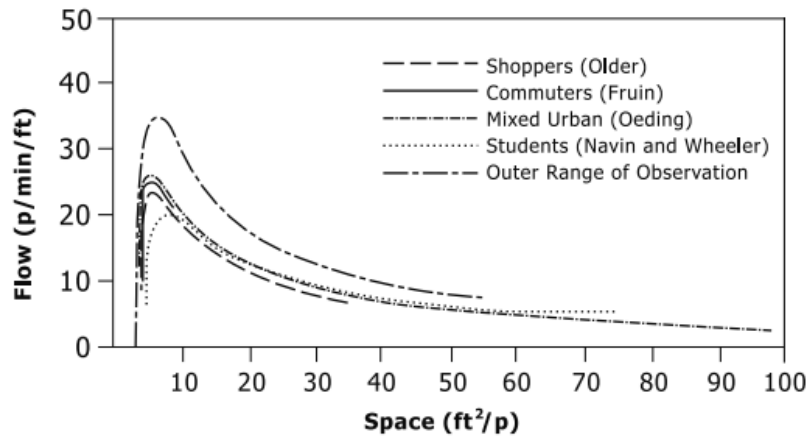
Figure 1. Relationships between pedestrian flow and space for different populations. (TRB, 2010)

Generally, pedestrian flow increases with increasing pedestrian space up to a certain range of space. Then, flow rates decline because of existing too much space between pedestrians. HCM determines the capacity of walking facilities specifying maximum observed pedestrian flow. Figure 1 indicates that the maximum pedestrian flow (i.e. capacity) varies between 65 p/min/m to 110 p/min/m and it lies within a certain range of space from 0.4 m^2/p to 0.9 m^2/p. Although the HCM guideline provides a systematic way for capacity analysis, there is a limitation in the proposed method. HCM analyzes the capacity of walkways using macroscopic properties of pedestrian flow. It does not consider microscopic behavior of pedestrians. Therefore, it is not possible to study on the impact of heterogeneity in pedestrian compositions and behaviors on the capacity of walking facilities. Also, the fundamental diagrams provided in the guidelines are limited for straight walkways and different walking geometrics were not studied.

2.1.2. International Building Code (IBC)

The International Building Code (IBC), which is published by the International Code Council (ICC), is a standard reference addressing design and building systems requirements. This manual establishes the minimum requirements to guarantee the performance of buildings during emergency situations. The sizing requirements are mainly based on past experiences in consequence of some hazardous situations. The manual contains 35 chapters defining regulations for different building components. Chapter 10, "Means of egress," mainly focuses on designing

indoor walking facilities in buildings including corridors, ramps, and stairways. This chapter defines minimum sizing for different building elements in order to provide an effective means of egress (i.e. unobstructed egress path from occupied portion of a building to a public way). The code classifies the buildings into different types, including residential buildings, business buildings, and high rise buildings, and establishes the minimum sizing with respect to building categories. For example, it requires that corridor widths should be at least 36 inches for buildings with occupant loads lower than 50. This code also determines the capacity (i.e. maximum occupant loads) for different built environments with respect to building category. For instance, it considers requirements of 100 gross floor area (GFA) for each occupant in business area. Thus, a 120,000 sq. ft. building used for business occupancy can accommodate a maximum 1200 people. It can be found that pedestrian flow characteristics and occupant specifications were not investigated and requirements were established only based on safety considerations. Therefore, this code may either overestimate or underestimate the capacity of built environments.

2.1.3. Americans with Disabilities Act Accessibility Guidelines (ADAAG)

While vulnerable groups of people (including individuals with disabilities) are a significant portion of the population of United States, most walking facility and building design guidelines overlook them in their design considerations. To account for the needs of individuals with disabilities in society, U.S. Congress established a federal act called "Americans with Disabilities Act (ADA)" in 1990 (ADA, 1990). This law prohibits discrimination based on disability in the United States. Americans with Disabilities Act Accessibility Guidelines (ADAAG) is a manual containing requirements for building and walking facility designs to accommodate the needs of individuals with disabilities. This guideline includes 15 chapters containing regulations for different public environments. Chapter 4, "Accessible elements and spaces: scope and technical requirements," mainly describes sizing requirements for different building components such as corridors, ramps, stairs, etc. Figure 2 presents the required sizing for a corridor to consider people with wheelchair specifications.
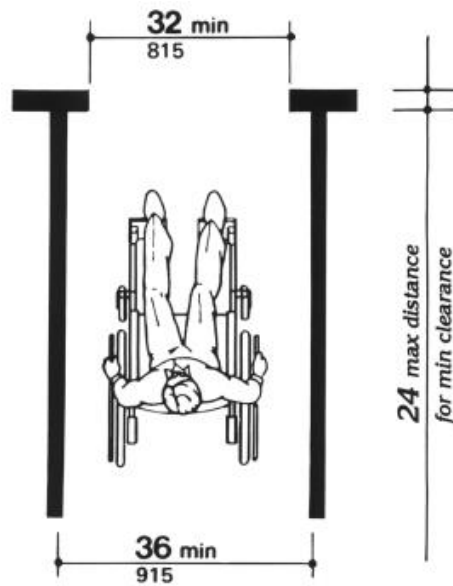
Figure 2. Required sizing for a corridor considering wheelchair dimensions. (ADAAG, 2002)

Although ADAAG considers vulnerable pedestrian groups in indoor walking facility design, the regulations are not able to account for interactions between people with and without disabilities. On the other hand, this guideline does not provide any systematic way to determine the capacity of different walking environments considering individuals with disabilities. Therefore, whether this regulatory standard can accommodate all walking needs of individuals with disabilities is questionable.

2.2. Literature review on pedestrian facilities capacity analysis

In order to provide effective walking infrastructure, designers should have insight into the capacity of walking facilities to meet the preferred level of service for planned walking demands. In the pedestrian literature, many researchers have extensively explored macroscopic pedestrian traffic flow characteristics to study walkway capacities and operational performance of walking facilities. These studies began in 1963 with an attempt to study on pedestrian flow characteristics in Germany. Oeding (1963) collected pedestrian volumes, densities and speeds in a shopping street and examined relationships between them. Five years later, he collected and analyzed macroscopic characteristics of pedestrian flow in a shopping street in London, United Kingdom (Older 1968). He then developed a fundamental diagram to specify the performance of walkways. Navin and Wheeler (1969) recorded pedestrian flow variables on walkways at three

locations on the University of Missouri campus in Columbia. They provided fundamental relationships between pedestrian speed, density, and flow. Polous et al. (1983) collected pedestrian data in the central business district of Haifa, Israel, using a videotape recorder and a digital clock. They analyzed properties of pedestrian flow on sidewalks and calibrated pedestrian traffic flow models. Tanariboon et al. (1991) conducted research on several sidewalks in Singapore and recorded pedestrian movements using a video recorder. They extracted macroscopic pedestrian flow variables using photographic techniques and proposed mathematical models for fundamental flow relationships (i.e. speed-density, speed-flow, and flow, density). Calibrated models revealed that the optimal pedestrian space and maximum observed flow (i.e. capacity) were about 0.7 m^2/p and 90 p/min/min, respectively. Other primary efforts on pedestrian flow modeling can be found in studies by Pushkarev and Zupan (1975), Khisty (1985), Tanaboriboon and Guyano (1991), Daly et al. (1991), Ando et al. (1988), and Virkler and Elayadath (1994).

Later, more advanced technologies were used to collect pedestrian stream characteristics. Lam and Cheung (2000) empirically investigated the effects of bi-directional pedestrian flows on free-flow walking speed, at-capacity walking speed, and effective capacity for a selected indoor walkway in Hong Kong.   Helbing et al. (2007) analyzed a crowd disaster in Makkah, Saudi Arabia during the Hajj pilgrimage using video recording data. They explored relationships between macroscopic fundamental variables and analyzed various self-organization phenomena during the disaster. Ye et al. (2008) collected data for longitudinal pedestrian flows (i.e. unidirectional and multidirectional flows) in a metro station in Shanghai, China using video recordings. They calibrated pedestrian fundamental traffic flow diagrams for different indoor walking facilities including level passageway and stairs (ascending, descending and two-way). Based on calibration results, they concluded that the capacity of ascending stairways are slightly higher than descending stairways and two-way stairs have considerable lower capacities than one-way stairs. Most of the mentioned studies have been conducted in in urban areas. Pedestrian traffic density on sidewalks does not regularly reach to high extreme levels. Therefore, there is a significant lack of observations in density ranges in which the walking facility is operating at its capacity level. In response, controllable walking experiments have been conducted by many researches to collect pedestrian data for extreme conditions such as highly congested situations.

Daamen and Hoogendoorn (2003) conducted walking experiments at Delft University of Technology in Netherlands to derive walking behavior in passageways and bottlenecks under different pedestrian flow scenarios such as un-directional, bi-directional, and cross pedestrian flows. A sample representative for the Dutch population with 80 participants was invited and ten experiments were performed to observe pedestrian walking behavior in standard, station, and shopping conditions. They observed and analyzed pedestrian stream characteristics for a wide range of density levels, from free-flow conditions to extremely congested situations. A fundamental diagram was developed to analyze operation performance of the walking facilities. Specifically, they found that the capacity of the bottleneck facility was approximately 90 p/min/m for uni-directional pedestrian flow.

Another set of controlled walking experiments was administered in Germany to analyze and evaluate performance of various walking facilities such as circular passageway (Seyfried et al., 2005), a corridor (Kretz et al., 2006 (a)), a bottleneck (Kretz et al., 2006 (b)), a T-junction (Zhang et al., 2011), and a set of stairs (Burghardt et al., 2013). Seyfried et al. (2009) examined the capacity of bottlenecks with different widths under uni-directional pedestrian stream. 18 runs of experiments were conducted using 20, 40 and 60 pedestrians. Data analysis revealed that the bottleneck capacity grew linearly with increasing width. Wong et al. (2010) developed and calibrated a bidirectional pedestrian model with an oblique intersecting angle through controlled walking experiments. They used the calibrated model to explore pedestrian flow characteristics in oblique angle environment.

2.3. Criticism on existing capacity analysis approaches

As summarized above, a great deal of study has been conducted on pedestrian stream characteristics and capacity of different walking environments. However, there are two limitations embedded in the existing regulations and pedestrian studies: (1) these studies did not address the pedestrian flow characteristics involving people with mobility and visual constraints, and (2) the proposed capacity estimation methods were not able to account for pedestrian microscopic behaviors.

While individuals with disabilities constitute a significant proportion of the population of United States, little is understood concerning the effect of involving such individuals (the

heterogeneous crowd) on the capacity and flow conductibility of different build environments. Most of existing walking facility guidelines and regulations such as the HCM and the IBC code overlook individuals with disabilities as part of pedestrian stream and they do not account for the impact of individuals with disabilities on walkway capacity evaluations. Only the ADAAG manual proposes building facility design considering individuals with disability needs. However, this code establishes the sizing of the walking facilities based only on dimensions and space needs of individuals with disabilities; it does not account for interactions between individuals and built environments. In addition, the guideline does not provide a systematic way to evaluate the capacity of walking environments in presence of individuals with disabilities. There is a limited number of studies considering people with low mobility, including individuals with disabilities in capacity analysis process. Daamen and Hoogendoorn (2011) conducted a research experiment in the Netherlands to investigate the capacity of doorways with consideration of elderly and disabled people. They analyzed the relation between doorway capacities, population compositions, and stress level during emergency situations.

Generally, proposed capacity estimation approaches use macroscopic fundamental diagrams to estimate the capacities. These diagrams are developed based on macroscopic flow characteristics. Therefore, these approaches are incapable of capturing the impacts of any one individual's behavior on the capacity of walking facilities. Only one study by Hoogendoorn (2004) investigated pedestrian behavior implications for bottleneck capacities. However, the study did not include individuals with disabilities. The presence of special components in the pedestrian flow stream, such as individuals with mobility and visual constraints, may have a substantial impact on walkway capacities. In this case, walking design requirements need to be modified accordingly to consider walking needs of all types of pedestrians. Table 1 summarizes some existing walking facility guidelines and pedestrian studies and their approaches in walking capacity analysis.

Table 1. Summary of capacity analysis specifications in manuals and pedestrian studies.

Reference	Approach		Considering Individuals with disabilities	Facility types
	Macroscopic	Microscopic		
HCM	•		No	Crosswalk
IBC	•		No	Building components
ADAAG	•		Yes	Building components, crosswalk
Oeding (1963)	•		No	Crosswalk
Older (1968)	•		No	Crosswalk
Navin and Wheeler (1969)	•		No	Crosswalk
Polous et al. (1983)	•		No	Crosswalk
Tanariboon et al. (1991)	•		No	Crosswalk
Lam and Cheung (2000)	•		No	Indoor walkways
Helbing et al. (2007)	•		No	Circular passageway
Ye et al. (2008)	•		No	Level passageway, stairs
Daamen and Hoogendoorn (2003)	•		No	Passageway, bottleneck
kretz et al. (2006)	•		No	Corridor
Zhang et al. (2011)	•			T-junction
Burghardt et al. (2013)	•			Stair
Seyfried et al. (2009)	•		No	Bottleneck
Wong et al. (2010)	•			Oblique angle
Daamen and Hoogendoorn (2011)	•		Yes	Doorway
Hoogendoorn (2004)		•	No	Bottleneck

3. Data Collection

The research goal was to study the pedestrian stream, including people with disabilities, in a variety of walking facilities at varying congestion levels. To accomplish this goal, a large-scale walking experiment was conducted at Utah State University. To conduct the experiments, a temporary circuit was constructed using six-foot, self-standing wall panels. This circuit constitutes the desired walking facilities (level passageway and bottleneck). Figure 3 shows the circuit layout.
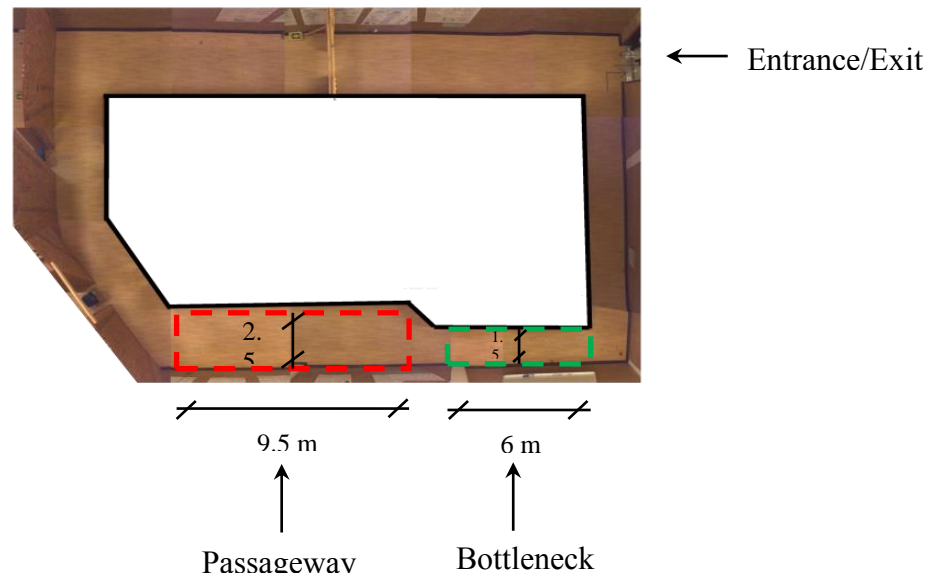
Figure 3. Layout of the circuit with passageway and bottleneck facilities.

