METHODS FOR HUMAN MOTION ANALYSIS FOR AMERICAN SIGN LANGUAGE RECOGNITION AND ASSISTIVE ENVIRONMENTS

by

ZHONG ZHANG

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2015

To my mother Ping Lu and my father Shuoping Zhang

who set the example and who made me who I am.

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Vassilis Athitsos for constantly motivating and encouraging me, and also for his invaluable advice during the course of my doctoral studies. I wish to thank my committee members Dr. Farhad Kamangar, Dr. David Levine and Dr. Gian Luca Mariottini for their interest in my research and for taking time out of their extremely busy schedules to serve in my dissertation committee.

I would also like to extend my appreciation to all the members of Vision-Learning-Mining research lab I worked with. When I first joined the lab, Haijing provided much help for me both in study and in life. I wish also to thank Sajjad for the helpful discussions on the hand shape registration project and designing the annotation GUI which frees me from onerous and monotonous annotating work. I'm especially grateful to Paul Doliotis and Alexios Kotsifakos for taking time to help me improving my English including correcting my pronunciation and teaching me a lot of native usages. I also owe other members a favor. Without your guys' support, I can not make it. I also want to thank the tens of students, professors, and staff that I have interacted with in the Computer Science Department, for all the times they have helped and supported me.

I am grateful to all my friends in and outside UTA. There is not enough time and space to list everyone here. However, I should especially thank Yan Ma, Shanshan Lv, Yanliang Liu, Miaomiao Zhang ,Shuo li and Linbing Yu. We've spent a great four years together. And you guys help me throughout my PHD.

Finally, I would like to express my deepest gratitude to my father and mother for their unwavering support and encouragement for my PHD study.

April 2, 2015

ABSTRACT

METHODS FOR HUMAN MOTION ANALYSIS FOR AMERICAN SIGN LANGUAGE RECOGNITION AND ASSISTIVE ENVIRONMENTS

ZHONG ZHANG, Ph.D.

The University of Texas at Arlington, 2015

Supervising Professor: Vassilis Athitsos

The broad application domain of the work presented in this thesis is human motion analysis with a focus on hand detection for American Sign Language recognition and fall detection for assistive environments.

One of the motivations of the proposed thesis is a semi-automatic vision based American Sign Language recognition system. This system allows a user to submit as query a video of the sign of interest, or simply perform the sign in front of a camera. The system then asks the user to annotate the hands' locations in the sign. Next, the hand trajectory of the query sign is compared with the models in a large sign database to find the best matches. At last, the user reviews the top results to verify which of them best matches the query sign. Towards making the system more automatic, a novel hand detection method is introduced which is a combination of four representative hand detection methods published in these years.

On the topic of fall detection for assistive environments, the work in this thesis aims at improving the safety of patients and elderly persons living unaccompanied at home. More specifically, this thesis proposes a fully automatic vision based fall detection method which can serve as a component of a home monitoring system for elderly people. The major contributions of the fall detection work can be summarized as: (i) This thesis collects three kinds of fall datasets using Microsoft Kinect depth cameras: non-occlusion dataset, partial occlusion dataset and complete occlusion dataset. The non-occlusion dataset refers to the performer being always visible to the camera when he/she falls down. A partial occlusion fall refers to a fall where part of the body is occluded by a certain object when the person performs the fall action. When the end of a fall is totally occluded by a certain object, like a bed, the fall is called a complete occlusion case. All of these datasets are freely available online, together with annotations marking the beginning and end of each fall event. As far as we know, this is the first public fall datasets captured by depth camera. These datasets will enable researchers to explore their own fall detection methods. (ii) This thesis proposes a statistical fall detection method based on a single Kinect depth camera, that makes a decision based on information about how the human moved during the last few frames. Our method proposes novel features to be used for fall detection, and combines those features using a Bayesian framework. The proposed method is quantitatively compared with three most related publications which also use a single depth camera on the collected datasets. Experimental results demonstrate that the proposed method obtains much better detection accuracy than other competitors on non-occlusion and partial occlusion datasets. As for the complete occlusion dataset, although the proposed method does not get the best detection accuracy, the evaluation between the proposed method and the competitors can be taken as a benchmark for the assessment of more advanced fall detection method.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Researchers have been studying human motion analysis for several years, and it is still one of the most active research topics in the computer vision community. This topic has attracted such great interest from computer vision researchers due to its wide spectrum of applications such as sign language recognition, assistive environments, etc. The tasks involved in human motion analysis can be broadly divided into three categories: human activity recognition, human motion tracking, and analysis of body and body part movement. The focus of this thesis lies in recognizing human activities from image sequences. More specifically, this thesis is motivated by a semi-automatic vision-based American Sign Language recognition system and explores several hand detections to make this system towards more automatic. In addition, a statistical-based method using a Kinect depth camera is proposed for fall activity recognition for assistive environments.

American Sign Language (ASL) is used by millions of people in the U.S. [7, 8, 9]. Unfortunately, many resources taken for granted by users of spoken languages are not available to users of ASL, given its visual nature and its lack of a standard written form. One such resource would be the ability to look up the meaning of a sign. This functionality is especially useful for ASL learners. Imagine this example: A student learns many signs in class and goes back home to do homework. He forgets the meaning of a sign but remembers how to perform it. This situation is similar to knowing how to spell an English word but forgetting what it means, but we can easily look up the meaning of the word in an English dictionary. However, this is not the case for American Sign Language since a typical printed ASL/English dictionary allows users to find out which sign corresponds to an English word but not to discover the meaning of an unknown sign. A vision-based sign recognition system that allows users to look up the meanings of ASL signs would be very useful to these ASL learners.

Figure 1.1 shows the framework of a semi-automatic vision-based American Sign Language recognition system. A user submits a video of a sign of interest to the system. The system then asks the user to annotate where the hands are in each frame, which is the only user interaction step in the entire system. The hand trajectory extracted from the query sign is then compared with models in a large sign dataset to find the best matches. At last, the system presents the user with the top matches so the user can review them one by one to verify which one best matches the query sign.

We first evaluated four representative hand detectors on three sign language datasets to determine whether it was possible to apply an existing hand detection algorithm to the system. Unfortunately, none of these four detectors provided good accuracy. Thus, a novel hand detection algorithm which combines the four representative methods has been proposed. The experimental results demonstrate that the proposed algorithm obtains significantly better accuracy than these four representative methods on three sign language datasets.

Motion analysis can also be useful in detecting falls, which are a major cause of fatal injury for the elderly population, and fear of falling prevents them from living independently. According to the World Health Organization [10], approximately 28-35% of people aged 65 and over fall each year increasing to 32-42% for those over 70 years of age. To improve the living quality of seniors, developing intelligent surveillance systems with the functionality of automatic fall detection is becoming more and more important. Research has already been done to design algorithms to detect

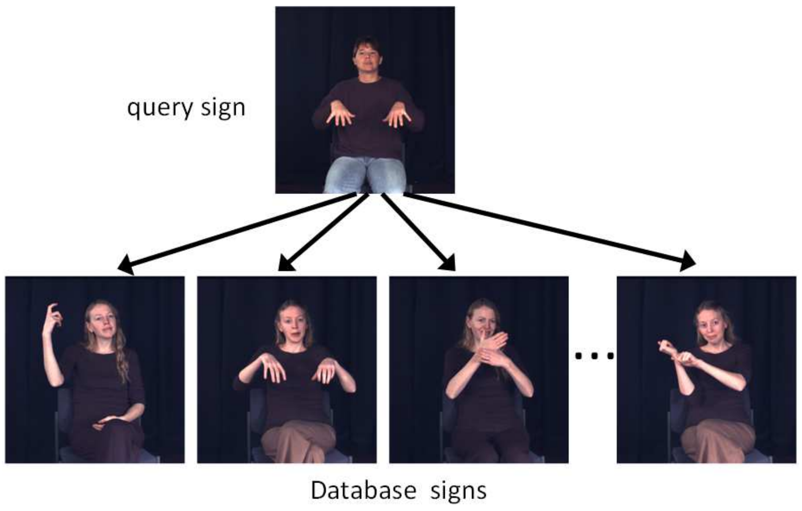


Figure 1.1: The framework of the proposed sign recognition system.

falls. Existing fall detection approaches can be broadly categorized into two groups: using sensors (various accelerometers) and being exclusively vision based. This thesis focuses on vision-based fall detection and proposes a Kinect depth camera-based statistical method, which determines whether a fall has occurred based on information about how the person has moved during the last few frames. Novel features for fall detection are proposed by our method, and these features are fused using a Bayesian framework.

1.1 Main Contributions

In this section we briefly go over the main contributions of this thesis in the aspects of hand detection and fall detection.

1.1.1 Hand Detection for American Sign Language Recognition

This thesis evaluates four hand detection methods [11, 12, 13, 2] on three sign language datasets. Although hundreds of publications regarding hand detection have been released in recent years, we are not aware of any work which evaluates the representative hand detectors on public American Sign Language datasets. This evaluation can serve as a benchmark for the assessment of more advanced hand detection methods on American Sign Language datasets.

A novel hand detection method is proposed by combining these four evaluated hand detection methods. More specifically, the proposed method introduces a two-stage hand detector. The first stage uses a simple skin and motion hand detector to generate hand candidate boxes in each frame. Several features based on these four hand detectors are then computed for each candidate box. Finally, a pre-trained Support Vector Machine (SVM) classifier is applied to these features to obtain a final decision. Experimental results with user-dependent experimental protocol demonstrate the proposed method achieves much better detection accuracy than these four detectors used individually.

1.1.2 Fall Detection Using a Single Depth Camera

The contributions of fall detection work lie in three aspects: (1) creating several publicly available datasets, (2) proposing a novel vision-based fall detection method, and (3) evaluating four fall detection methods on a complete occlusion dataset. We have made one non-occlusion fall dataset, three partial occlusion datasets, and one complete occlusion dataset. All of our datasets and annotations are publicly available online. Researchers can use them as a benchmark to develop their own fall detection methods. As far we know, these are the first publicly available fall datasets captured by Kinect depth cameras.

The proposed statistical method detects falls based on depth images captured using a Kinect camera. Using the depth images, we detect and track the person in both image coordinates and world coordinates. From every frame, features are extracted based on the position and velocity of the person. A Bayesian classifier is built on top of those features. For each frame, the system uses this Bayesian classifier to decide whether a fall event has just occurred. Compared to most existing vision-based fall detection methods, the proposed method in this thesis has the following properties:

- Modeling fall events using a camera-independent world coordinate system allows our method to be viewpoint invariant and to be quite insensitive to the choice of position and viewing direction for the camera.
- A key feature of the experiments is that they include evaluations where all of the training data has been captured from one specific viewpoint, and all of the test data has been captured from a different viewpoint. This experimental setup highlights the fact that camera placement is really not important and that if the camera is moved on purpose or by accident, minimal effort is needed to adjust the system to the new viewpoint.
- The proposed method is also robust to partial occlusion falls, which refers to part of the person being occluded by a certain object in each frame when he/she is falling. As long as the head of the person is visible, the proposed method will work as expected.
- Experimental results show that the proposed method has produced significantly better accuracy than the three most related competitors on both non-occlusion and partial occlusion datasets.

To test the proposed method's capability for handling complete occlusion falls, we evaluate the proposed method and three most related methods on the collected complete occlusion dataset. This evaluation is one of the earliest to discuss handling the complete occlusion falls.

1.2 Overview of This Thesis

In Chapter 2, we first briefly introduce American Sign Language and the overall design of the system. Then the user interaction hand trajectory extraction step is presented in detail. Finally, the Dynamic Time Warping (DTW) based hand trajectory similarity matching is discussed.

Chapter 3 discusses the research work on hand detection. We first evaluate four representative hand detectors published recently on three sign language datasets and then talk about how to combine these four methods to build the proposed hand detector.

In Chapter 4, a novel vision-based fall detection method using a single depth camera is proposed. More specifically, the person detection in both image coordinates and world coordinates is introduced first. After the person detection step, five features extracted from each frame are presented. A Bayesian classifier is then described to combine these five features to decide whether a fall event has just happened. The collected datasets, user-independent and viewpoint-independent experimental protocol, and experimental results are discussed in the experiment section.

Chapter 5 evaluates the fall detection method proposed in chapter 4 and three other related works on a complete occlusion fall dataset.

CHAPTER 2

A SEMI-AUTOMATIC ASL RECOGNITION SYSTEM

This chapter first provides a brief overview of American Sign Language (ASL) and then describes a semi-automatic vision-based ASL recognition system, in which a user is asked to annotate the locations of hands manually. This system motivates the hand detection work described in Chapter 3.

2.1 Introduction to American Sign Language (ASL)

ASL, the predominant sign language of deaf communities in the United States and English-speaking parts of Canada, is a visual-manual language with a structure independent of and very different from spoken English. The sign order of ASL does not always match the word order of spoken English. Signers also can convey grammatical information with their faces, bodies, and the surrounding space.

Manual signs have five parameters or parts: the handshape, which is the specific configuration of the hand; the location of the hand–on the body, on the head, in the space in front of the body; the movement of the hand–up and down, side to side, in an arc; the orientation of the palm–up or down, facing the signer or facing away from the signer; and non-manual signals, which include obligatory facial expressions, eye gaze, specific head positions, or particular body positions [14].

American Sign Language is used by $500,000$ to two million people in the U.S. [7, 8]. Unfortunately, many resources that are taken for granted by users of spoken languages are not available to users of ASL, given its visual nature and its lack of a standard written form. One such resource is the ability to look up the meaning of an unknown sign. When we encounter an English word that we do not understand, we can look it up in a dictionary. Unfortunately, when an ASL user encounters an unknown sign, it is anything but straightforward to find the meaning of that sign. Using a typical printed ASL/English dictionary, one can easily find out which sign corresponds to an English word, but this does not work in the other direction, to discover the meaning of an unknown sign. Some dictionaries do allow look-up based on articulatory properties of the signs. For example, the American Sign Language Handshape Dictionary [15] arranges signs based on the initial handshape, from among 40 basic handshapes. However, even with this dictionary, substantial effort is needed to find a specific sign from among the 1600 included.

A system that lets users search dictionaries of ASL to look up the meaning of unknown signs would definitely be useful to ASL users. In the next section, we will introduce a semi-automatic vision-based sign language recognition system.

2.2 Description of the Sign Recognition System

We describe a semi-automatic vision-based sign language recognition system in this section [16, 17, 18, 19, 20]. In this sign recognition system, the user submits a query of a video of the sign of interest or simply performs the sign in front of a camera. The system then searches a large database of sign videos to find the best matches for the query video and presents the top results to the user. The user can then visually review the top results to verify which (if any) of them best matches the query sign. The database of sign videos used in this recognition system is constituted of a subset of the American Sign Language Lexicon Video Dataset (ASLLVD) [9] (named TL1113) and a subset of the Gallaudet Video Dataset (GVD) [14] (named G1113). ASLLVD is collected and maintained by Boston University and University of Texas, Arlington while GVD is available with the book *The Gallaudet Dictionary* *of American Sign Language* [14]. We will give a brief introduction of the ASLLVD dataset, the GVD dataset and the created subsets in the next section.

2.2.1 Dataset Description

The subset of the dataset ASLLVD, named TL1113, and the subset of the dataset GVD, named as G1113 together constitute the video database used for the sign lookup system. With the help of this video database, the sign lookup system is treated as a video database retrieval problem.

2.2.1.1 Introduction of the Dataset ASLLVD and Its Subset: TL1113

The ASLLVD dataset is introduced by Athitsos et al. [9] for researchers in sign language recognition, gesture recognition, and human activity analysis. An important aspect of this dataset is its comprehensiveness: it includes a set of signs similar in scale and scope to the set of lexical entries in existing English-to-ASL dictionaries [14, 15]. Each sign is performed by at least of one of five native ASL signers: Lana, Dana, Liz, Tyler and Naomi. At this point, the ASLLVD dataset includes at least one video example per sign from a native signer for almost all of the 3,000 signs contained in the *Gallaudet Dictionary of American Sign Language* [14].