Five classifications of individuals participated in the experiment: individuals without disabilities, individuals with visual impairments, individuals who use mobility canes, individuals who use non-motorized devices for walking (e.g., wheelchair/roller walker), and individuals using motorized wheelchairs. In total, 202 participants (180 without disabilities and 42 with disabilities) were recruited for the experiments.

The circuit experiments were conducted over three days (November 2nd, 9th, and 15th, 2012). Experiments were categorized into two groups: uni-directional and bi-directional. In the first case all participants walked in the same direction but, in the latter case, the experiments were done with different flow composition scenarios (90% major stream, 10% minor stream, 80

major 20% minor, 70% major 30% minor, 60% major 40% minor, and 50% major 50% minor). Each scenario was split into 10-minute data collection with four hours of data collection each day. Note that only data collected from the uni-directional stream, the 90% major stream, and the 10% minor stream was used in this study. The minor stream was assumed to not have substantial impacts.

Automated video identification and tracking technology were used for data collection. The technology can locate participant positions with an average accuracy of 0.3 meters or a footstep, which enables tracking and collection of each individual participant's walking trajectory. Derived for augmented reality, ARToolKitPlus (ARTKP) is a software library that allows the tracking of up to 512 identifiable markers in a camera field at once (Wagner and Schmalstieg, 2007). A system was designed using this technology to track and separably identify participants. To utilize this system, markers were attached to participants using mortar boards, or graduation caps, which were then read by cameras suspended above the experimental area. Power-over-Ethernet (POE) cameras, which only need one cable for power and communication, were used. The chosen POE camera is compact at 29 x 29 x 41 mm, but still affords a high resolution of 1280x1024 pixels at a maximum frame rate of 50 frames per second. Twelve cameras provided full coverage with overlap for the circuit experiments. A sample of a camera, encoded tracking pattern, and the camera gimbal can be found in Figure 4.
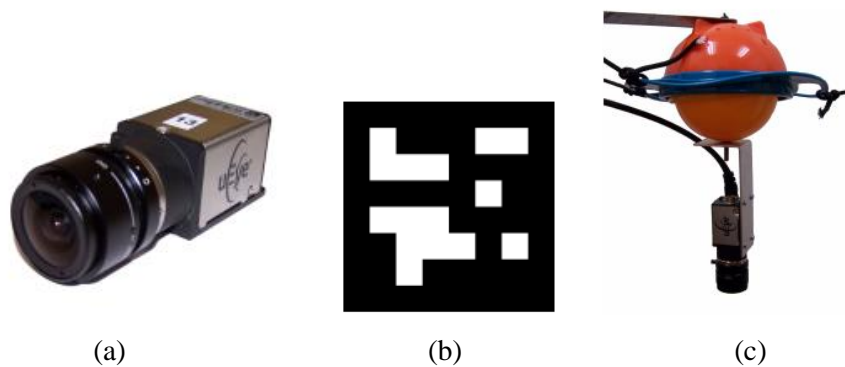

(a)   (b)   (c)

Figure 4. Tracking hardware: (a) Power-over-Ethernet (POE) camera, (b) encoded tracking pattern, and (c) camera gimbal.

4. Methodology

4.1. Instantaneous time headway definition

In vehicle traffic flow, the time headway is defined as the time that elapses between the arrival of the leading vehicle and following vehicle at a designated cross section. This concept can be slightly modified and extended for pedestrian flow. The first step is to define a personal space for each individual. This space determines a region surrounding each individual to specify pedestrian groups which potentially can have substantial effects on their walking behaviors. The personal space can be considered as a rectangular space defining the lateral and longitudinal boundaries. Considering the width of the human shoulders, body sway, and avoidance of contact with others, Fruin suggested minimum lateral space of 0.71 m (28 in) to 0.76 m (30 in), and 2.5 m (8 ft) to 3 m (10 ft) for lateral and longitudinal space, respectively (Fruin, 1971). In this study, the latitude personal space is assumed to be 0.71 m and the longitudinal personal space is considered to be 2.5 m. Two groups of pedestrians can have influence on a particular pedestrian's walking behavior: 1) Leader group and 2) Collider group. The Leader group is defined as the set of pedestrians who are effectively being followed by individuals. The Collider group is the set of pedestrians walking toward individuals. Figure 5 depicts the concept of personal space and illustrates the leader/collider definitions.

In this study, instantaneous time headway is proposed as a temporal distance measure between followers and leaders. Trajectory data makes it possible to differentiate leader and collider groups and compute the instantaneous time headway for each individual using the following basic relationship:

$$TH_f(t) = \frac{\left\| \overrightarrow{r_f(t)} - \overrightarrow{r_l(t)} \right\|}{\overrightarrow{v_f(t)}} \qquad \overrightarrow{v_f(t)} > 0 \tag{1}$$

Where $\overrightarrow{r_f(t)}$, $\overrightarrow{r_l(t)}$, and $\overrightarrow{v_f(t)}$ stand for follower position in time t, leader position in time t, and instantaneous follower speed in time t, respectively. The relationship implies that the instantaneous time headway for each time frame can be obtained by spacing between follower and leaders divided by the follower walking speed. Note that the definition is slightly different

than the time headway concept used in highway traffic. While time headway is directly measured in highway traffic at a specific location, the proposed method computes instantaneous time headway (temporal distances) by keeping track of follower and leader trajectories in each time frame.
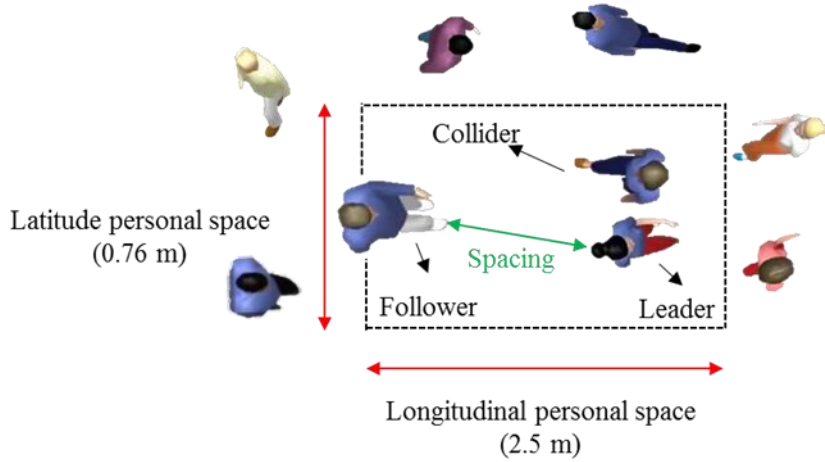
Figure 5. Personal space definition.

4.2. Time headway modeling

A large number of simple headway distribution models have been proposed in traffic flow studies. However, the main problem of the simple distributions is their inability to identify minimum or adequate time headways for capacity estimation purposes. Therefore, a mixed headway distribution model, distinguishing between unconstrained (or freely moving) and constrained (or following) time headways, was applied in this study. The Generalized Queuing Model (GQM), proposed by Cowan (1975) and Branston (1976), is a mixed probabilistic distribution model handling time headway as a sum of two mutually independent variables: the constrained and the free flowing headway. The general form of GQM can be defined as follow:

$$f(t) = \theta \times g(t) + (1 - \theta) \times h(t) \tag{2}$$

Where

$f(t)$ = time headway probability density function

θ = fraction of constrained time headways

$g(t)$ = probability density function of the constrained headway (empty zone distribution)

$h(t)$ = probability density function of the free flowing headway

Cowan (1975) derived the model, assuming that the empty zone distribution (constrained time headways) could be represented by Gamma distribution while free flowing time headways can be represented by Poisson distribution. The model is called Gamma-GQM.

$$f(t) = \theta \frac{\beta^{\alpha} t^{\alpha-1}}{\Gamma(\alpha)} e^{-\beta t} + (1-\theta) \frac{\beta^{\alpha}}{\Gamma(\alpha)} \lambda e^{-\lambda t} \int_{0}^{t} x^{\alpha-1} e^{-x(\beta-\lambda)} dx \qquad (3)$$

Where α, β, denote shape and scale parameters of Gamma distribution, respectively. λ stands for average arrival rate in Poisson distribution, and Γ is the gamma function. The parameters can be estimated using the Maximum Likelihood Estimation (MLE) method. The likelihood function of Gamma-GQM can be obtained using the following equation:

$$LL(t) = -n \ln \Gamma(\alpha) + \sum_{j=1}^{n} \ln[\theta(\beta t_j)^{\alpha-1} \beta e^{-\beta t_j} + (1-\theta)(\frac{\beta}{\beta-\lambda})^{\alpha} \frac{\gamma[\alpha, t(\beta-\lambda)]}{\Gamma(\alpha)} \lambda e^{-\lambda t}] \qquad (4)$$

$s.t. \qquad 0 \leq \theta \leq 1$

Where n stands for total number of observations, and γ represents incomplete gamma function. To examine the performance of the model, the Kolmogorov-Smirnov (K-S test) was used to measure the goodness of fit of the model to the observed time headway data. This test examines the null hypothesis that observed data is drawn from Gamma-GQM distribution quantifying the maximum distance between observed and Gamma-GQM cumulative distribution function.

The Gamma-GQM parameters can be used for capacity estimation purposes where capacity of a walking facility equals the inverse of minimum pedestrian time headways. The empty zone reflects the minimum time headway that a pedestrian adopts to follow the leaders. Therefore, capacity can be estimated by inversing the mean empty zone distribution, assuming that in capacity-flow conditions all pedestrians maintain constrained time headways respect to their leaders.

$$Cap = \frac{1}{W \times E(\text{X})} \qquad (5)$$

Where *W* and *E(X)* stand for average pedestrian lane width [m] and mean empty zone distribution [s], respectively. In fact, inverse of mean empty zone yields the capacity per pedestrian lane width unit and it can be converted to capacity per meter unit by dividing to pedestrian lane width. In the proposed method, time headway model can be separately calibrated for each leader type and impacts of different leader types on capacity estimation can be identified using corresponded empty zone distribution.

5. Data Processing

To control the large amount of collected trajectory data, a tool with database management and visualization capability was developed using MATLAB software. This user-friendly GUI is able to manage, process, and visualize the video data collected from the walking experiments. The developed GUI consists of three main components: visualization, processing, and behavioral data extraction. To visualize the experimental process, a simple CAD drawing of the study area was incorporated into the GUI. This map replicates pedestrian movements using their identification IDs during the experiments. The processing component makes it possible to extract the raw trajectory data for a selective area or selected time duration for all pedestrians or for a selective group of pedestrians (e.g. pedestrians with disabilities). In addition, microscopic behavioral variables (e.g., instantaneous speed and acceleration longitudinal and lateral spacing, time headway, orientation, local speed, flow and density) can be extracted using the GUI. The software is able to pull out the behavioral data for all pedestrians or for a particular target pedestrian. Figure 6 presents the GUI structure and components.
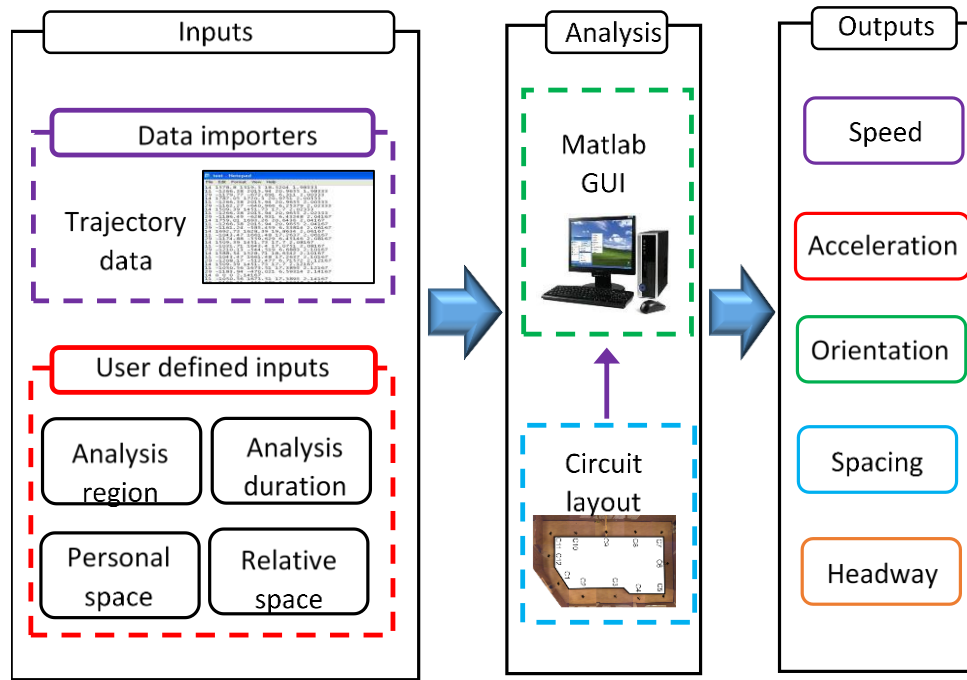
Figure 6. GUI structure.

Figure 7 shows a snapshot of the developed GUI. Detailed applications of the developed components including preview circuit map, toolbar, and analysis functions of the GUI are illustrated as follows.