The video sequences for the ASLLVD dataset are captured simultaneously from four different cameras, providing a side view, two frontal views, and a view zoomed in on the face of the signer. For the side view, first frontal view, and face view, video is captured at 60 frames per second, non-interlaced, at a resolution of 640x480 pixels per frame. For the second frontal view, video is captured at 30 frames per second, non-interlaced, at a resolution of 1600x1200 pixels per frame. The annotations include, for each sign in a video sequence, information such as class label, type of sign (one-

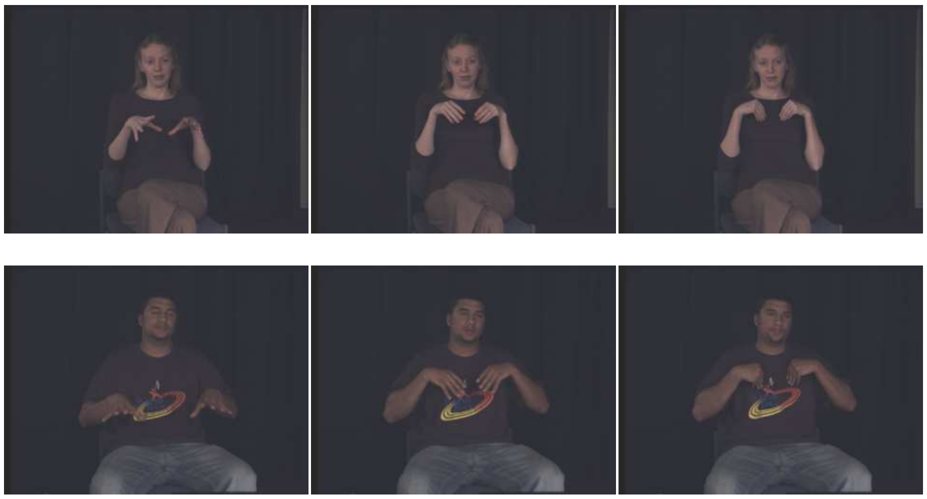


Figure 2.1: Two examples of the sign ADOPT from the ASL Lexicon Video Dataset. The top example is performed by signer Liz while the bottom is performed by signer Tyler. From left to right, we show the first frame, a middle frame and the last frame.

handed or two-handed), start/end frame for that sign, signer ID and the locations of the hands.

A subset of ASLLVD, named TL1113, was generated for the proposed sign recognition system. The dataset TL1113 consists of 1,113 unique signs. Each sign has one video example performed by the native signers Liz and Tyler separately; thus, there are 2,226 video sequences in total. Out of the four camera views recorded, only the 60fps, 640x480 frontal view is used in the dataset TL1113. Figure 2.1 shows two examples of the sign ADOPT performed by signers Liz and Tyler separately in the ASL Lexicon Video Dataset (ASLLVD).

2.2.1.2 Introduction of the Dataset GVD and Its Subset: G1113

More than 3,000 signs are contained in the *Gallaudet Dictionary of American Sign Language* [14]. Each sign is performed by a signer once; thus, there are more than 3,000 sign videos in total. These videos are recorded at 30 frames per second

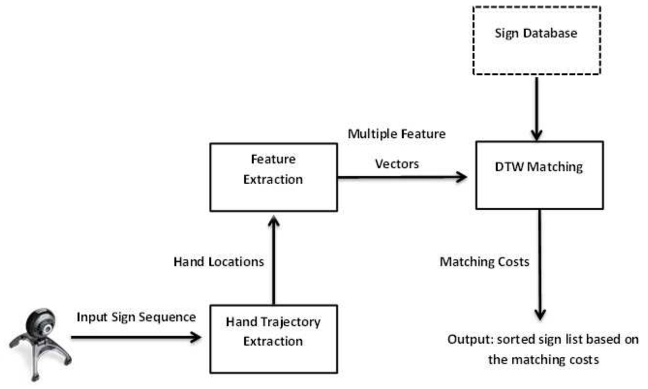


Figure 2.2: The flowchart of the proposed sign recognition system.

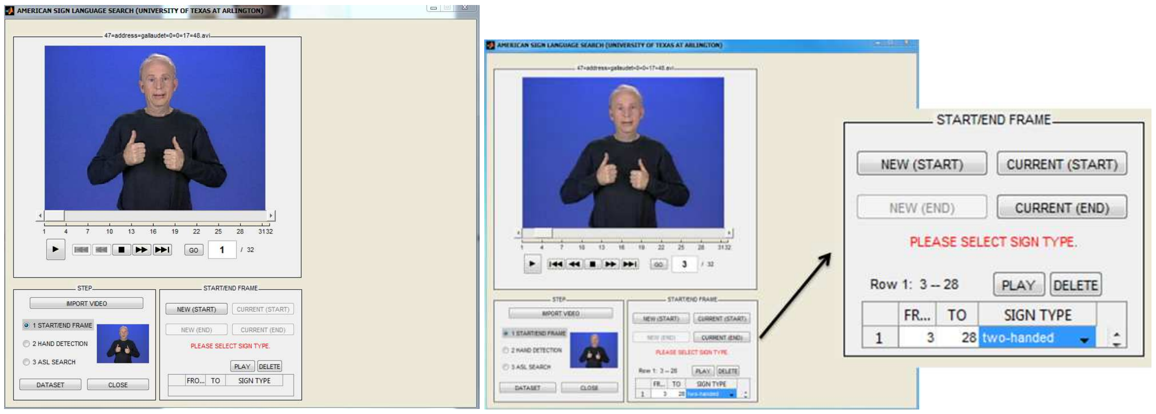
and a resolution of 240x352 pixels per frame. We choose the same 1,113 unique signs as the TL1113 dataset to generate the subset G1113.

The TL1113 and G1113 datasets together form the video database used for the sign recognition system. Thus, the final database includes 1,113 unique signs and each sign is performed by three signers.

2.2.2 Flowchart of the Sign Recognition System

Sign lookup is treated as a video database retrieval problem. The query sign is compared with each sign in the database, and the most similar database matches are returned to the user. The key problem is how to calculate the similarity between the query sign and the model sign in the database accurately. The solution adopted by the proposed sign recognition system is to match the hand trajectory of the query and the model trajectory using the well-known Dynamic Time Warping (DTW) algorithm.

Following the system flowchart introduced in Figure 2.2, the proposed sign recognition system can be further broken down into the following steps:

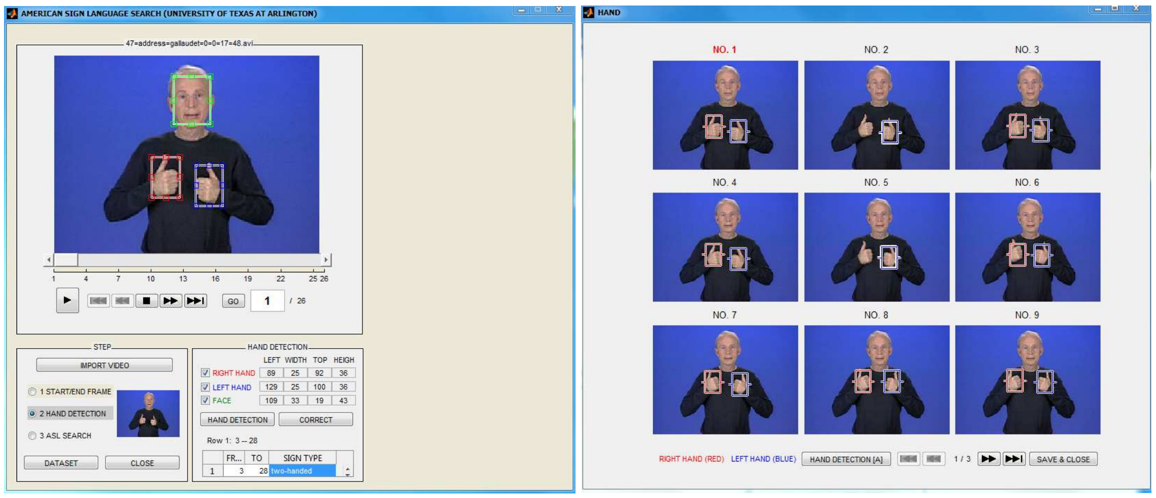


(a) Video submission

(b) Marking start and end frames

Figure 2.3: The video submission step in the proposed sign recognition system.

- When a user encounters an unknown sign, the user can perform the sign in front of a webcam or submit an existing video including that sign as a query to the system. In Figure 2.3a, a sign (ADDRESS) video is submitted to the system.
- Then, the system asks the user to mark the start and end frames of the actual sign in the video and to indicate whether the sign is one-handed or two-handed. In the START/END FRAME part of the GUI (Figure 2.3b), we can see that the submitted sign starts from the 3-th frame and ends at 28-th frame.
- At the next step, the system first asks the user to annotate the locations of the hands at the first frame and then track the hands in the remaining frames. The left side of Figure 2.4 shows the hands initialization step. After the initialization, the user clicks the HAND DETECTION button, and the system will generate the hand tracking results shown in the right side of Figure 2.4. The hand detection/tracking is still an open problem in the computer vision community, and no method guarantees 100% detection accuracy. Thus, the system allows the user to correct the false detections manually.



	LEFT	WIDTH	TOP	HEIGH
✓ RIGHT HAND	89	25	92	36
✓ LEFT HAND	129	25	100	36
✓ FACE	109	33	19	43

Row 1: 3 -- 28

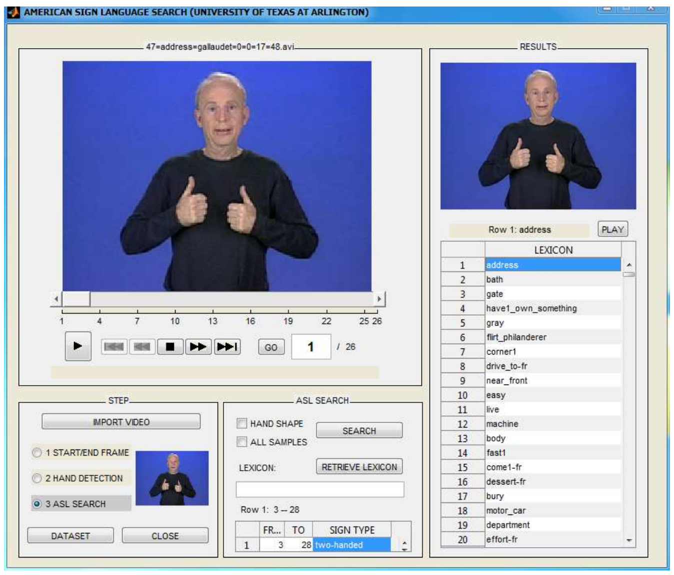
	FR...	TO	SIGN TYPE
1	3	28	two-handed

Figure 2.4: Hand tracking and correction step in the proposed sign recognition system.

- After hand detection results have been approved by the user, the system computes the similarity between the query sign and all database signs using the Dynamic Time Warping (DTW) similarity matching algorithm. The system ranks all 1,113 distinct signs in the database in decreasing order of similarity to the query. Once the signs have been ranked, the system presents the user with an ordered list of the best matching signs. The user then views the results, starting from the highest-ranked sign, until encountering the video displaying the actual sign of interest. Figure 2.5 shows the most similar matching sign is ADDRESS which is exactly the query sign.
- When the user identifies the correct database sign, the user can readily view any additional information associated with that sign. Currently, our signs are labeled with very rough English glosses.

2.2.3 Hand Trajectory Extraction Algorithm

As far as we can determine, detecting hands automatically is still an open problem in the computer vision community. So the proposed system resorts to the



Row 1: 3 -- 28

	FR...	TO	SIGN TYPE
1	3	28	two-handed

Row 1: address

	LEXICON
1	address
2	bath
3	gate
4	have1_own_something
5	gray
6	flirt_philanderer
7	corner1
8	drive_to-fr
9	near_front
10	easy
11	live
12	machine
13	body
14	fast1
15	come1-fr
16	dessert-fr
17	bury
18	motor_car
19	department
20	effort-fr

Figure 2.5: The step of matching the query with models in the proposed sign recognition system.

user interaction to achieve the hands' locations. The system first requests the user to annotate the hands' locations in the first frame. Then a tracking algorithm is adopted to search the hands' locations in the rest of the frames. Finally, the user's corrections are accounted for failed tracking cases. In this section, we describe the tracking algorithm, which is shown in Algorithm 1, used in the system.

In Algorithm 1, the similarity measure between two rectangular areas is defined by the correlation coefficient. Given two matrices A and B, the correlation coefficient is calculated using the following equation:

$$r = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{\left(\sum_m \sum_n (A_{mn} - \bar{A})^2\right) \sum_m \sum_n (B_{mn} - \bar{B})^2\right)}} \tag{2.1}$$

where $\bar{A} = \frac{\sum_m \sum_n A_{mn}}{mn}$ and $\bar{B} = \frac{\sum_m \sum_n B_{mn}}{mn}$.

Algorithm 1 Hand Tracking Algorithm

```
Input: A sequence of images $V = (I_1, ..., I_m)$.
Output: A sequence of hand locations $L = (L_1, ..., L_m)$.
Initialization: the user annotates the hands' location in 1-th frame $I_1$.
for $i = 2$ to $m$ do
    extract the hand area, indicated as $h_{i-1}$, in the previous frame $I_{i-1}$.
    search the most similar area to $h_{i-1}$ in the current frame $I_i$ and assign this area to $L_i$.
end for
```

2.2.4 Features and Normalization

Let X be a video of a sign. We denote by $|X|$ the number of frames in the video, and by $X(t)$ the t-th frame of that video, t ranging from 1 to $|X|$. From sign X we extract the following location features:

- $L_d(X,t)$ and $L_{nd}(X,t)$: The (x,y) centroid respectively of the dominant hand and non-dominant hand of the signer at frame t.
- $L_\delta(X,t)$: The relative position of the dominant hand with respect to the non-dominant hand at frame t. $L_\delta(X,t) = L_d(X,t) - L_{nd}(X,t)$

For notational convenience, all features referring to the non-dominant hand are set to zero vectors for one-handed signs. In defining location features, the choice of coordinate system is important. To account for differences in translation and spatial scale between the query video and the matching training videos, we use a face-centric coordinate system. We use the face detector provided by OpenCV to detect the face of each signer at the first frame of the sign. The coordinate system is defined so that the center of the face is at the origin, and the diagonal of the face bounding box has length 1. The same scaling factor is applied to both the x and the y direction.

Features $L_d(X,t)$, $L_{nd}(X,t)$, $L_\delta(X,t)$ are all defined in this normalized coordinate system.

2.2.5 Hand Trajectories Matching Using DTW Algorithm

Let X be a video of a sign. We can represent X as a time series $(X_1, ..., X_{|X|})$, where each X_t is simply a concatenation of the features extracted at frame t:

$$X_t = (L_d(X,t), L_{nd}(X,t), L_\delta(X,t)). \tag{2.2}$$

As a reminder, features $L_{nd}(X,t), L_\delta(X,t)$ are set to 0 for one-handed signs.

Dynamic Time Warping [21] is a commonly used distance measure for time series. Given two sign videos Q and X, DTW computes a warping path W establishing correspondences between frames of Q and frames of X:

$$W = ((q_1, x_1), ..., (q_{|W|}, x_{|W|})), \tag{2.3}$$

where $|W|$ is the length of the warping path, and pair (q_i, x_i) means that frame q_i of Q corresponds to frames x_i of X. A warping path must follow two constraints:

- boundary constraints: $q_1 = 1$, $x_1 = 1$, $q_{|W|} = |Q|$, $x_{|W|} = |X|$.
- monotonicity and continuity: $0 \leq q_{i+1} - q_i \leq 1$, $0 \leq x_{i+1} - x_i \leq 1$.