Loading experimental data: After each experimental session, each of the 14 cameras was processed and 14 text files of the raw trajectory data were generated. These text files include IDs of each tracked participant and the positions (x, y, z) of tracked patterns in relation to the camera's center. Each file was named using the session time and camera number. To further process these raw data, the data needed to be loaded in the GUI. Data loading can be done by entering the session time and camera number into "Session time" and "Cam Num" fields respectively.
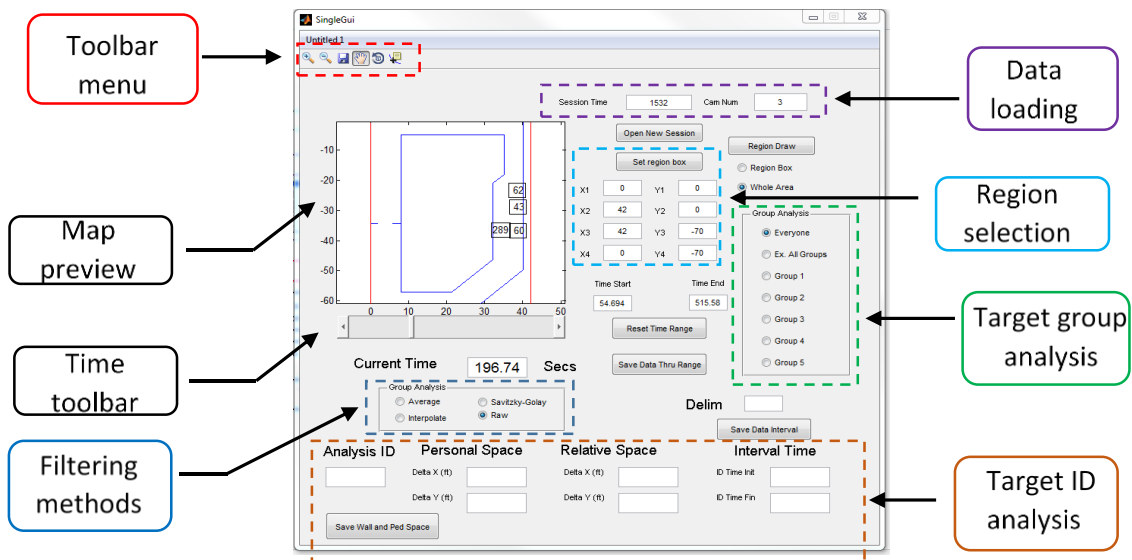
Figure 7. Snapshot of GUI.

Visualization: A preview of the circuit map was incorporated into the GUI to graphically observe pedestrian interactions during the experiments. The map shows positions of observed pedestrian IDs for selected time frames (current time field). Pedestrian movements and their interactions can be tracked by gradually increasing the time using the time bar.

Toolbar: The toolbar provides functions to adjust the circuit map. It allows users to view the circuit map closer or view more of the map by using the zoom in and zoom out buttons. In addition, the current view can be moved to any desired direction by using the pan button. The desired view can be saved using save button.

Study area and time duration selection: Defining proper spatial scales (i.e., area unit for computing density, speed and flow) is crucial in the processing procedures for obtaining reasonable results. The GUI makes it possible to process the raw trajectory data for a selective area and time duration. The desired area can be specified either by drawing the region on the map using the "Region draw" button or by inserting coordinates of the corners of selective area in the "Set region box". The selective time duration of data process can be determined by defining "Time Start" and "Time End".

Target group analysis: Sometimes it may be important to study on the behavior of a particular group (i.e. pedestrian with motorized wheelchair). Users can create up to five groups of

pedestrians using their IDs. The GUI can pull out and analyze the trajectory data for the target group for specified region and time duration. In addition, it is possible to smooth the walking trajectory data for each group by removing errors from the data set. The GUI provides different filtering procedures including average, interpolation, and Savitzky-Golay filtering method to smooth the data. Users can insert the desired time step into the "Delim" field and select the filtering method to obtain the trajectory data with less noise.

Target ID analysis: In addition to group analysis, the GUI is able to extract microscopic behavioral data for a particular pedestrian in a pre-defined time duration. This can be done by inserting the pedestrian ID and defining personal space, relative space, and interval time. The GUI reports the mean value of behavioral variables for the selected interval time.

6. Results and Discussions

This section presents the findings of the research. First, a descriptive analysis on observed follower speed and spacing distributions are presented to examine how these components contribute to time headways between followers and different leader types in passageway and bottleneck facilities. Then, the Gamma-GQM model estimation results are discussed. Finally, the capacity analysis of passageway and bottleneck is presented and the impacts of including different individuals with disabilities on capacity of facilities are discussed.

6.1. Descriptive data analysis

To explore the interactions between different pedestrian groups, follower speeds and spacing data were extracted for passageway and bottleneck facilities. These time headway components can clearly show how pedestrians changed their walking behavior with respect to their leaders. Figure 8 shows the observed distributions for different leader types and Table 2 presents some basic descriptive statistics including number of observations, mean, and standard deviation. In the Figure 8, the points on the horizontal surface show the joint observed distribution of spacing and follower speed distribution; the projected histograms show the observed marginal distributions. It can be seen that pedestrians behaved differently in different facilities and with respect to different leader types. By comparing observations for pedestrians who followed individuals without disability leaders in the passageway and bottleneck

formations, it can be shown that followers in passageway walked with higher mean speed and they maintained higher spacing respect to their leaders. Specifically, switching from passageway to bottleneck leads to a mean speed reduction from 1.19 m/s (3.9 ft/s) to 1.07 m/s (3.51 ft/s) (about 10% reduction) and mean spacing reduction from 1.52 m (4.99 ft) to 1.2 m (3.94 ft) (about 23% reduction).

Analysis revealed that leaders with varying disabilities changed their followers' behaviors differently. While visually impaired leaders had minimum impacts on followers' speed and spacing, individuals who use motorized wheelchair and individuals with mobility canes changed the followers' behaviors considerably in the passageway facility. Observations indicated that followers reduced their walking speed and kept substantially higher spacing when they had leaders with disabilities. Results showed that while follower mean speed reduced by 29% and 35%, mean spacing increased by 12% and 2% for leaders with mobility canes and leaders who use motorized wheelchairs, respectively. In the bottleneck facility, analysis showed that followers had marginal changing in spacing with respect to individuals who use ambulatory devices (e.g., non-motorized wheelchair/walker) leaders and they maintained higher spacing with respect to individuals using motorized wheelchairs and individuals with mobility canes. Specifically, the spacing between followers and leaders using motorized wheelchairs and with mobility canes was about 5% and 12% higher compared to individuals without disability leaders. Overall, the observations show that walking facility configurations and leader types may significantly change the follower speed and spacing. The next section examines the hypothesis that these behavior changes are effective on pedestrian time headways.

Passageway   Bottleneck

(a)

(b)

(c)

(d)

(e)
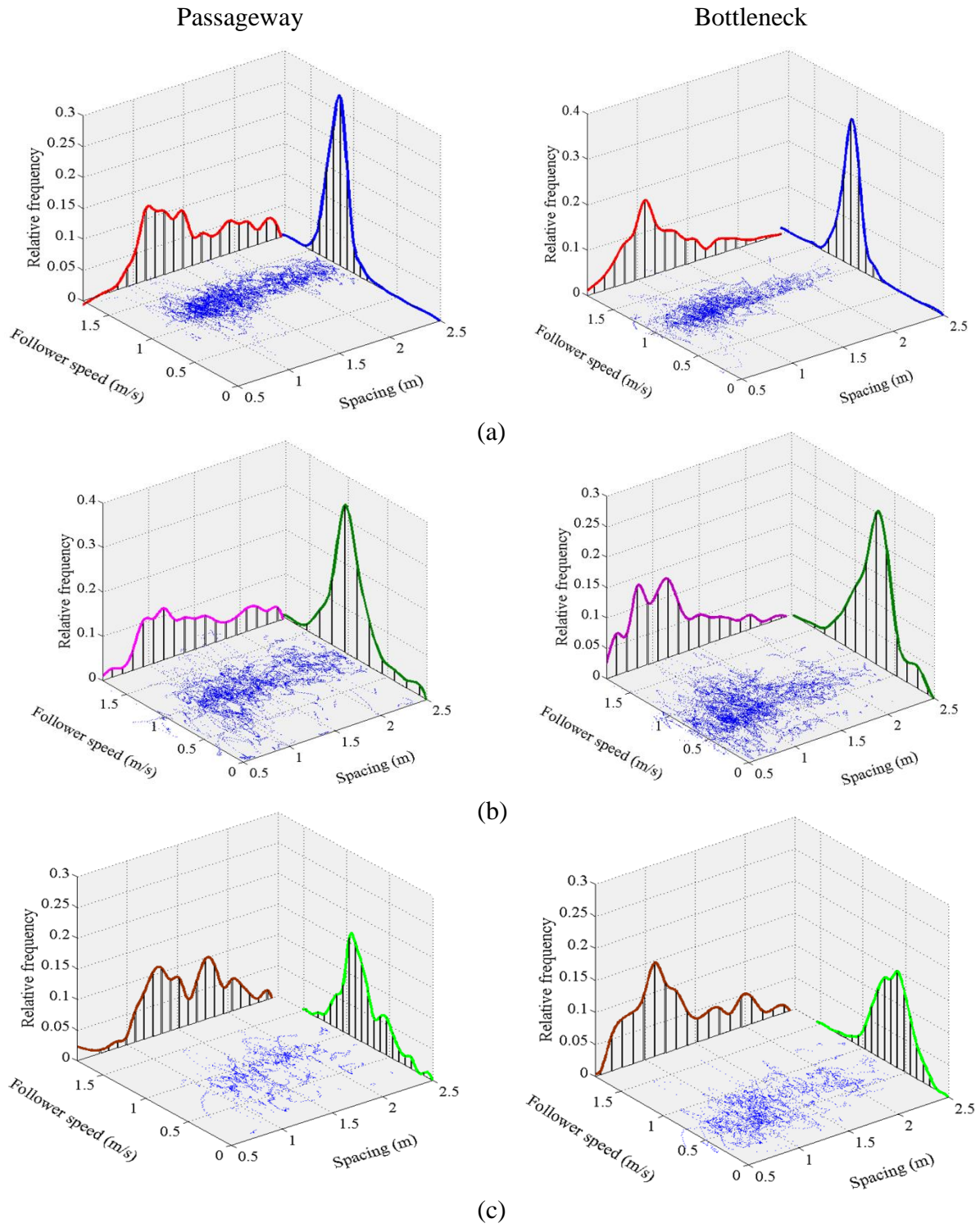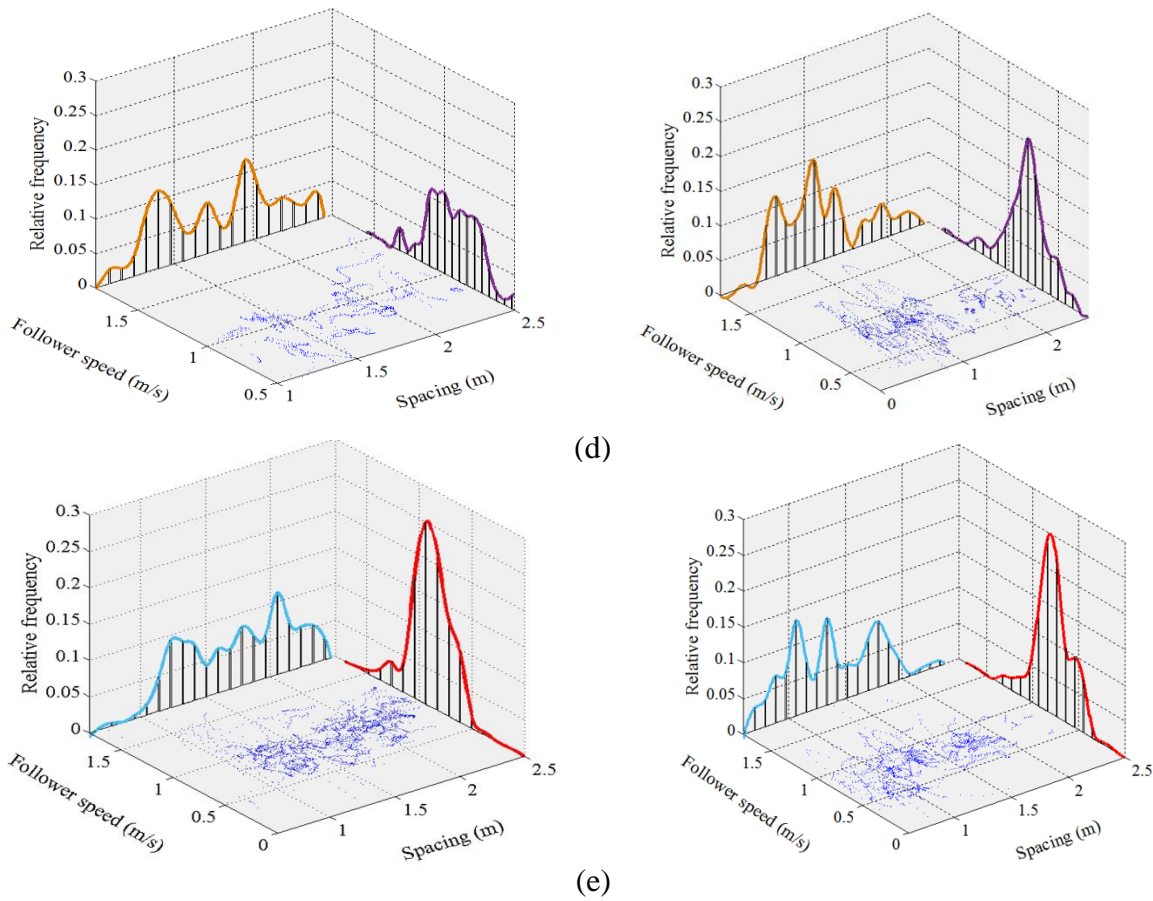
Figure 8. Observed follower speed and spacing distributions in passageway and bottleneck respect to different leader types. (a) individuals without disabilities, (b) visual impairment, (c) motorized wheelchair, (d) non-motorized wheelchair/walker, and (e) individuals with mobility canes.

Table 2. Followers' speed and spacing descriptive statistics.

| Facility | Leader type | N | Followers' speed | | Spacing | |
			Mean (m/s)	Std (m/s)	Mean (m)	Std (m)
Passageway	Without disabilities	10149	1.19	0.13	1.52	0.4
	Visual	9570	0.97	0.29	1.42	0.48
	Motorized	2409	0.83	0.23	1.56	0.35
	Non-motorized/walker	1134	0.93	0.17	1.74	0.35
	Cane	2452	0.89	0.2	1.72	0.39
Bottleneck	Without disabilities	6588	1.07	0.14	1.2	0.35
	Visual	10618	0.75	0.27	1.15	0.4
	Motorized	3864	0.65	0.2	1.26	0.43
	Non-motorized/walker	1902	0.81	0.21	1.19	0.48
	Cane	2015	0.78	0.21	1.35	0.37

6.2. Time Headway modeling

Parameters of the Gamma-GQM model need to be properly estimated before this model can be applied for capacity estimations. In this study, the maximum likelihood estimation method was used. The non-linear likelihood function presented in Eq. (4) was maximized using the Generalized Reduced Gradient (GRG) method (Lasdon et al., 1974) which is recognized as one of the most precise and accurate methods for solving nonlinear programming problems (Lee et al., 2004). Figure 9 presents the results of applying the estimation method to the passageway and bottleneck facilities only for individuals without disability leaders. The histograms show the observed time headway distribution with the bin size of 0.2 sec and the curves present the estimated model. The figures indicate that the model fitted to observed data well and it verifies that the model performance is acceptable for pedestrian time headway modeling. Observations revealed that a sharp peak can be identified in time headway ranges from 0.5 sec to 1.3 sec in the passageway facility but this peak is slightly sharper in the bottleneck facility. In other words, larger portions of pedestrians in the bottleneck followed their leaders in lower time headway ranges comparing to passageway. This observation indicates that the physical configuration of the walking environment can have a significant impact on time headway variable.

(a)   (b)
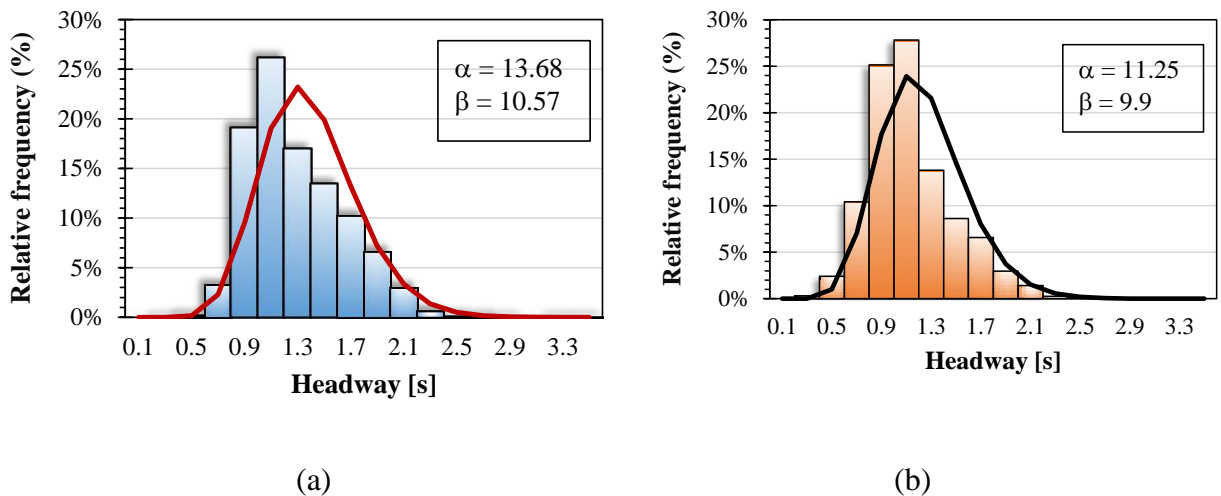
Figure 9. Results of headway estimations considering individuals without disabilities leader type for (a) passageway, and (b) bottleneck.

Table 3 summarizes the estimation results of the Gamma-GQM model for different facilities and different leader types. The columns of this table respectively presents the estimation results including number of observations (N), fraction of constrained time headways (θ), Gamma distribution shape parameter (α), Gamma distribution scale parameter (β), and *P*-values for the Kolmogorov-Smirnov test (K-S test). Estimation results reveal that all *P*-values for the K-S test are significantly higher than 0.05 indicating the K-S test null hypothesis cannot be rejected. In other words, the model provided a good fit to observed data for all cases. Analysis showed that the constrained time headway fractions (θ) are close to 1 for both the passageway and bottleneck as well as all leader types which is a plausible result considering dimensions of personal space. Note that the personal space for a follower pedestrian was defined as an area which other pedestrians impact on his/her movement and it is plausible to observe that the follower pedestrian is constrained by leaders in most of times. Therefore, the results verify that the initial personal space dimension choices were rational. Estimation results shows that there are significance differences in estimated model parameters for different facilities as well as different leader types. These differences support the hypothesis that time headways change significantly respect to facility configuration and leader types. Note that except in some cases, estimated shape and scale parameters do not reflect any straightforward interpretation from a traffic flow theory point of view.

Table 3. Summary of Gamma-GQM estimation results.

Facility	Leader type	N	θ	α	β	K-S test (p-value)
Passageway	Without disabilities	10149	1	13.68	10.57	0.95
	Visual	9570	0.97	4.6	3	0.74
	Motorized	2409	0.99	6.75	3.28	0.46
	Non-motorized/walker	1134	1	18.13	9.48	0.11
	Cane	2452	1	11.37	5.65	0.99
Bottleneck	Without disabilities	6588	1	11.25	9.9	0.95
	Visual	10618	0.98	3.86	2.28	0.74
	Motorized	3864	1	5.87	2.8	0.74
	Non-motorized/walker	1902	1	4.5	2.85	0.99
	Cane	2015	1	8.24	4.49	0.74

6.3. Capacity analysis

To estimate capacity, it is necessary to estimate the width of formed lanes (see Eq. 5). Video data showed that lane formation patterns are different when compared to passageway and bottleneck. The sketch in Figure 10 presents the walking patterns, expressing that pedestrians walked in staggered fashion in the passageway and used the available space optimally compared to the bottleneck. In other words, pedestrians need more lateral space in bottleneck formations due to space constraints. Analysis revealed that the average distance of lateral centers of tracked pedestrians were about 0.3 m and 0.21 m in the passageway and bottleneck, respectively. Therefore, lane width for passageway considered to be equal to distance of lateral centers (0.3 m) and personal latitude personal spacing (0.76 m) was considered for bottleneck accounting for additional safety and comfort distances. The number of lanes obtained from selected lane width for bottleneck (Number of lanes $= \dfrac{\text{Facility width}}{\text{Lane width}}$) was verified using video observations expressing that the assumption was a rational choice.
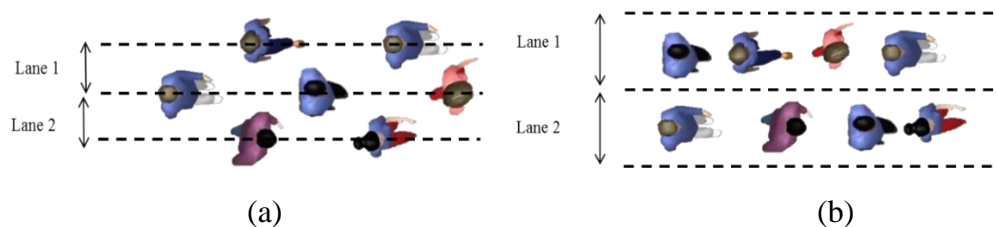
Figure 10. Lane formation pattern in (a) passageway, and (b) bottleneck

Empty zone parameters can be used to analyze the capacity of different facilities. The parameters also make it possible to explore the impacts of different leader types on walking facilities performance. The results of the estimations can be found in Table 4, which includes empty zone mean ($E(x)$) [s], capacity per pedestrian lane width (C) [ped/lane/s], and capacity per meter width (Cap) [ped/m/s] for all leader types. Results revealed that the empty zone mean of the passageway was much larger compared to the empty zone mean of the bottleneck for most of leader types. In comparing disabled leader types in the passageway and bottleneck formations, it is revealed that mean empty zone for the visually impaired was the lowest (1.53 sec and 1.69 sec,

respectively) and it was the highest for individuals using motorized wheelchairs (2.06 sec and 2.09 sec, respectively). This intuitively supports the observations that showed how followers changed their behavior with respect to these leader types in these two facilities.

Capacity estimation results for individuals without disability leaders showed the capacity of passageway and bottleneck is 2.57, and 1.17 ped/m/s, respectively. The finding can be explained by the ability of pedestrians to maneuver in these two environments. In the bottleneck formation, pedestrians are have much less space due to the configuration of the bottleneck facility. Results also revealed that all types of individuals with disabilities leaders had reduction impacts on the capacity of facilities. Visually impaired individuals and individuals using motorized wheelchairs had the minimum (about 17% reduction) and maximum impact (about 46% reduction) on the passageway capacity. This result can be interpreted by a number of factors. Sharifi et al. (2015) showed that among individuals with disability groups, these two groups had the highest and lowest speeds in passageway, respectively. It indicates that followers of individuals using motorized wheelchairs needed to adjust their speed considerably compared to visually impaired leaders. On the other hand, video records showed that followers were more conservative in keeping a safe distance from individuals using motorized wheelchairs. Therefore, these two factors caused a considerable time headway change and led to a substantial capacity reduction. The same conclusion applies for the bottleneck. While the visually impaired and individuals with non-motorized wheelchair/walkers showed the minimum effects, individuals with mobility canes and motorized wheelchairs showed the highest effects on bottleneck capacity reduction. However, reduction effects are not similar for different leader types. For instance, visually impaired leaders reduced the capacity of the passageway and bottleneck by 17% and 39%, respectively. But, these reduction effects are 44% and 48% for individuals using motorized wheelchair. This finding can be explained by comparing walking behaviors in these two facilities. Pedestrian maneuverability is substantially constrained by individuals with motorized wheelchairs in both passageway and bottleneck facilities but observations showed that followers could walk much faster behind visually impaired leaders in passageway comparing to bottleneck. Therefore reduction effects of visually impaired leaders are more visible in the bottleneck.

Table 4. Mean empty zone and capacity estimation results.

Facility	Leader type	E(x)	C	Cap
Passageway	Without disabilities	1.29	0.77	2.57
	Visual	1.53	0.65	2.16
	Motorized	2.06	0.48	1.6
	Non-motorized/walker	1.91	0.52	1.73
	Cane	2.01	0.49	1.63
Bottleneck	Without disabilities	1.14	0.88	1.17
	Visual	1.69	0.59	0.79
	Motorized	2.09	0.48	0.64
	Non-motorized/walker	1.58	0.63	0.84
	Cane	1.83	0.54	0.72

7. Summary and Conclusion

To effectively design walking facility infrastructures, it is imperative for planners and designers to have an understanding of the interactions of different pedestrian groups. Unfortunately, current design guidelines overlook heterogeneity in pedestrian composition and there are a limited number of studies examining the behaviors of heterogeneous pedestrian streams containing individuals with disabilities. The purpose of this research was to explore interactions between different pedestrian groups. To this end, large-scale controlled walking experiments involving a heterogeneous mixture of individuals without and with disabilities were conducted and accurate trajectory data were collected. From this data, this paper presented a statistical framework to model time headway distributions for one-directional heterogeneous pedestrian streams. Specifically, a mixed time headway distribution model was applied to collected data and results were discussed for passageway and bottleneck facilities. Capacity of these facilities were estimated and impacts of including individuals with disabilities on capacity were investigated. Analysis of the collected data suggested the following key findings:

- Gamma-GQM model had a good performance for pedestrian time headway modeling.

- Observations showed that followers changed their speed and spacing in passageway and bottleneck as well as respect to different leader types.

- The time headway model parameters supported the hypothesis that behavior changes are statistically significant.

- Capacity of the passageway was about 75% higher than the bottleneck considering individuals without disability as leaders.

- All types of individuals with disabilities impacted the capacity of the passageway and bottleneck.

- Visually impaired individuals and individuals using motorized wheelchairs had the minimum and maximum capacity reduction effect on passageway and bottleneck.

The findings are expected to improve the estimation of facility capacities, which is required to meet a preferred level-of-service for heterogeneous populations. Furthermore, the results of the research may be used to improve best practices for the design of new built environments and help public policy officials make better informed development decisions.

8. References

ADA (Americans with Disabilities Act), 1990. U.S. Department of Labor, Washington, DC.

ADAAG (Americans with disabilities Act Accessibility Guidelines), 2002. ADA accessibility guidelines for buildings and facilities. U.S. Department of Justice, Washington, DC.

Ando, K., Ota, H., Oki, T., 1988. Forecasting the flow of people. Railway Research Review 45 (8), 8-14. (in Japanese).

Branston, D., 1976. Models of single lane time headway distributions. Transportation Science 10, 125-148.

Burghardt, S., Seyfried, A., Klingsch, W., 2013. Performance of stairs-Fundamental diagram and topographical measurements. Transportation Research Part C 37, 268-278.

Cowan, R. J., 1975. Useful headway models. Transportation Research 9, 371-375.

Daamen, W., Hoogendoorn, S.P., 2011. Emergency door capacity: Influence of door width, population composition and stress level. Fire technology 48 (1), 55-71.

Daamen, W., Hoogendoorn, S.P., 2003. Controlled experiments to drive walking behavior. European Journal of Transport and Infrastructure Research 3 (1), 39-59.

Daly, P.N., McGrath, F., Annesley, T.J., 1991. Pedestrian speed/flow relationships for underground stations. Traffic Engineering and Control 32 (2), 75-78.

Fruin, J. J., 1987. Pedestrian planning and design. Elevator world, Mobile, Ala.

Helbing, D., Johansson, A., Al-Abideen., H., 2007. Dynamics of crowd disasters: An empirical study. Physical Review E 75 (4), 1-7.

Hoogendoorn, S.P., 2004. Walking behavior in bottlenecks and its implications for capacity. Proceeding of Annual Meeting of the Transportation Research Board, Washington, D.C.

Hoogendoorn, S. P., Bovy, P., 1998. New estimation technique for vehicle-type-specific headway distributions. Transportation Research Record 1646, 18-28.

ICC (International Code Council), 2012. International building code. Washington, DC.

Khisty, C.J., 1985. Pedestrian flow characteristics on stairways during disaster evacuation. Transportation Research Record 1047, 97-102.

Kretz, T., Grunebohm, A., Kaufman, M., Mazur, F., Schreckenberg, M., 2006 (a). Experimental study of pedestrian counterflow in a corridor. Journal of Statistical Mechanics P10001.

Kretz, T., Grunebohm, A., Schreckenberg, M., 2006 (b). Experimental study of pedestrian flow through a bottleneck. Journal of Statistical Mechanics P06004.

Lam, W.H.K., Cheung, C.Y., 2000. Pedestrian speed/flow relationships for walking facilities in Hong Kong. Journal of Transportation Engineering 126, 343-349.

Lasdon, L.S., Fox, R.L., Ratner, M.W., 1974. Nonlinear optimization using the generalized reduced gradient method. Operations Research 8 (3), 73-103.

Lee, H., Chen, S., Kang, H., 2004. A study of generalized reduced gradient method with different search directions. Journal of Quantitative Management 1 (1), 25-38.

Navin, P.D., Wheeler, R.J., 1969. Pedestrian flow characteristics. Traffic Engineering 39 (9), 30-36.

Oeding, D., 1963. Traffic loads and dimensions of walkways and other pedestrian circulation facilities. Strassembau and Strassenverkehrstenchnik 22.

Older, S. J., 1968. Movement of pedestrians on footways in shopping streets. Traffic Engineering and Control 10, 160-163.

Polus, A., Schafer, J. L., Ushpiz, A., 1983. Pedestrian flow and level of service. Journal of Transportation Engineering 109, 46-57.

Pushkarev, B., Zupan, J.M., 1975. Urban space for pedestrians. Massachusetts Institute of Technology Press, Cambridge, MA.

Seyfried, A., Steffen, B., Klingsch, W., Boltes, M., 2005. The fundamental diagram of pedestrian movement revisited. Journal of Statistical Mechanics P100002.

Seyfried, A., Passon, O., Bernhard, S., Boltes, M., Rupprecht, T., Klingsch, W., 2009. New insights into pedestrian flow through bottlenecks. Transportation Science 43 (3), pp. 395-406.