The cost $C(W,Q,X)$ of a warping path W is the sum of individual local costs $c(Q_{q_i}, X_{x_i})$, corresponding to matching each Q_{q_i} with the corresponding X_{x_i}:

$$C(W,Q,X) = \sum_{i=1}^{|W|} c(Q_{q_i}, X_{x_i}). \tag{2.4}$$

As local cost c, we use a combination of the individual Euclidean distances between the three features extracted from the two frames:

$$c(Q_{q_i}, X_{x_i}) = ||L_d(Q, q_i) - L_d(X, x_i)|| + ||L_{nd}(Q, q_i) - L_{nd}(X, x_i)|| + ||L_\delta(Q, q_i) - L_\delta(X, x_i)|| \tag{2.5}$$

The DTW distance $D_{DTW}(Q, X)$ between sign videos Q and X is defined as the cost of the lowest-cost warping path between Q and X:

$$D_{DTW}(Q, X) = \min_W C(W, Q, X) \tag{2.6}$$

The optimal warping path and the distance $D_{DTW}(Q, X)$ can be computed using dynamic programming, with a time complexity of $O(|Q||X|)$.

Algorithm 2 DTW Algorithm

```
**Input:** A sequence of model feature vectors $M = (M_1, ..., M_m)$, and a sequence of query feature vectors $Q = (Q_1, ..., Q_n)$.
**Output:** A global matching cost $D^*$.
Initialization: $D(1,1) = d(1,1)$
for $i = 2$ to $m$ do
    $D(i,1) = D(i-1,1) + d(i,1)$
end for
for $j = 2$ to $n$ do
    $D(1,j) = D(1,j-1) + d(1,j)$
end for
for $i = 2$ to $m$ do
    for $j = 2$ to $n$ do
        $D(i,j) = d(i,j) + \min(D(i-1,j), D(i,j-1), D(i-1,j-1))$
    end for
end for
$D^* = D(m,n)$
```

CHAPTER 3

HAND DETECTION ON SIGN LANGUAGE VIDEOS

In the computer vision community, hand detection has been a subject of study for several years due to its obvious applicability in domains such as sign language recognition, gesture recognition, and human-computer interfaces. Accurate detection of hands in still images or videos is still a challenging problem because of the variability of hand appearance. Hands do not have a fixed shape (Figure 3.1); thus, their shape is hard to describe computationally. This is in contrast to faces, for example, which have a well-defined shape (with two eyes, a nose, a mouth), and thus can be detected these days by commercial products such as cameras and cell phones. Colored gloves and magnetic trackers can give accurate detection results, but they are expensive and inconvenient (users have to wear special equipment).

In this chapter, we first review the classic hand detection work published in recent years and then choose four representative methods ([2, 13, 11, 12]) which are evaluated on three ASL datasets. These four methods are based on different sources of information: the appearance of the hand, the context parts which surround the hand and the motion information happening in sign videos. By combining these four methods, we propose a novel hand detector which fuses all of these different sources of information.

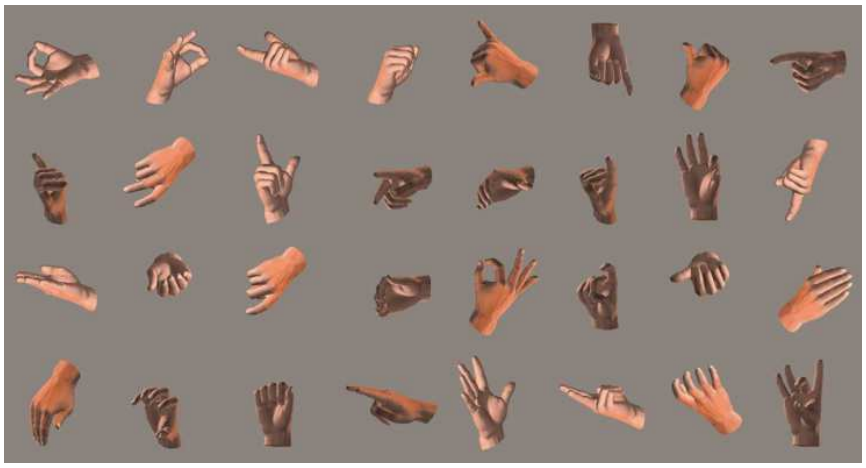


Figure 3.1: Various hand shapes in sign language videos.

3.1 Related Work

Methods for detecting hands in the computer vision community can be categorized into four groups: (1) appearance-based hand detection, (2) hand detection by using context information, (3) hand tracking and (4) hand shape detection.

3.1.1 Appearance-Based Hand Detection

Color information has been used in computer vision community for a long time [22, 23, 24]. Some methods use skin color information to localize and track hands in signing videos [25, 26]. Mittal et al. [2] describe a method for detecting hands and their orientation using skin color, hand shape, and context. Kölsch et al. [27] study view-specific hand posture detection with an object recognition method proposed by Viola and Jones [28]. Ong et al. [29] present a novel, unsupervised approach to train an efficient and robust detector which is capable of not only detecting the presence of human hands in an image but also of classifying the hand shape. Zhong et al. [30] evaluate four features for hand detection: color, temporal motion, gradient norm, and motion residue.

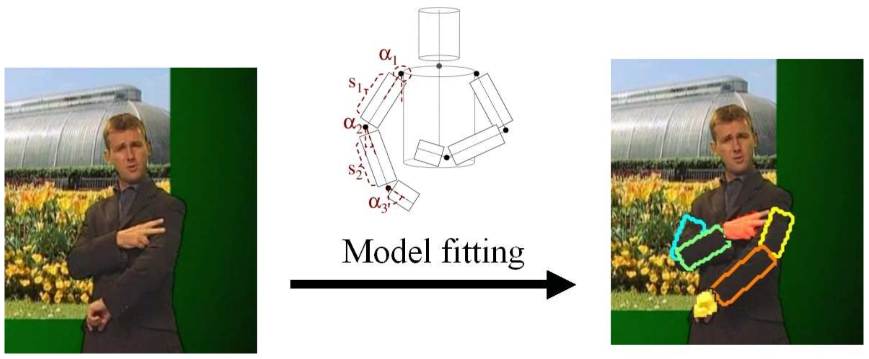


Figure 3.2: The framework of the pictorial structure (This figure comes from [1])

3.1.2 Hand Detection by Using Context Information

Karlinsky et al. [13] develop a chains model in which the relation between context features and the object of interest is modeled using an ensemble of feature chains. Pictorial structure models, originally introduced by Fischlet and Elschlager [31], provide a statistical model of objects. Using these pictorial structure models, objects in an image can be recognized and their constituent parts can be located in the image. Figure 3.2 shows the typical process of using pictorial structure to register the interest object. Buehler et al. [32, 1] use a generative model for upper body detection and propose a complete model which accounts for self-occlusion of the arms. Kumar [33] shows how this seemingly difficult problem can be solved by reducing it to an equivalent convex problem with a small, polynomial number of constraints. Pfister et al. [34] present a hand and arm tracker that detects joint positions in continuous sign language video sequences. Their method does not require manual annotation and performs tracking in real-time using a frame-by-frame random forest regressor.

3.1.3 Hand Tracking

Hand tracking can be categorized into two groups: 3D and 2D hand tracking.

3.1.3.1 2D Hand Tracking

Yuan et al. [11] propose a temporal filtering framework for hand tracking. In each frame, simple features such as color and motion residue are exploited to identify multiple candidate hand locations. The temporal filter then uses the Viterbi algorithm to select among the candidates from frame to frame. Trinh et al. [35] use binary quadratic programming to integrate appearance, motion and complex interaction between the hands. Morariu [36] describes a framework that uses probabilistic and deterministic networks and their AND/OR search space to detect and track the hands and feet of multiple interacting persons from a single camera view.