Sharifi, M.S., Stuart, D., Christensen, K.M., Chen, A., Kim, Y., Chen, Y., 2015. Analysis of walking speeds involving individuals with disabilities in different indoor walking environments. Journal of Urban Planning and Development. **10.1061/(ASCE)UP.1943-5444.0000288**.

Tanaboriboon, Y., Hwa, Chor, C.H., 1986. Pedestrian characteristics study in Singapore. Journal of Transportation Engineering 112 (3), 229-235.

Tanaboriboon, Y., Guyano, J.A., 1991. Analysis of pedestrian movements in Bangkok. Transportation Research Record 1294, pp. 52-56.

TRB (Transportation Research Board), 2010. Highway Capacity Manual, Washington, DC.

U.S. Census Bureau. American Community Survey, 2010. http://www.census.gov/prod/2012pubs/p70-131.pdf. Accessed Jul. 24, 2015.

Virkler, M.R., Elayadath, S., 1994. Pedestrian speed-flow-density relationships. Transportation Research Record 1438, 51-58.

Wong, S.C., Leung, W., Chan, S., Lam, W., Yung, N., Liu, C., Zhang, P., 2010. Bidirectional pedestrian stream model with oblique intersecting angle. Journal of. Transportation. Engineering 136 (3), 2010, 234-242.

Ye, J., Chen, X., Yang, C., Wu, J., 2008. Walking Behavior and pedestrian flow characteristics for different types of walking facilities. Transportation Research Record 2048, 43-51.

Zhang, J., Klingsch, W., Schadschneider, A., Seyfried, A., 2011. Transitions in pedestrian fundamental diagrams of straight corridors and T-junctions. Journal of Statistical Mechanics P06004.

9. APPENDIX: GUI Code

```matlab
function varargout = SingleGui(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @SingleGui_OpeningFcn, ...
                   'gui_OutputFcn',  @SingleGui_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before SingleGui is made visible.
function SingleGui_OpeningFcn(hObject, eventdata, handles, varargin)
global x1; global x2; global x3; global x4; global y1; global y2; global y3;
global y4;
global RegionBoxX;
global RegionBoxY;
global Doorlx;
global Doorly;
global Doorrx;
global Doorry;
```

```matlab
global outsidex;
global outsidey;
global insidex;
global insidey;
global FinalCameraMatrix;
global CameraMatrixNew;
global SessionTime;
global ExpTime;
global TStart;
global TEnd;
global CurTime;
global Average; global Inter; global SmthFilt; global raw; global delim;
delim = 1;
Average = 0;
Inter = 0;
SmthFilt = 0;
raw = 1;
CurTime = 0;
TStart = 0;
TEnd = 1000;
global RegionBoxFlag; global WholeBoxFlag; global EveryoneFlag; global
ExEveryoneFlag; global Group1Flag; global Group2Flag; global Group3Flag;
global Group4Flag; global Group5Flag;
RegionBoxFlag = 0; WholeBoxFlag =1; EveryoneFlag = 1; ExEveryoneFlag = 0;
Group1Flag = 0; Group2Flag = 0; Group3Flag = 0; Group4Flag = 0; Group5Flag=0;
%Set regionbox internal variables
x1 = '0'; x2='42';x3='42';x4='0';y1='0';y2='0';y3='-70';y4='-70';
%Set intially set radiobutton defaults and edit button defaults
set(handles.RegionBox,'Value',0);
set(handles.WholeBox,'Value',1);
set(handles.Everyone,'Value',1);
set(handles.ExEveryone,'Value',0);
set(handles.Group1,'Value',0);
set(handles.Group2,'Value',0);
set(handles.Group3,'Value',0);
set(handles.Group4,'Value',0);
set(handles.Group5,'Value',0);
set(handles.X1,'String',num2str(x1))
set(handles.X2,'String',num2str(x2))
set(handles.X3,'String',num2str(x3))
set(handles.X4,'String',num2str(x4))
set(handles.Y1,'String',num2str(y1))
set(handles.Y2,'String',num2str(y2))
set(handles.Y3,'String',num2str(y3))
set(handles.Y4,'String',num2str(y4))
set(handles.SessionTime,'String',num2str(0))
set(handles.TStart,'String',num2str(TStart))
set(handles.TEnd,'String',num2str(TEnd))
set(handles.CurrentTime,'String',num2str(CurTime))

%Turn off all buttons but open button so that user can't do anything until
%file is opened for first time.
set(handles.RegionBox,'Enable','off');
set(handles.WholeBox,'Enable','off');
set(handles.Everyone,'Enable','off');
```

```matlab
set(handles.ExEveryone,'Enable','off');
set(handles.Group1,'Enable','off');
set(handles.Group2,'Enable','off');
set(handles.Group3,'Enable','off');
set(handles.Group4,'Enable','off');
set(handles.Group5,'Enable','off');
set(handles.X1,'Enable','off');
set(handles.X2,'Enable','off');
set(handles.X3,'Enable','off');
set(handles.X4,'Enable','off');
set(handles.Y1,'Enable','off');
set(handles.Y2,'Enable','off');
set(handles.Y3,'Enable','off');
set(handles.Y4,'Enable','off');
set(handles.TStart,'Enable','off');
set(handles.TEnd,'Enable','off');
set(handles.CurrentTime,'Enable','off');
set(handles.TimeSlider,'Enable','off');
set(handles.SetRegion,'Enable','off');
set(handles.ResetTime,'Enable','off');
set(handles.SaveData,'Enable','off');
set(handles.RegionDraw,'Enable','off');


RegionBoxX = [0 42 42 0 0];


RegionBoxY = [0 0 -70 -70 0];


%This specified corners for whole circuit.
 Doorlx = [0;2.11];
Doorly = [-34.34;-34.34];
Doorrx = [5.86;8];
Doorry = [-34.34;-34.34];
outsidex = [0;40.2010;40.2010;27.7139;0;0];
outsidey = [0;0;-49.6270;-62.0889;-62.0889;0];


insidex = [8;35.2;35.2;32.2;32.2;21.46;8;8];
insidey = [-5;-5;-18;-20.9;-46.31;-57.03;-57.03;-5];
%Start out by displaying whole circuit.
plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
hold on;
plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
hold off;
% Choose default command line output for SingleGui
handles.output = hObject;


% Update handles structure
guidata(hObject, handles);


function varargout = SingleGui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```matlab
% Get default command line output from handles structure
varargout{1} = handles.output;



function SessionTime_Callback(hObject, eventdata, handles)
global SessionTime;
SessionTime = get(hObject,'String'); %Set intial value for opening first file

% --- Executes during object creation, after setting all properties.
function SessionTime_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


  function X1_Callback(hObject, eventdata, handles)
global x1;
%Gets x1 edit box value and sets radio buttons toggled appropriately
x1 =get(hObject,'String');
set(handles.WholeBox,'Value',0);
 set(handles.RegionBox,'Value',1);

guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function X1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Y1_Callback(hObject, eventdata, handles)
global y1;
%Gets y1 edit box value and sets radio buttons toggled appropriately
y1 =get(hObject,'String');
set(handles.WholeBox,'Value',0);
 set(handles.RegionBox,'Value',1);

guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function Y1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

 function X3_Callback(hObject, eventdata, handles)
global x3;
```

```matlab
%Gets x3 edit box value and sets radio buttons toggled appropriately
x3 =get(hObject,'String');
set(handles.WholeBox,'Value',0);
 set(handles.RegionBox,'Value',1);

guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function X3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function Y3_Callback(hObject, eventdata, handles)
global y3;
%Gets y3 edit box value and sets radio buttons toggled appropriately
y3 =get(hObject,'String');
set(handles.WholeBox,'Value',0);
 set(handles.RegionBox,'Value',1);

guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function Y3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


 function X2_Callback(hObject, eventdata, handles)
global x2;
%Gets x2 edit box value and sets radio buttons toggled appropriately
x2 =get(hObject,'String');
set(handles.WholeBox,'Value',0);
 set(handles.RegionBox,'Value',1);

guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function X2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function Y2_Callback(hObject, eventdata, handles)
global y2;
%Gets y2 edit box value and sets radio buttons toggled appropriately
y2 =get(hObject,'String');
```

```matlab
set(handles.WholeBox,'Value',0);
 set(handles.RegionBox,'Value',1);

guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function Y2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Y2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function X4_Callback(hObject, eventdata, handles)
global x4;
%Gets x4 edit box value and sets radio buttons toggled appropriately
x4 =get(hObject,'String');
set(handles.WholeBox,'Value',0);
 set(handles.RegionBox,'Value',1);

guidata(hObject, handles);

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function Y4_Callback(hObject, eventdata, handles)
global y4;
%Gets y4 edit box value and sets radio buttons toggled appropriately
y4 =get(hObject,'String');
set(handles.WholeBox,'Value',0);
 set(handles.RegionBox,'Value',1);
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function Y4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

  function TStart_Callback(hObject, eventdata, handles)
TStart = str2double(get(hObject,'String')); %Grab entered string for Tstart
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function TStart_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```matlab
    set(hObject,'BackgroundColor','white');
end


function TEnd_Callback(hObject, eventdata, handles)
global TEnd;
TEnd=str2double(get(hObject,'String'));%Grab entered string for TEnd
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function TEnd_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
function TimeSlider_Callback(hObject, eventdata, handles)
global CurTime; global TStart; global TEnd; global ExpTime; global
timeindex;global CameraMatrixNew;
global RegionBoxX;
global RegionBoxY;
global Doorlx;
global Doorly;
global Doorrx;
global Doorry;
global outsidex;
global outsidey;
global insidex;
global insidey;
global xmin; global xmax; global ymin; global ymax;
global CamEver; global CamEx; global CamG1; global CamG2; global CamG3;
global CamG4; global CamG5;
global EveryoneFlag; global ExEveryoneFlag; global Group1Flag; global
Group2Flag; global Group3Flag; global Group4Flag; global Group5Flag;
timeindex = round(get(hObject,'Value')); %Grab slider value and round as
indexes to arrays must be integers
CurTime = ExpTime(timeindex); %Find corresponding time value for integer
index
set(handles.CurrentTime,'String',num2str(CurTime)) %Display what array time
is

 if (EveryoneFlag) %Each conditional is for corresponding radio box value,
    index =
(CamEver(:,3)>round(ExpTime(timeindex)*100)/100)&(CamEver(:,3)<round(ExpTime(
timeindex)*100)/100+0.03);
    Sugar = CamEver(index,:);
    %Display points over region box and circuit
   plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
   axis([xmin xmax ymin ymax]); %This changes the figure display to be around
the region of the specific camera.
    hold on;
```

```matlab
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 elseif(ExEveryoneFlag)
     index =
(CamEx(:,3)>round(ExpTime(timeindex)*100)/100)&(CamEx(:,3)<round(ExpTime(time
index)*100)/100+0.03);
    Sugar = CamEx(index,:);
    plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
    axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 elseif(Group1Flag)
     index =
(CamG1(:,3)>round(ExpTime(timeindex)*100)/100)&(CamG1(:,3)<round(ExpTime(time
index)*100)/100+0.03);
    Sugar = CamG1(index,:);
    plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
    axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 elseif(Group2Flag)
     index =
(CamG2(:,3)>round(ExpTime(timeindex)*100)/100)&(CamG2(:,3)<round(ExpTime(time
index)*100)/100+0.03);
    Sugar = CamG2(index,:);
    plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
    axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 elseif(Group3Flag)
```

```matlab
    index =
(CamG3(:,3)>round(ExpTime(timeindex)*100)/100)&(CamG3(:,3)<round(ExpTime(time
index)*100)/100+0.03);
    Sugar = CamG3(index,:);
    plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
    axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 elseif(Group4Flag)
    index =
(CamG4(:,3)>round(ExpTime(timeindex)*100)/100)&(CamG4(:,3)<round(ExpTime(time
index)*100)/100+0.03);
    Sugar = CamG4(index,:);
    plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
    axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 elseif(Group5Flag)
    index =
(CamG5(:,3)>round(ExpTime(timeindex)*100)/100)&(CamG5(:,3)<round(ExpTime(time
index)*100)/100+0.03);
    Sugar = CamG5(index,:);
    plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
    axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 end


 % --- Executes during object creation, after setting all properties.
function TimeSlider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to TimeSlider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
```

```matlab
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


 function CurrentTime_Callback(hObject, eventdata, handles)
CurTime = str2double(get(hObject,'String')); %Grab user entered current time



% --- Executes during object creation, after setting all properties.
function CurrentTime_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

 % --- Executes on button press in WholeBox.
function WholeBox_Callback(hObject, eventdata, handles)
global WholeBoxFlag;global x1; global x2; global x3; global x4; global y1;
global y2; global y3; global y4;
if (get(hObject,'Value'))
   set(handles.RegionBox,'Value',0);
   WholeBoxFlag=1;
else
    set(handles.RegionBox,'Value',1);
    WholeBoxFlag=0;
end
x1 = '0'; x2='42';x3='42';x4='0';y1='0';y2='0';y3='-70';y4='-70';
set(handles.X1,'String',num2str(x1))
set(handles.X2,'String',num2str(x2))
set(handles.X3,'String',num2str(x3))
set(handles.X4,'String',num2str(x4))
set(handles.Y1,'String',num2str(y1))
set(handles.Y2,'String',num2str(y2))
set(handles.Y3,'String',num2str(y3))
set(handles.Y4,'String',num2str(y4))




% --- Executes on button press in RegionBox.
function RegionBox_Callback(hObject, eventdata, handles)
% hObject    handle to RegionBox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of RegionBox
global RegionBoxFlag;
%Toggle raido buttons and set flag for region box.
if (get(hObject,'Value'))
   set(handles.WholeBox,'Value',0);
   RegionBoxFlag = 1;
else
    set(handles.WholeBox,'Value',1);
    RegionBoxFlag = 0;
```

```matlab
end

 % --- Executes during object creation, after setting all properties.
function RegionBox_CreateFcn(hObject, eventdata, handles)


function Untitled_1_Callback(hObject, eventdata, handles)
function Everyone_ButtonDownFcn(hObject, eventdata, handles)


% --- Executes on key press with focus on X1 and none of its controls.
function X1_KeyPressFcn(hObject, eventdata, handles)


function SetRegion_ButtonDownFcn(hObject, eventdata, handles)

% --- Executes on button press in SetRegion.
function SetRegion_Callback(hObject, eventdata, handles)
% hObject    handle to SetRegion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global CurTime; global TStart; global TEnd; global ExpTime; global
timeindex;global CameraMatrixNew;
global RegionBoxX;
global RegionBoxY;
global Doorlx;
global Doorly;
global Doorrx;
global Doorry;
global outsidex;
global outsidey;
global insidex;
global insidey;
global xmin; global xmax; global ymin; global ymax;
global CamEver; global CamEx; global CamG1; global CamG2; global CamG3;
global CamG4; global CamG5;
global EveryoneFlag; global ExEveryoneFlag; global Group1Flag; global
Group2Flag; global Group3Flag; global Group4Flag; global Group5Flag;
global g1; global g2; global g3; global g4; global g5;
global x1; global x2; global x3; global x4; global y1; global y2; global y3;
global y4;
%Grab values eneterd for region box.
RegionBoxX(1) = str2double(x1);
RegionBoxX(5) = str2double(x1);
RegionBoxX(2) = str2double(x2);
RegionBoxX(3) = str2double(x3);
RegionBoxX(4) = str2double(x4);
RegionBoxY(1) = str2double(y1);
RegionBoxY(5) = str2double(y1);
RegionBoxY(2) = str2double(y2);
RegionBoxY(3) = str2double(y3);
RegionBoxY(4) = str2double(y4);
%This stuff is the same plotting alg as above, but this will display the
%changed values of the region box.
```

```matlab
 if (EveryoneFlag)
    index =
(CamEver(:,3)>round(CurTime*100)/100)&(CamEver(:,3)<round(CurTime*100)/100+0.
02);
    Sugar = CamEver(index,:);
    plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
    axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 elseif(ExEveryoneFlag)
     index =
(CamEx(:,3)>round(CurTime*100)/100)&(CamEx(:,3)<round(CurTime*100)/100+0.02);
    Sugar = CamEx(index,:);
   plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
   axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 elseif(Group1Flag)
     index =
(CamG1(:,3)>round(CurTime*100)/100)&(CamG1(:,3)<round(CurTime*100)/100+0.02);
    Sugar = CamG1(index,:);
    plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
    axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 elseif(Group2Flag)
     index =
(CamG2(:,3)>round(CurTime*100)/100)&(CamG2(:,3)<round(CurTime*100)/100+0.02);
    Sugar = CamG2(index,:);
    plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
    axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
```

```matlab
    hold off;
 elseif(Group3Flag)
     index =
(CamG3(:,3)>round(CurTime*100)/100)&(CamG3(:,3)<round(CurTime*100)/100+0.02);
    Sugar = CamG3(index,:);
    plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
    axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 elseif(Group4Flag)
     index =
(CamG4(:,3)>round(CurTime*100)/100)&(CamG4(:,3)<round(CurTime*100)/100+0.02);
    Sugar = CamG4(index,:);
    plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
    axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 elseif(Group5Flag)
     index =
(CamG5(:,3)>round(CurTime*100)/100)&(CamG5(:,3)<round(CurTime*100)/100+0.02);
    Sugar = CamG5(index,:);
    plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
    axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 end


function uipanel2_SelectionChangeFcn(hObject, eventdata, handles)
global EveryoneFlag; global ExEveryoneFlag; global Group1Flag; global
Group2Flag; global Group3Flag; global Group4Flag; global Group5Flag;
switch get(eventdata.NewValue,'Tag') % Get Tag of selected object.
    case 'Everyone'
        % Code for when radiobutton1 is selected.
        EveryoneFlag = 1;
        ExEveryoneFlag = 0;
        Group1Flag = 0;
```

```
        Group2Flag = 0;
        Group3Flag = 0;
        Group4Flag = 0;
        Group5Flag = 0;
    case 'ExEveryone'
        % Code for when radiobutton2 is selected.
        EveryoneFlag = 0;
        ExEveryoneFlag = 1;
        Group1Flag = 0;
        Group2Flag = 0;
        Group3Flag = 0;
        Group4Flag = 0;
        Group5Flag = 0;
    case 'Group1'
        % Code for when togglebutton1 is selected.
        EveryoneFlag = 0;
        ExEveryoneFlag = 0;
        Group1Flag = 1;
        Group2Flag = 0;