3.1.3.2 3D Hand Tracking

Zhou et al. [37] introduce the concept of eigen-dynamics and propose an eigen dynamics analysis (EDA) method to learn the dynamics of natural hand motion from labelled sets of motion captured with a data glove. Based on the EDA mode, a dynamic Bayesian network (DBN) is constructed to analyze the generative process of an image sequence of natural hand motion. Using the DBN, a hand tracking system is implemented. Martin et al. [38] present a novel model-based approach to 3D hand tracking from monocular video. The 3D hand pose, the hand texture and the illuminant are dynamically estimated through minimization of an objective function. Derived from an inverse problem of formulation, the objective function enables explicit use of texture temporal continuity and shading information, while handling important self-occlusions and time-varying illumination. Stenger et al. [39] set out a tracking framework, which is applied to the recovery of three-dimensional hand motion from an image sequence. The method handles the issues of initialization, tracking, and recovery in a unified way. Chen et al. [40] present a head and hands tracking method using a monocular camera for human machine interaction (HMI). The targets are tracked independently when they are far from each other; however, they are merged with dependent likelihood measurements in higher dimensions while they are likely to interrupt each other. Oikonomidis et al. [41] present a novel solution to the problem of recovering and tracking the 3D position, orientation and full articulation of a human hand from markerless visual observations obtained by a Kinect sensor. They treat this as an optimization problem, seeking for the hand model parameters that minimize the discrepancy between the appearance and 3D structure of hypothesized instances of a hand model and actual hand observations. The optimization problem is effectively solved using a variant of Particle Swarm Optimization (PSO). Sudderth et al. [42] develops probabilistic methods for visual tracking of a three-dimensional geometric hand model from monocular image sequences. Wang et al. [43] propose an easy-to-use and inexpensive system that facilitates 3D articulated user-input using hands. Their method uses a single camera to track a hand wearing an ordinary cloth glove that is imprinted with a custom pattern. The pattern is designed to simplify the pose estimation problem, allowing employment of a nearest-neighbor approach to track hands at interactive rates. Oikonomidis et al. [44] propose a method that relies on markerless visual observations to track the full articulation of two hands that interact with each other in a complex, unconstrained manner. They formulate this as an optimization problem whose 54-dimensional parameter space represents all possible configurations of two hands, each represented as a kinematic structure with 26 degrees of Freedom (DoFs). To solve the problem, they employ PSO, an evolutionary, stochastic optimization method with the objective of finding the two-hands configuration that best explains observations provided by an RGB-D sensor.

3.1.4 Hand Shape Detection

Shape is a very important information for vision-related tasks [45, 46, 47]. Athitsos el al. [48] propose a method for detecting shapes of variable structure in images with clutter. Variable structure means that some shape parts can be repeated an arbitrary number of times, some parts can be optional, and some parts can have several alternative appearances. A new class of shape models is introduced, called Hidden State Shape Models, that can naturally represent shapes of variable structure. A detection algorithm is described that finds instances of such shapes in images with large amounts of clutter by finding globally optimal correspondences between image features and shape models. Thayananthan el al. [49] compare two methods for object localization from contours: shape context and chamfer matching of templates.

3.2 Five Hand Detection Methods on Sign Language Videos

Although several hand detection methods have been published, a general solution for the hand detection problem still does not exist. Most of the existing hand detection methods focus on still images. Thus, they do not take advantage of the motion information which happens in almost every frame of the sign language videos. As far as we know, these methods' capabilities on ASL videos have not yet been discussed in any literature. Thus, in this chapter, we briefly introduce four representative hand detection methods and evaluate them on three ASL datasets. We also propose a novel hand detector by combining these four methods and compare the proposed method with the four detectors on the American sign datasets.

3.2.1 Skin and Motion Detector

This section introduces a skin and motion detector proposed in [12].

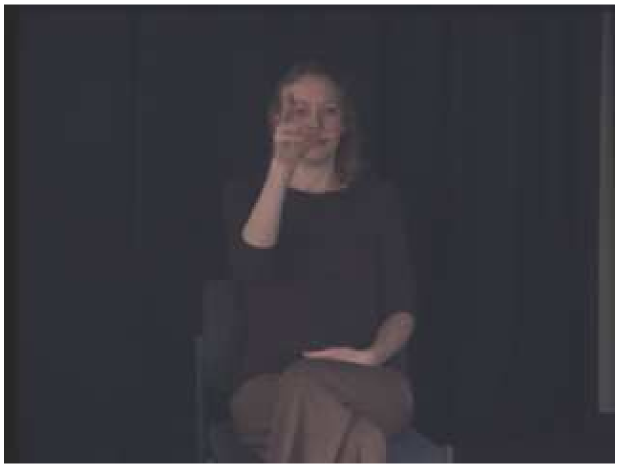


Figure 3.3: An example frame from a video of a sign.

3.2.1.1 Detecting Skin

Since human skin is relatively uniform in color, a statistical color model can be employed to compute the probability of every pixel being skin color. In [50], a skin color likelihood distribution and a non-skin color distribution, denoted as $P(r, g, b|\text{skin})$ and $P(r, g, b|\neg\text{skin})$, respectively, are proposed, in which the RGB color space is quantized to $32 \times 32 \times 32$ values. Based on these two distributions, the probability of a pixel, whose color vector is $[r, g, b]$, being skin is defined using Bayes rule:

$$P(\text{skin}|r, g, b) = \frac{P(r, g, b|\text{skin})P(\text{skin})}{P(r, g, b)} \tag{3.1}$$

Figure 3.4 shows an image visualizing the $P(\text{skin}|r, g, b)$ score computed for every pixel of the video frame shown on Figure 3.3.

3.2.1.2 Temporal Motion

Motion information is another useful cue for hand detection in gesture videos since a user needs to move at least one hand to perform a hand gesture.

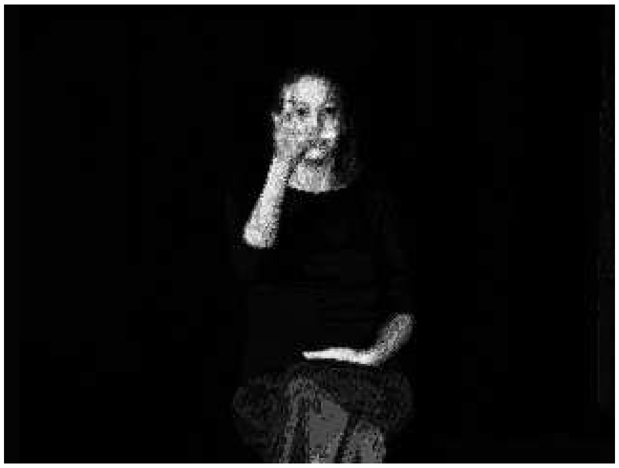


Figure 3.4: Skin score map of the image shown in Figure 3.3.

To detect motion, we used a simple method based on frame differencing. Other more sophisticated background subtraction methods such as Mixtures of Gaussian (MoGs) [51, 52] can be used instead, but the simple frame differencing method has worked sufficiently well in our experiments.

Frame differencing works as follows: let $I(x, y, i)$ denote the intensity value at pixel (x, y), at the i-th frame. By comparing $I(x, y, i)$ with $I(x, y, i - z)$ and $I(x, y, i + z)$, we compute a motion indicator value $M(x, y, i)$. Motion indicator value $M(x, y, i)$ is defined using the following equations:

$$\begin{aligned} I_1(x,y,i) &= |I(x,y,i) - I(x,y,i-z)| \\ I_2(x,y,i) &= |I(x,y,i) - I(x,y,i+z)| \\ M(x,y,i) &= min(I_1(x,y,i), I_2(x,y,i)) \end{aligned} \tag{3.2}$$

We compute the motion indicator value for each pixel and thus obtain the motion score image, referred to as M. Figure 3.5 shows a motion score image computed for the video frame shown in Figure 3.3.

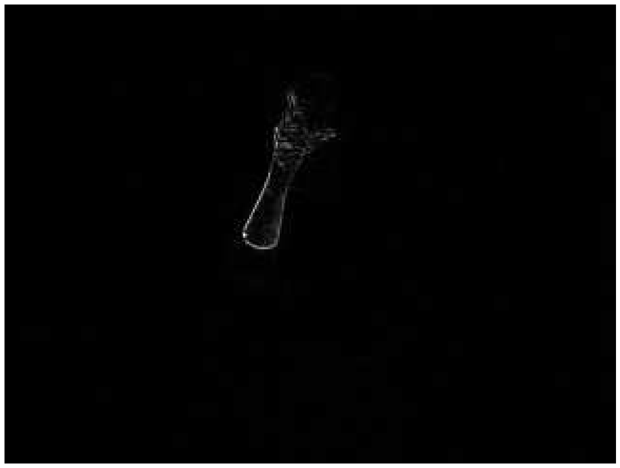


Figure 3.5: Motion score map of the image shown in Figure 3.3.