        Group3Flag = 0;
        Group4Flag = 0;
        Group5Flag = 0;
    case 'Group2'
        % Code for when togglebutton2 is selected.
        EveryoneFlag = 0;
        ExEveryoneFlag = 0;
        Group1Flag = 0;
        Group2Flag = 1;
        Group3Flag = 0;
        Group4Flag = 0;
        Group5Flag = 0;
    case 'Group3'
        % Code for when togglebutton2 is selected.
        EveryoneFlag = 0;
        ExEveryoneFlag = 0;
        Group1Flag = 0;
        Group2Flag = 0;
        Group3Flag = 1;
        Group4Flag = 0;
        Group5Flag = 0;
    case 'Group4'
        % Code for when togglebutton2 is selected.
        EveryoneFlag = 0;
        ExEveryoneFlag = 0;
        Group1Flag = 0;
        Group2Flag = 0;
        Group3Flag = 0;
        Group4Flag = 1;
        Group5Flag = 0;
    case 'Group5'
        % Code for when togglebutton2 is selected.
        EveryoneFlag = 0;
        ExEveryoneFlag = 0;
        Group1Flag = 0;
        Group2Flag = 0;
        Group3Flag = 0;
```

```matlab
            Group4Flag = 0;
            Group5Flag = 1;
    % Continue with more cases as necessary.
    otherwise
            a=0
            % Code for when there is no match.
end


% --- Executes on button press in OpenSession.
function OpenSession_Callback(hObject, eventdata, handles)
global CameraMatrixNew; global SessionTime; global ExpTime; global CamNum;
global xmin; global xmax; global ymin; global ymax;
global TStart; global TEnd; global CurTime;
global RegionBoxX;
global RegionBoxY;
global Doorlx;
global Doorly;
global Doorrx;
global Doorry;
global outsidex;
global outsidey;
global insidex;
global insidey;
global g1; global g2; global g3; global g4; global g5;
global CamEver; global CamEx; global CamG1; global CamG2; global CamG3;
global CamG4; global CamG5;
global EveryoneFlag; global ExEveryoneFlag; global Group1Flag; global
Group2Flag; global Group3Flag; global Group4Flag; global Group5Flag;

%Hardcoded position of cameras
Pose = [3.2308333e+001 -5.0408333e+001; 3.5836450e+001 -4.0200000e+001;...
    3.5519700e+001 -3.1500000e+001; 3.6628125e+001 -1.8500000e+001;...
    3.5236400e+001 -4.8160000e+000; 2.0708330e+001 -3.7500000e+000;...
    7.2916600e+000 -3.2080000e+000; 7.4666600e+000 -2.2416660e+001;...
    5.1500000e+000 -3.4333333e+001; 5.5666660e+000 -4.8833333e+001;...
    7.9833300e+000 -6.0161458e+001; 2.1616660e+001 -5.8986458e+001];

%Do Zoom here to Zoom in based on Camera Number
xmin = Pose(str2double(CamNum),1)-15;
xmax = Pose(str2double(CamNum),1)+15;
ymin = Pose(str2double(CamNum),2)-15;
ymax = Pose(str2double(CamNum),2)+15;


try %Try to see if the following code works, otherwise catch and display
error
    %Open individual text file, based on session time and camera
CameraMatrixNew = load(strcat(SessionTime,'_',CamNum,'.txt'));
ExpTime = unique(CameraMatrixNew(:,3)); % Create time array from time column
TStart = min(ExpTime); set(handles.TStart,'String',num2str(TStart)) %Set
start time
TEnd = max(ExpTime);set(handles.TEnd,'String',num2str(TEnd)) %Set end time
CurTime = min(ExpTime);set(handles.CurrentTime,'String',num2str(CurTime))
%Set current time to beginning
length(ExpTime)
```

```matlab
%Change defaults of displayed data and turn on all other buttons.
set(handles.TimeSlider,'Value',1);
set(handles.TimeSlider,'Min',1);
set(handles.TimeSlider,'Max',length(ExpTime));
set(handles.TimeSlider,'SliderStep',([1, 1] / (length(ExpTime) - 1)));

set(handles.RegionBox,'Enable','on');
set(handles.WholeBox,'Enable','on');
set(handles.Everyone,'Enable','on');
set(handles.ExEveryone,'Enable','on');
set(handles.Group1,'Enable','on');
set(handles.Group2,'Enable','on');
set(handles.Group3,'Enable','on');
set(handles.Group4,'Enable','on');
set(handles.Group5,'Enable','on');
set(handles.X1,'Enable','on');
set(handles.X2,'Enable','on');
set(handles.X3,'Enable','on');
set(handles.X4,'Enable','on');
set(handles.Y1,'Enable','on');
set(handles.Y2,'Enable','on');
set(handles.Y3,'Enable','on');
set(handles.Y4,'Enable','on');
set(handles.TStart,'Enable','on');
set(handles.TEnd,'Enable','on');
set(handles.CurrentTime,'Enable','on');
set(handles.TimeSlider,'Enable','on');
set(handles.SetRegion,'Enable','on');
set(handles.ResetTime,'Enable','on');
set(handles.SaveData,'Enable','on');
set(handles.RegionDraw,'Enable','on');
catch err %Display error if above did not work
errordlg('File not found','File Error');
end

 try%Try to see if the following code works, otherwise catch and display
error
g1 = load('Group1.txt');
g2 = load('Group2.txt');
g3 = load('Group3.txt');
g4 = load('Group4.txt');
g5 = load('Group5.txt');
%Make Everything Matrix
CamEver = CameraMatrixNew;
%Make ExEverthing Matrix
CamEx=CameraMatrixNew;
for i = 1:length(g1)
index = CamEx(:,5)==g1(i);CamEx(index,:)=[];
end
for i = 1:length(g2)
index = CamEx(:,5)==g2(i);CamEx(index,:)=[];
end
for i = 1:length(g3)
index = CamEx(:,5)==g3(i);CamEx(index,:)=[];
```

```matlab
end
for i = 1:length(g4)
index = CamEx(:,5)==g4(i);CamEx(index,:)=[];
end
for i = 1:length(g5)
index = CamEx(:,5)==g5(i);CamEx(index,:)=[];
end

%make G1 Matrix
CamG1 = [];
for i = 1:length(g1)
index = CameraMatrixNew(:,5)==g1(i); CamG1=[CamG1; CameraMatrixNew(index,:)];
end
CamG1
%make G2 Matrix
CamG2 = [];
for i = 1:length(g2)
index = CameraMatrixNew(:,5)==g2(i); CamG2=[CamG2; CameraMatrixNew(index,:)];
end

%make G3 Matrix
CamG3 = [];
for i = 1:length(g3)
index = CameraMatrixNew(:,5)==g3(i); CamG3=[CamG3; CameraMatrixNew(index,:)];
end

%make G4 Matrix
CamG4 = [];
for i = 1:length(g4)
index = CameraMatrixNew(:,5)==g4(i); CamG4=[CamG4; CameraMatrixNew(index,:)];
end

%make G5 Matrix
CamG5 = [];
for i = 1:length(g5)
index = CameraMatrixNew(:,5)==g5(i); CamG5=[CamG5; CameraMatrixNew(index,:)];
end

%Same plotted data as above.
 if (EveryoneFlag)
    index =
(CamEver(:,3)>round(ExpTime(1)*100)/100)&(CamEver(:,3)<round(ExpTime(1)*100)/
100+0.02);
    Sugar = CamEver(index,:);
   plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
   axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 elseif(ExEveryoneFlag)
```

```matlab
    index =
(CamEx(:,3)>round(ExpTime(1)*100)/100)&(CamEx(:,3)<round(ExpTime(1)*100)/100+
0.02);
    Sugar = CamEx(index,:);
    plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
    axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 elseif(Group1Flag)
     index =
(CamG1(:,3)>round(ExpTime(1)*100)/100)&(CamG1(:,3)<round(ExpTime(1)*100)/100+
0.02);
    Sugar = CamG1(index,:);
    plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
    axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 elseif(Group2Flag)
     index =
(CamG2(:,3)>round(ExpTime(1)*100)/100)&(CamG2(:,3)<round(ExpTime(1)*100)/100+
0.02);
    Sugar = CamG2(index,:);
   plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
   axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 elseif(Group3Flag)
     index =
(CamG3(:,3)>round(ExpTime(1)*100)/100)&(CamG3(:,3)<round(ExpTime(1)*100)/100+
0.02);
    Sugar = CamG3(index,:);
    plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
    axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;
```

```matlab
text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 elseif(Group4Flag)
     index =
(CamG4(:,3)>round(ExpTime(1)*100)/100)&(CamG4(:,3)<round(ExpTime(1)*100)/100+
0.02);
    Sugar = CamG4(index,:);
    plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
    axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 elseif(Group5Flag)
     index =
(CamG5(:,3)>round(ExpTime(1)*100)/100)&(CamG5(:,3)<round(ExpTime(1)*100)/100+
0.02);
    Sugar = CamG5(index,:);
    plot(insidex,insidey,'b',outsidex,
outsidey,'b',Doorlx,Doorly,'b',Doorrx,Doorry,'b','parent',handles.axes1);
    axis([xmin xmax ymin ymax]);
    hold on;
    plot(RegionBoxX,RegionBoxY,'r','parent',handles.axes1);
    hold on;

text(Sugar(:,1),Sugar(:,2),num2str(Sugar(:,5),3),'HorizontalAlignment','cente
r','EdgeColor','black');
    hold off;
 end
catch erro%Display error if above did not work
    errordlg('Group Files not Found or Error in Files. Make sure they take
format #,#,#,#, etc ','File Error');
end


% --- Executes on button press in ResetTime.
function ResetTime_Callback(hObject, eventdata, handles)
global TStart; global TEnd; global ExpTime;
TStart = min(ExpTime); set(handles.TStart,'String',num2str(TStart))
TEnd = max(ExpTime);set(handles.TEnd,'String',num2str(TEnd))


% --- Executes on button press in RegionDraw.
function RegionDraw_Callback(hObject, eventdata, handles)
global x1; global x2; global x3; global x4; global y1; global y2; global y3;
global y4;
global RegionBoxX; global RegionBoxY;
[x,y] = ginput(4);
```

```matlab
set(handles.X1,'String',num2str(x(1)))
set(handles.X2,'String',num2str(x(2)))
set(handles.X3,'String',num2str(x(3)))
set(handles.X4,'String',num2str(x(4)))
set(handles.Y1,'String',num2str(y(1)))
set(handles.Y2,'String',num2str(y(2)))
set(handles.Y3,'String',num2str(y(3)))
set(handles.Y4,'String',num2str(y(4)))
set(handles.WholeBox,'Value',0);
set(handles.RegionBox,'Value',1);


x1 =get(handles.X1,'String');
x2 =get(handles.X2,'String');
x3 =get(handles.X3,'String');
x4 =get(handles.X4,'String');
y1 =get(handles.Y1,'String');
y2 =get(handles.Y2,'String');
y3 =get(handles.Y3,'String');
y4 =get(handles.Y4,'String');


 function Out = DelimAverage(Input,Delim)


A = Input;
delim = Delim;
Ids = unique(A(:,5));
Final = [];
for b = 1:length(Ids)


    index = A(:,5)== Ids(b);
    %index = A(:,5)== 55;
    Salt = A(index,:);
    time = unique(floor(Salt(:,3)));
    Anew = [length(time*delim) length(time*delim) length(time*delim)
length(time*delim) length(time*delim)];
    sumx = 0;
    sumy = 0;
    count = 0;
    insidecount = 0;
    Cam= 0;
    Id = 0;
    shiftindex = 1;
    for a = 1:length(time);
        for i = 1:length(Salt(:,3))
            if floor(Salt(i,3)) == time(a);
            count = count +1;
            end
        end
        for i = 1:length(Salt(:,3))
            if floor(Salt(i,3)) == time(a);
                sumx = sumx + Salt(i,1);
                sumy = sumy + Salt(i,2);
                insidecount = insidecount + 1;
                Cam = Salt(i,4);
                Id = Salt(i,5);
                if insidecount == floor(count/delim);
```

```
                        Anew(shiftindex,1) = sumx/(insidecount)  ;
                        Anew(shiftindex,2) = sumy/(insidecount );
                        Anew(shiftindex,3) = Salt(i,3);
                        Anew(shiftindex,4) = Cam;
                        Anew(shiftindex,5) = Id;
                        sumx = 0;
                        sumy = 0;
                        insidecount = 0;
                        Cam = 0;
                        Id  = 0;
                        shiftindex = shiftindex+1;
                    end

            end
        end
       % pause;
         sumx = 0;
        sumy = 0;
        count = 0;
        insidecount = 0;
        Cam= 0;
        Id = 0;
    end
    Final = [Final;Anew];
end


Out = Final;



function Out = Delim(Input,Delim)

A = Input;
delim = Delim;
Ids = unique(A(:,5));
Final = [];
for b = 1:length(Ids)

    index = A(:,5)== Ids(b);
    %index = A(:,5)== 55;
    Salt = A(index,:);
    time = unique(floor(Salt(:,3)));
    Anew = [length(time*delim) length(time*delim) length(time*delim)
length(time*delim) length(time*delim)];
    sumx = 0;
    sumy = 0;
    count = 0;
    insidecount = 0;
    Cam= 0;
    Id = 0;
    shiftindex = 1;
    for a = 1:length(time);
        for i = 1:length(Salt(:,3))
            if floor(Salt(i,3)) == time(a);
```

```matlab
                count = count +1;
            end
        end
        for i = 1:length(Salt(:,3))
            if floor(Salt(i,3)) == time(a);
                insidecount = insidecount + 1;
                Cam = Salt(i,4);
                Id = Salt(i,5);
                if insidecount == floor(count/delim);
                    Anew(shiftindex,1) = Salt(i,1) ;
                    Anew(shiftindex,2) = Salt(i,2);
                    Anew(shiftindex,3) = Salt(i,3);
                    Anew(shiftindex,4) = Cam;
                    Anew(shiftindex,5) = Id;
                    insidecount = 0;
                    Cam = 0;
                    Id  = 0;
                    shiftindex = shiftindex+1;
                end

            end
        end
        % pause;
        count = 0;
        insidecount = 0;
        Cam= 0;
        Id = 0;
    end
    Final = [Final;Anew];
end

Out = Final;


function Out = SmoothSec(Input)

A = Input;
counsol=A
Ids = unique(A(:,5));
Final = [];
for b = 1:length(Ids)

    index = A(:,5)== Ids(b);
    Salt = A(index,:);
    Anew = [smooth(Salt(:,1),'sgolay') smooth(Salt(:,2),'sgolay') Salt(:,3)
Salt(:,4) Salt(:,5)];
    Final = [Final;Anew];
end

Out = Final;

% --- Executes on button press in SaveData.
function SaveData_Callback(hObject, eventdata, handles)
global CameraMatrixNew; global SessionTime; global ExpTime;
global TStart; global TEnd; global CurTime; global CamNum;
```

```matlab
global RegionBoxX;
global RegionBoxY;
global FinalCameraMatrix;
global Doorlx;
global Doorly;
global Doorrx;
global Doorry;
global outsidex;
global outsidey;
global insidex;
global insidey;
global IDTFin;
global IDTInit;
global SpacingWall;
global ID;
global FinalCameraMatrix; global DirectionValues; global V; global Ac; global
Orient;
global AnalID; global pdeltax; global pdeltay; global Reldeltax; global
Reldeltay;
global Average; global Inter; global SmthFilt; global raw; global delim;
global g1; global g2; global g3; global g4; global g5;
global CamEver; global CamEx; global CamG1; global CamG2; global CamG3;
global CamG4; global CamG5;
global EveryoneFlag; global ExEveryoneFlag; global Group1Flag; global
Group2Flag; global Group3Flag; global Group4Flag; global Group5Flag;
%All of this code is exactly like overall gui, but has changed only in that
%the output file also now has camera number in filename.
index = (ExpTime>TStart)&(ExpTime<TEnd);
DataTime = ExpTime(index);
 tic
h = waitbar(0,'Please wait...Processing Trajectories');
steps = length(DataTime);
%steps
A = [];

for i=1:length(DataTime)
    waitbar(i / steps);
     if (EveryoneFlag)
        %disp('in everyoneflag');
        index =
(CamEver(:,3)>round(DataTime(i)*100)/100)&(CamEver(:,3)<round(DataTime(i)*100
)/100+0.1);
        Sugar = CamEver(index,:);
       in = inpolygon(Sugar(:,1),Sugar(:,2),RegionBoxX,RegionBoxY);
       Temp = Sugar(in,:);

        A = [A;Temp];
     elseif(ExEveryoneFlag)
        %disp('in exeveryoneflag');
        index =
(CamEx(:,3)>round(DataTime(i)*100)/100)&(CamEx(:,3)<round(DataTime(i)*100)/10
0+0.1);
        Sugar = CamEx(index,:);
      in = inpolygon(Sugar(:,1),Sugar(:,2),RegionBoxX,RegionBoxY);
       Temp = Sugar(in,:);
```

```matlab
        A = [A;Temp];
    elseif(Group1Flag)
        %disp('in gp1');
        %DataTime(i)
        index =
(CamG1(:,3)>round(DataTime(i)*100)/100)&(CamG1(:,3)<round(DataTime(i)*100)/10
0+0.1);
        Sugar = CamG1(index,:);
     in = inpolygon(Sugar(:,1),Sugar(:,2),RegionBoxX,RegionBoxY);
      Temp = Sugar(in,:);
      A = [A;Temp];
    elseif(Group2Flag)
        %disp('in gp2');
        index =
(CamG2(:,3)>round(DataTime(i)*100)/100)&(CamG2(:,3)<round(DataTime(i)*100)/10
0+0.1);
        Sugar = CamG2(index,:);
     in = inpolygon(Sugar(:,1),Sugar(:,2),RegionBoxX,RegionBoxY);
      Temp = Sugar(in,:);
        A = [A;Temp];
    elseif(Group3Flag)
        %disp('in gp3');
        index =
(CamG3(:,3)>round(DataTime(i)*100)/100)&(CamG3(:,3)<round(DataTime(i)*100)/10
0+0.1);
        Sugar = CamG3(index,:);
     in = inpolygon(Sugar(:,1),Sugar(:,2),RegionBoxX,RegionBoxY);
      Temp = Sugar(in,:);
        A = [A;Temp];
    elseif(Group4Flag)
        %disp('in gp4');
        index =
(CamG4(:,3)>round(DataTime(i)*100)/100)&(CamG4(:,3)<round(DataTime(i)*100)/10
0+0.1);
        Sugar = CamG4(index,:);
     in = inpolygon(Sugar(:,1),Sugar(:,2),RegionBoxX,RegionBoxY);
      Temp = Sugar(in,:);
        A = [A;Temp];
    elseif(Group5Flag)
       % disp('in gp5');
        index =
(CamG5(:,3)>round(DataTime(i)*100)/100)&(CamG5(:,3)<round(DataTime(i)*100)/10
0+0.1);
        Sugar = CamG5(index,:);
     in = inpolygon(Sugar(:,1),Sugar(:,2),RegionBoxX,RegionBoxY);
      Temp = Sugar(in,:);
        A = [A;Temp];
    end
end
%assignin('base','BeforeA',A);
A = UNIQUE(A,'rows');
A = sortrows(A,3);
if(raw)
    dlmwrite(strcat(SessionTime,'_',CamNum,'_NewOutput.txt'),
[A(:,1)';A(:,2)';A(:,3)';A(:,4)';A(:,5)']','-append')
 elseif(Average)
```

```matlab
    A = DelimAverage(A,delim);
    dlmwrite(strcat(SessionTime,'_',CamNum,'_NewOutput.txt'),
[A(:,1)';A(:,2)';A(:,3)';A(:,4)';A(:,5)']','-append')
 elseif(SmthFilt)
     A = SmoothSec(A);
    dlmwrite(strcat(SessionTime,'_',CamNum,'_NewOutput.txt'),
[A(:,1)';A(:,2)';A(:,3)';A(:,4)';A(:,5)']','-append')
 elseif(Inter)
    A = Delim(A,delim);
    dlmwrite(strcat(SessionTime,'_',CamNum,'_NewOutput.txt'),
[A(:,1)';A(:,2)';A(:,3)';A(:,4)';A(:,5)']','-append')
 end
  close(h);


%Matrix to be written to a file for trajectories
FinalCameraMatrix = [A(:,1)';A(:,2)';A(:,3)';A(:,4)';A(:,5)']';
global AllocSize;
AllocSize= length(A(:,1));
ID = unique(A(:,5));
p = waitbar(0,'Please wait...Processing Velocities');
% velocity=11111
% tic
V = zeros(AllocSize*3,5);
%V = [];
ind = 1;
for i=1:length(ID)
     waitbar(i / length(ID));
      index = (FinalCameraMatrix(:,5) == ID(i));
        Sugar = FinalCameraMatrix(index,:); %separates out to individual ID
        for j=1:length(Sugar(:,1))-1
            DistanceX=Sugar(j,1)-Sugar(j+1,1);
            DistanceY=Sugar(j,2)-Sugar(j+1,2);
            TimeDiff=Sugar(j,3)-Sugar(j+1,3);
            VX = DistanceX/TimeDiff;
            VY = DistanceY/TimeDiff;
            Temp = [VX VY Sugar(j+1,3) Sugar(j,4) Sugar(j,5)];
             V(ind,:)=Temp;
%            V=[V;Temp];
            ind=ind+1;
        end
end

   V(all(V==0,2),:)=[];

VXmean = sum(V(:,1))/length(V(:,1));
VYmean = sum(V(:,2))/length(V(:,2));
VMean = sqrt(VXmean^2+VYmean^2)*0.3048;
dlmwrite(strcat(SessionTime,'_',CamNum,'_NewInstVelOutput.txt'),
[V(:,1)';V(:,2)';V(:,3)';V(:,4)';V(:,5)']','-append')
close(p);
% toc
```

```matlab
q = waitbar(0,'Please wait...Processing Accelerations');
% acc=11111
% tic
Ac = zeros(AllocSize*3,5);
%Ac = [];
ind=1;
for i=1:length(ID)
    waitbar(i / length(ID));
      index = (V(:,5) == ID(i));
        Sugar = V(index,:); %separates out to individual ID
        for j=1:length(Sugar(:,1))-1;
            delVX=Sugar(j,1)-Sugar(j+1,1);
            delVY=Sugar(j,2)-Sugar(j+1,2);
            TimeDiff=Sugar(j,3)-Sugar(j+1,3);
            AX = delVX/TimeDiff;
            AY = delVY/TimeDiff;
            Temp = [AX AY Sugar(j+1,3) Sugar(j,4) Sugar(j,5)];
             Ac(ind,:)=Temp;
%              Ac=[Ac;Temp];
            ind=ind+1;
        end
end

 Ac(all(Ac==0,2),:)=[];

AXmean = sum(Ac(:,1))/length(Ac(:,1));
AYmean = sum(Ac(:,2))/length(Ac(:,2));
AMean = sqrt(AXmean^2+AYmean^2)*0.3048;
dlmwrite(strcat(SessionTime,'_',CamNum,'_NewInstAccOutput.txt'),
[Ac(:,1)';Ac(:,2)';Ac(:,3)';Ac(:,4)';Ac(:,5)']','-append')
close(q);
y = waitbar(0,'Please wait...Processing Headings');
% toc


 Orient = zeros(AllocSize*3,4);
%Orient = [];
ind=1;
% Orient6=1111
% tic

for i=1:length(ID)
    waitbar(i / length(ID));
      index = (FinalCameraMatrix(:,5) == ID(i));
        Sugar = FinalCameraMatrix(index,:); %separates out to individual ID
        for j=1:length(Sugar(:,1))-1;
            x1 = Sugar(j,1);
            x2 = Sugar(j+1,1);
            y1 = Sugar(j,2);
            y2 = Sugar(j+1,2);
            [Din1, Dout1,long11, long12] = WallDistance(x1, y1);
            [Din2, Dout2, long21, long22] = WallDistance(x2, y2);
            D = sqrt((x2-x1)^2+(y2-y1)^2);
            opp = sqrt(D^2-(Dout1-Dout2)^2);
            Theta = atan2(opp,Dout1-Dout2)*(180/pi);
```

```matlab
            Temp = [Theta Sugar(j,3) Sugar(j,4) Sugar(j,5)];
        Orient(ind,:) = Temp;
%           Orient=[Orient;Temp];
         ind = ind+1;
          end
end


 Orient(all(Orient==0,2),:)=[];

dlmwrite(strcat(SessionTime,'_',CamNum,'_NewOrientOutput.txt'),
[Orient(:,1)';Orient(:,2)';Orient(:,3)';Orient(:,4)']','-append')
close(y)
% toc




p = waitbar(0,'Please wait...Processing Direction');
 DirectionValues = zeros(AllocSize*3,5);
 ind = 1;
%  direction=1111
%  tic

%DirectionValues = [];
Directionxtot=0;
Directionytot=0;
count =0;
for i=1:length(ID)
    waitbar(i / length(ID));
      index = (FinalCameraMatrix(:,5) == ID(i));
        Sugar = FinalCameraMatrix(index,:); %separates out to individual ID
        for j=1:length(Sugar(:,1))-1;
            count =1;
            while((abs(Sugar(j,3)-Sugar(j+count,3))< 1))
                 count = count +1;
                 if j+count > length(Sugar(:,3))
                     count = count-1;
                     break;
                 end
            end
             [Directionx,Directiony] = DirectionTest(Sugar(j,1), Sugar(j,2),
Sugar(j+count-1,1), Sugar(j+count-1,2));
             Temp = [Directionx Directiony Sugar(j+1,3) Sugar(j,4)
Sugar(j,5)];
              DirectionValues(ind,:) = Temp;
            %DirectionValues=[DirectionValues;Temp];
             ind = ind+1;
        end
end

% DirectionValues(all(DirectionValues==0,2),:)=[];
```

```matlab
dlmwrite(strcat(SessionTime,'_',CamNum,'_NewDirectionOutput.txt'),
[DirectionValues(:,1)';DirectionValues(:,2)';DirectionValues(:,3)';DirectionV
alues(:,4)';DirectionValues(:,5)']','-append')
close(p);
% toc
toc


 function CamNum_Callback(hObject, eventdata, handles)
global CamNum;
CamNum = get(hObject,'String');




% --- Executes during object creation, after setting all properties.
function CamNum_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

  % --- Executes when selected object is changed in Filtering.
function Filtering_SelectionChangeFcn(hObject, eventdata, handles)
global Average; global Inter; global SmthFilt; global raw;
switch get(eventdata.NewValue,'Tag') % Get Tag of selected object.
    case 'DelimAverage'
        % Code for when radiobutton1 is selected.
        Average = 1;
        Inter = 0;
        SmthFilt = 0;
        raw = 0;
    case 'Delim'
        % Code for when radiobutton2 is selected.
        Average = 0;
        Inter = 1;
        SmthFilt = 0;
        raw = 0;
    case 'smooth'
        % Code for when togglebutton1 is selected.
        Average = 0;
        Inter = 0;
        SmthFilt = 1;
        raw = 0;
    case 'Raw'
        % Code for when togglebutton2 is selected.
        Average = 0;
        Inter = 0;
        SmthFilt = 0;
        raw = 1;
    % Continue with more cases as necessary.
    otherwise
        a=0
        % Code for when there is no match.
end
```

```matlab
function delimbox_Callback(hObject, eventdata, handles)
global delim;
delim = str2double(get(hObject,'String')); %Grab entered string for Tstart
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function delimbox_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function AnalID_Callback(hObject, eventdata, handles)
global AnalID;
AnalID = str2double(get(hObject,'String')); %Analysis ID
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function AnalID_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function pdeltax_Callback(hObject, eventdata, handles)
global pdeltax;
pdeltax = str2double(get(hObject,'String')); %Grab entered string for Tstart
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function pdeltax_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function pdeltay_Callback(hObject, eventdata, handles)

global pdeltay;
pdeltay = str2double(get(hObject,'String')); %Grab entered string for Tstart
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function pdeltay_CreateFcn(hObject, eventdata, handles)
```

```matlab
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


 function Reldeltax_Callback(hObject, eventdata, handles)
global Reldeltax;
Reldeltax = str2double(get(hObject,'String')); %Grab entered string for
Tstart
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function Reldeltax_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


 function Reldeltay_Callback(hObject, eventdata, handles)
global Reldeltay;
Reldeltay = str2double(get(hObject,'String')); %Grab entered string for
Tstart
guidata(hObject, handles);

 % --- Executes during object creation, after setting all properties.
function Reldeltay_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

  function IDTInit_Callback(hObject, eventdata, handles)
global IDTInit;
IDTInit = str2double(get(hObject,'String')); %Grab entered string for Tstart
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function IDTInit_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function IDTFin_Callback(hObject, eventdata, handles)
global IDTFin;
IDTFin = str2double(get(hObject,'String')); %Grab entered string for Tstart
guidata(hObject, handles);
```

```matlab
% --- Executes during object creation, after setting all properties.
function IDTFin_CreateFcn(hObject, eventdata, handles)
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in IntervalButton.
function IntervalButton_Callback(hObject, eventdata, handles)
global CameraMatrixNew; global SessionTime; global ExpTime;
global TStart; global TEnd; global CurTime; global CamNum;
global RegionBoxX;
global RegionBoxY;
global FinalCameraMatrix;
global Doorlx;
global Doorly;
global Doorrx;
global Doorry;
global outsidex;
global outsidey;
global insidex;
global insidey;
global IDTFin;
global IDTInit;

global FinalCameraMatrix; global DirectionValues; global V; global Ac; global
Orient;
global AnalID; global pdeltax; global pdeltay; global Reldeltax; global
Reldeltay;
global Average; global Inter; global SmthFilt; global raw; global delim;
global g1; global g2; global g3; global g4; global g5;
global CamEver; global CamEx; global CamG1; global CamG2; global CamG3;
global CamG4; global CamG5;
global EveryoneFlag; global ExEveryoneFlag; global Group1Flag; global
Group2Flag; global Group3Flag; global Group4Flag; global Group5Flag;
global AllocSize; global SpacingWall;


q = waitbar(0,'Please wait...Processing Leader/Collider');
% figure;
 LC=zeros(AllocSize*4,8);
%LC=[];
ind = 1;
% LC1=1111111111
% tic

PSpacex = 0;
        PSpacey = 0;
```

```
index = (FinalCameraMatrix(:,5) == AnalID);
Sugar = FinalCameraMatrix(index,:); %separates out to individual ID



for j=1:length(Sugar(:,1))-1;
    index = ((DirectionValues(:,3)> Sugar(j,3)-
1)&(DirectionValues(:,3)<Sugar(j,3)+1));
    Bane = DirectionValues(index,:);
    index = (Bane(:,5)==AnalID);
    Abel = Bane(index,:);
    DirectionXtarget = sum(Abel(:,1))/length(Abel(:,1));
    DirectionYtarget = sum(Abel(:,2))/length(Abel(:,1));
    index = (DirectionValues(:,3)==Sugar(j,3));
    Bane = DirectionValues(index,:);
    index = (Bane(:,5)==AnalID);
    Abel = Bane(index,:);

    if (Abel)
        Sugar(j,1)
        Sugar(j,2)
        DirectionXtarget
        DirectionYtarget
        pdeltax
        pdeltay

        [PSpacex,PSpacey] = PersonalSpace(Sugar(j,1), Sugar(j,2),
DirectionXtarget, DirectionYtarget, pdeltax, pdeltay);
       % DirectionYtarget

%          Sugar(j,1)
%          Sugar(j,2)
%          Sugar(j,3)
        index = (FinalCameraMatrix(:,3)> Sugar(j,3)-
0.01)&(FinalCameraMatrix(:,3)<Sugar(j,3)+0.01)&(FinalCameraMatrix(:,5)~=
AnalID);
        temp1 = FinalCameraMatrix(index,:);
        in = inpolygon(temp1(:,1),temp1(:,2),PSpacex,PSpacey); %This is
        temp2 = temp1(in,:);

         PSpacex = 0;
         PSpacey = 0;

    if ~isempty(temp2)
     dlmwrite(strcat(SessionTime,'_',CamNum,'_temp2.txt'),
[temp2(:,1)';temp2(:,2)';temp2(:,3)';temp2(:,4)';temp2(:,5)']','-append')
    end
        if (temp2)

            for i = 1:length(temp2(:,1));
                %temp2
                index = ((DirectionValues(:,3)> temp2(i,3)-
3)&(DirectionValues(:,3)<temp2(i,3)+3)); %Find time of ID for each ID loop
through
                apple = DirectionValues(index,:);
```

```matlab
                index = (apple(:,5)==(temp2(i,5)));
                orange = apple(index,:);
                apple = orange;

                DirectionXtest = sum(apple(:,1))/length(apple(:,1));
                DirectionYtest = sum(apple(:,2))/length(apple(:,2));
%
                DirectionXtarget;
                DirectionYtarget;
                %Determine Leader/Collider STatus
%                 apple
                [Leader,Collider] = LeaderCollider(DirectionXtarget,
DirectionYtarget,DirectionXtest, DirectionYtest, temp2(i,1), temp2(i,2));
                Temp = [Leader Collider temp2(i,1) temp2(i,2) temp2(i,3)
temp2(i,5) temp2(i,4) AnalID];
                 LC(ind,:) = Temp;
%                 LC=[LC;Temp];
                ind = ind+1;
                %Write to matrix [leader collider ID timeAnalID AnalID
Camera]
            end
        end
    end
end
% movie2avi(mov, 'movie.avi',  'Compression','none','FPS',6);
 LC(all(LC==0,2),:)=[];

if ~isempty(LC)
    dlmwrite(strcat(SessionTime,'_',CamNum,'_NewLCOutput.txt'),
[LC(:,1)';LC(:,2)';LC(:,3)';LC(:,4)';LC(:,5)';LC(:,6)';LC(:,7)';LC(:,8)']','-
append')
end
close(q);
% toc


p = waitbar(0,'Please wait...Processing relative Vel,acc, and spacing');
RelVel=zeros(AllocSize*10,6);
RelAcc=zeros(AllocSize*10,6);
RelSpace=zeros(AllocSize*10,7);
ind=1;
ind2=1;
ind3=1;

Vtest = [];
Atest = [];
index = (FinalCameraMatrix(:,5) == AnalID);
Sugar = FinalCameraMatrix(index,:); %separates out to individual ID
bottomrangeV=0;
bottomrangeA=0;


for j=1:length(Sugar(:,1))-1;
    waitbar(j / (length(Sugar(:,1))-1));
```

```matlab
    [RSpacex,RSpacey] = RelativeSpace(Sugar(j,1), Sugar(j,2),Reldeltax,
Reldeltay);
    index = (FinalCameraMatrix(:,3)> Sugar(j,3)-
0.02)&(FinalCameraMatrix(:,3)<Sugar(j,3)+0.02)&(FinalCameraMatrix(:,5)~=
AnalID);
    temp1 = FinalCameraMatrix(index,:);
    in = inpolygon(temp1(:,1),temp1(:,2),RSpacex,RSpacey); %Test to see if
    temp2 = temp1(in,:);
    time = Sugar(j,3);
    timeint = 0.018;
    index = (V(:,5) == AnalID)& (V(:,3) < Sugar(j,3)+timeint)& (V(:,3) >
bottomrangeV);
    tempV = V(index,:);
    while (isempty(tempV))
        timeint = timeint+0.001;
        index = (V(:,5) == AnalID)& (V(:,3) < Sugar(j,3)+timeint)& (V(:,3) >
bottomrangeV);
        tempV = V(index,:);
        if((Sugar(j,3)+timeint)>max(Sugar(:,3)))
            break;
        end
    end
    if((Sugar(j,3)+timeint)>max(Sugar(:,3)))
        VX = 0;
        VY = 0;
        bottomrangeV =max(Sugar(:,3));
    else
%      tempV;
        VX = tempV(:,1);
        VY = tempV(:,2);
        bottomrangeV =tempV(:,3);
    end
%      distV = time-bottomrangeV;
    if (length(VX)>1)
        greatervx =1;
    pause;
    end
    if (length(VY)>1)
        greatervy =1;
    pause;
    end
    Vtest = [Vtest;VX VY];
    %***********Get ACCvx and ACCy for ID at current time\
    time = Sugar(j,3);
    timeint = 0.018;
    index = (Ac(:,5) == AnalID)& (Ac(:,3) < Sugar(j,3)+timeint)& (Ac(:,3) >
bottomrangeA);
    tempA = Ac(index,:);
    while (isempty(tempA))
        timeint = timeint+0.001;
        index = (Ac(:,5) == AnalID)& (Ac(:,3) < Sugar(j,3)+timeint)& (Ac(:,3)
> bottomrangeA);
        tempA = Ac(index,:);
%        Sugar(j,3)+timeint;
%        bottomrangeA;
        if((Sugar(j,3)+timeint)>max(Sugar(:,3)))
```

```matlab
            break;
        end
    end
    if((Sugar(j,3)+timeint)>max(Sugar(:,3)))
            AX = 0;
            AY = 0;
            bottomrangeA =max(Sugar(:,3));
    else
%         tempA;
        AX = tempA(:,1);
        AY = tempA(:,2);
        bottomrangeA =tempA(:,3);
    end
%     distA = time-bottomrangeA;
    if (length(AX)>1)
    pause;
    greaterax =1;
    end
    if (length(AY)>1)
        greateray =1;
    pause;
    end
    Atest = [Atest;AX AY];

    if (temp2)
    %If there are IDs within my relative space
        for i = 1:length(temp2(:,1));

                %***********Get relvx and Relvy for thisID at currentTime
                %do the same thing as above and see if it works
                index = (V(:,5) == temp2(i,5))& (V(:,3) < bottomrangeV+0.01)&
(V(:,3) > bottomrangeV-0.01);
                tempV = V(index,:);
                if isempty(tempV)
                    VXtarget = 0;
                    VYtarget = 0;
                else
                    VXtarget = tempV(1,1);
                    VYtarget = tempV(1,2);
                end


                TempVel = [(VX-VXtarget) (VY-VYtarget) bottomrangeV
temp2(i,5) temp2(i,4) AnalID];
                 RelVel(ind,:)=TempVel;
                %RelVel=[RelVel;TempVel];
                ind = ind+1;


                index = (Ac(:,5) == temp2(i,5))& (Ac(:,3) <
bottomrangeA+0.01)& (Ac(:,3) > bottomrangeA-0.01);
                tempA = Ac(index,:);
                if isempty(tempA)
```

```matlab
                AXtarget = 0;
                AYtarget = 0;
            else
                AXtarget = tempA(1,1);
                AYtarget = tempA(1,2);
            end

            TempAcc = [(AX-AXtarget) (AY-AYtarget) bottomrangeA
temp2(i,5) temp2(i,4) AnalID];
             RelAcc(ind2,:) = TempAcc;
%             RelAcc=[RelAcc;TempAcc];
            ind2=ind2+1;


            index = (temp2(:,3) < Sugar(j,3)+0.01)& (temp2(:,3) >
Sugar(j,3)-0.01);
            tempT = temp2(index,:);
            Sugar(j,3)
            if (tempT)
                [long, lat] =
RelDistance(Sugar(j,1),Sugar(j,2),tempT(1,1),tempT(1,2));
                TempSpace = [sqrt(long^2+lat^2) long lat Sugar(j,3)
temp2(i,5) temp2(i,4) AnalID];
                 RelSpace(ind3,:) = TempSpace;
%                 RelSpace=[RelSpace;TempSpace];
                ind3=ind3+1;
            end
        end
    end
end

RelVel(all(RelVel==0,2),:)=[];
  RelAcc(all(RelAcc==0,2),:)=[];
RelSpace(all(RelSpace==0,2),:)=[];

if ~isempty(RelVel)
%     RelVel
dlmwrite(strcat(SessionTime,'_',CamNum,'_NewRelVelOutput.txt'),
[RelVel(:,1)';RelVel(:,2)';RelVel(:,3)';RelVel(:,4)';RelVel(:,5)';RelVel(:,6)
']','-append')
end
if ~isempty(RelAcc)
%     RelAcc
dlmwrite(strcat(SessionTime,'_',CamNum,'_NewRelAccOutput.txt'),
[RelAcc(:,1)';RelAcc(:,2)';RelAcc(:,3)';RelAcc(:,4)';RelAcc(:,5)';RelAcc(:,6)
']','-append')
end
if ~isempty(RelSpace)
%     RelSpace
dlmwrite(strcat(SessionTime,'_',CamNum,'_NewRelSpaceOutput.txt'),
[RelSpace(:,1)';RelSpace(:,2)';RelSpace(:,3)';RelSpace(:,4)';RelSpace(:,5)';R
elSpace(:,6)';RelSpace(:,7)']','-append')
end
close(p)
```