3.2.1.3 Detecting Hands Based on Skin and Temporal Motion

For every pixel in a frame, we can compute the skin indicator value and motion indicator value of the pixel using the methods described above. Let $S(x, y, i)$ denote the skin indicator value at pixel (x, y), in the i-th frame and $M(x, y, i)$ denote the motion indicator value at pixel (x, y), in the i-th frame. The combined skin and motion indicator value for this pixel is defined using the following equation:

$$A(x, y, i) = S(x, y, i) * M(x, y, i) \tag{3.3}$$

The skin and motion indicator value is computed for each pixel, and thus we obtain the skin and motion score image. Figure 3.6 shows a skin and motion score image computed for the video frame shown in Figure 3.3.

The most likely hand candidate is defined as the region which has the largest summation of values in image A. Once we find a candidate region, before we identify the next most likely region, we overwrite with value zero the skin color probabilities of all pixels in the candidate region we have just identified. This helps to avoid identifying multiple candidate regions with significant mutual overlap. To determine

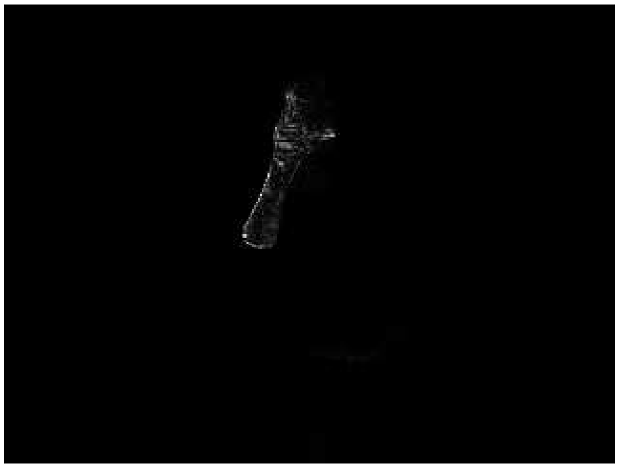


Figure 3.6: Skin and motion score map of the image shown in Figure 3.3.

the hand region's size, at every frame, the size of the hand is adjusted according to the size of the face. Faces are detected using the Viola-Jones method [28].

3.2.2 Motion Residue Detector

Hands typically undergo non-rigid motion because they are deformable articulated objects. This means that hand appearance changes more frequently from frame to frame compared to the appearance of other background objects. We can use this property to detect hands by identifying regions in each frame that have no good matches (in terms of appearance) among regions in the next frame. This is an idea proposed in [11].

Following the approach of [11], for every two consecutive frames, the first frame is partitioned into blocks (8x8), and then we try to find the best match for each block in the next frame by translation. Based on the best match for each block, motion residue is defined as the summation of differences in intensity level between the block and its best match in the next frame. Let A denote the one block in current frame and

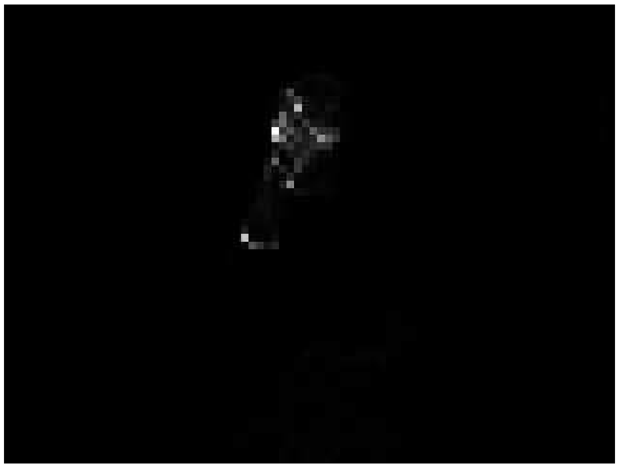


Figure 3.7: Motion residue score map of the image shown in Figure 3.3.

B denote the best match of A in the next frame. We can use the following equation to calculate residue:

$$R = \sum_{i \in \text{block}} (A_i - B_i)^2 \tag{3.4}$$

Every pixel in the block will be assigned as this residue. Because hands move nonrigidly in most cases, the blocks in a hand region tend to have high residues; therefore, we can use residue as a feature to detect hands. Hand candidates are identified as rectangular areas with the largest residue value.

Figure 3.7 shows a motion residue score image computed for the video frame shown in Figure 3.3.

3.2.3 Hand Detection Using Multi-proposals

Mittal et al. [2] describe a two-stage method for detecting hands and their orientation in unconstrained images. The first stage uses three complementary detectors (shape, context, and skin color) to propose hand bounding boxes. Each bounding box is then scored by the three detectors independently, and a second stage classifier is

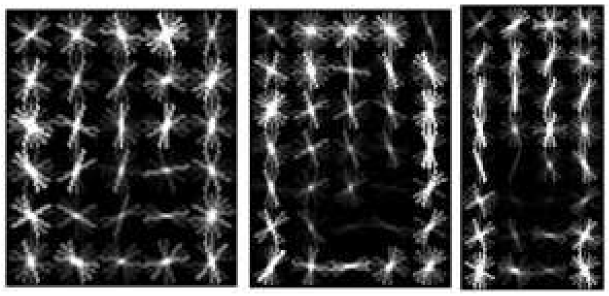


Figure 3.8: Root filters for the three components of the hand shape detector. The first two filters cover frontal pose and the third filter profile (This figure comes from [2]).

learned and used to compute a final confidence score for the proposals using these features.

3.2.3.1 Hand Shape Detector

The hand shape detector uses the deformable part model proposed by Felzenszwalb et al. [53], which is based on histogram of oriented gradients (HOG) features [54], to propose hand bounding boxes. In calculating bounding box scores, the detector uses a mixture model comprised of three complimentary components (Figure 3.8) that represent different properties of the hand. The set of bounding boxes B_{HD} in any given image is composed of the bounding boxes for which the hand detector score $\beta_{HD}(b)$ [53] is above a threshold t_h: $B_{HD} = \{b \in B | \beta_{HD}(b) > t_h\}$, where B is the set of all detected hand bounding boxes, and the threshold t_h is chosen so that the detector achieves 90% recall on the training set.

3.2.3.2 Context Detector

The context detector examines the surrounding area, or context, of a potential hand location when proposing bounding boxes since areas surrounding the hand may be more easily recognized than the hand itself. As with the hand shape detector

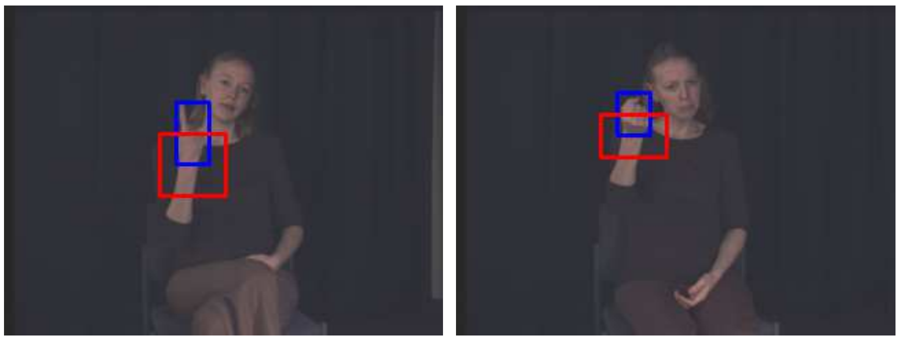


Figure 3.9: Context captured around the hand bounding box. The blue rectangle shows the hand bounding box and the red rectangle shows the extended box used to capture context around the hand. The context is captured over a region having the same height and twice the width as the hand.

described in section 3.2.3.1, this detector also uses the deformable part model of [53] to learn the context and is trained by expanding the annotated hand bounding boxes in training examples to include the surrounding area (Figure 3.9). The final hand bounding boxes are shrunk by the context boxes. For any given image, the set of context box proposals B_{CD} is comprised of those whose score $\beta_{CD}(b)$ [53] is above a threshold t_c: $B_{CD} = \{b \in B | \beta_{CD}(b) > t_c\}$, where B is the set of all detected hand bounding boxes. As with the above hand shape detector, the threshold is chosen so that the detector achieves 90% recall on the training set.

3.2.3.3 Skin-Based Detector

The third component in the multiple proposal method is a skin-based detector, which involves two steps: identification of skin regions and use of those regions to generate hand hypotheses. In the first step, a face detector is used to locate the face, and a histogram of face pixel colors is used to create a simple classifier of color likelihood. Doing so helps account for unusual, image-specific lighting conditions or color balances. To create the classifier, two confidence thresholds for likelihood, similar to hysteresis in canny edge detection, are first learned. Pixels whose skin likelihood falls above the upper threshold are classified as skin. Those whose likelihood falls between the thresholds are only classified as skin if a neighboring pixel's skin likelihood is above the upper threshold. The authors use cross validation on ground truth segmentation to learn the two thresholds. A bootstrapping stage is then performed to update the color likelihood classifier by using the color of the neighboring pixels, and the process is repeated.

The second step involves fitting lines to skin regions by using a Hough transform. Hands are then hypothesized at both ends of these lines. If the skin region resembles a blob, such as when the hand is visible but not the arms, the whole skin region is hypothesized as a hand instead of each end. The face skin pixels are ignored and not included in any hypotheses. The detection score is given by the proportion of skin pixels to other pixels in the largest super-pixel [55] within the hypothesized box.

3.2.3.4 Candidate Boxes Classification

The second stage of hand detection is the classification of hypotheses. All hand bounding boxes proposed in the first stage are scored by all three hand proposal methods, and the three scores are combined into a score vector. A linear support vector machine (SVM) classifier is then used to generate a final confidence score. The three hand proposal methods ensure good recall, and the discriminative classification ensures good precision.

3.2.4 Chains Model for Hand Detection

When the appearance information of the target is not reliable, such as low resolution hands, the target cannot be accurately recognized by the appearance of itself. For this case, the context information becomes the only valuable source. The

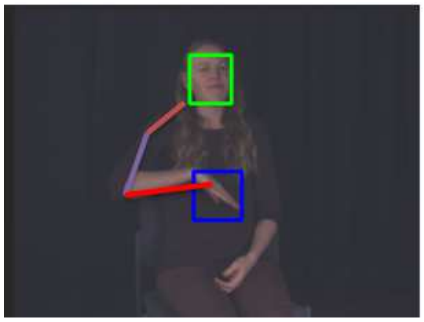


Figure 3.10: Detecting hand by extending from face.

chains model is proposed to take advantage of the context information of the target to detect the target object, which describes an assembly of feature chains of arbitrary length that start at some known reference point on the object, such as an automatically detected face, and terminate at the detection target, such as the hand. More specifically, the chains model method first detects the face and extracts some context features of the target image and then calculates the posterior probability of seeing the hand at a certain position by marginalizing over chains of context features. Thus, the hand posterior probability image can be determined by computing the hand posterior probability of seeing the hand at each position in the target image.

3.2.5 Combination of Four Hand Detectors

The previous four hand detection methods are complementary: residue [11] utilizes the fact that hands are articulated objects and thus undergo non-rigid motion which means that hand appearance changes more frequently from frame to frame, compared to appearance of the clothes, the face and background objects; the skin and motion detector [12] combines skin and temporal motion features to detect the hand; instead of directly exploring the features of the hand itself, the chains model [13] uses the context supplied by surrounding parts of the hand; and [2] combines the hand shape detector, hand context detector and skin-based detector. A natural improvement is to fuse these four complementary methods. The proposed hand detector describes a two-stage method for detecting hands on sign language videos. The first stage uses a simple hand detector based on skin and temporal motion features to generate hand candidate boxes in the target image. Then several feature scores are computed for each candidate box based on these four hand detection methods [11, 12, 13, 2]. Finally, the feature scores associated with each candidate box are fed to a pre-trained classifier to compute a final score for each box.

3.2.5.1 Skin and Motion Detector

Sliding window classification is a dominant paradigm in object detection. However, scanning each window in the target image is time consuming. To increase the efficiency, a commonly used skill in real life is to exclude as many unlikely candidate windows as possible in the preprocessing step. In the proposed method, we use a simple skin and motion [12] detector to exclude the non-skin and static areas in the target image. The background areas in sign language videos are quite clean and different from the skin color so they can be easily eliminated by skin feature. Since the only moving object in the videos is the upper body of the performer, a temporal motion filter removes the static areas. More specifically, the skin detector computes for every image pixel a skin likelihood term. The motion detector computes a score map by frame differencing. The hand likelihood image is the multiplication of the motion score map and skin likelihood image. Using the integral image [28] of the hand likelihood image, we efficiently compute for every subwindow of some predetermined size the sum of pixel likelihoods in that subwindow. Then we extract the K subwindows with the highest sum such that none of the K subwindows may include the center of another of the K subwindows. K is set to 30 in the proposed method. From our experience, 30 candidates are enough to cover the real hands. In this way, we reduce the candidate boxes in a target image from tens of thousands to 30. In Figure 3.11, we show the top five hand bounding boxes detected by the skin and motion detector. We can see that the top five candidates include the real hand in these frames.

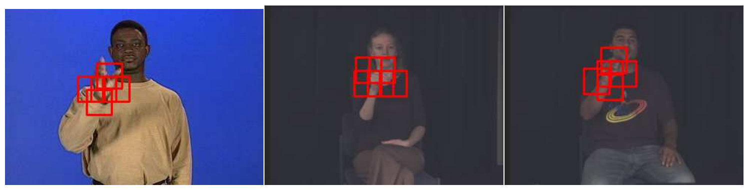


Figure 3.11: From left to right, we show three frames with the top five hand bounding boxes detected by skin and motion from **G100**, **L100** and **T100**, respectively. The hand bounding boxes are indicated by red rectangles.

3.2.5.2 Hypothesis Classification

The hypotheses proposed by the skin and motion detector are evaluated using a second stage classifier. We calculate twelve features for each hypothesized bounding box, and then a pre-trained SVM [56] classifier is applied to calculate the final confidence score for the hypothesis. Assuming there is a hypothesized bounding box B with the position $[l, t, w, h]$, where $[l, t]$ are the coordinates of the left-top point and $[w, h]$ is the width and height of the box, respectively, twelve features associated with this hypothesis are computed as follows:

Skin feature. We use a generic skin color histogram [50] to compute the skin likelihood image, referred to as S. The skin feature of the hypothesized bounding box B is calculated as the average of the sum of pixel likelihoods in the bounding box, which can be expressed in the following equation:

$$\mathrm{F}_{\text{skin}} = \frac{1}{wh} \sum_{\substack{x \in [l, l+w-1] \\ y \in [t, t+h-1]}} S(x,y) \tag{3.5}$$

Motion features. Motion information is another discriminant cue for hand detection in sign videos since a user needs to move at least one hand to perform a hand gesture. The calculation of the motion score image has been introduced in Section 3.2.1.2. Let M denote the motion score image. The motion feature of the hypothesized bounding box B is calculated as the average of the sum of motion indicator values in the bounding box, which can be expressed in the following equation:

$$\mathrm{F}_{\text{motion}} = \frac{1}{wh} \sum_{\substack{x \in [l, l+w-1] \\ y \in [t, t+h-1]}} M(x,y) \tag{3.6}$$

By choosing different values of the frame gap z in Equation 3.2, we can calculate multiple motion indicator images, thus giving us multiple motion features for the bounding box. The value of z spans from 1 to 5 in our experiments, so there are 5 motion features for one hypothesis.

Gradient feature. Before computing the gradients, we use a Gaussian mask to smooth each frame. Then the gradients are computed using a simple 1-D $[-1, 0, 1]$ mask. Let $I(x, y, i)$ denote the intensity value at pixel (x, y), at the i-th frame. The gradient at this pixel, $G(x, y, i)$, is computed using the following equations:

$$\begin{aligned} dx &= I(x-1, y, i) - I(x+1, y, i) \\ dy &= I(x, y-1, i) - I(x, y+1, i) \\ G(x, y, i) &= \sqrt[2]{dx^2 + dy^2} \end{aligned} \tag{3.7}$$

After calculating the gradient for every pixel in the frame, we thus have the gradient score image (Figure 3.12), referred to as G. The gradient feature of the

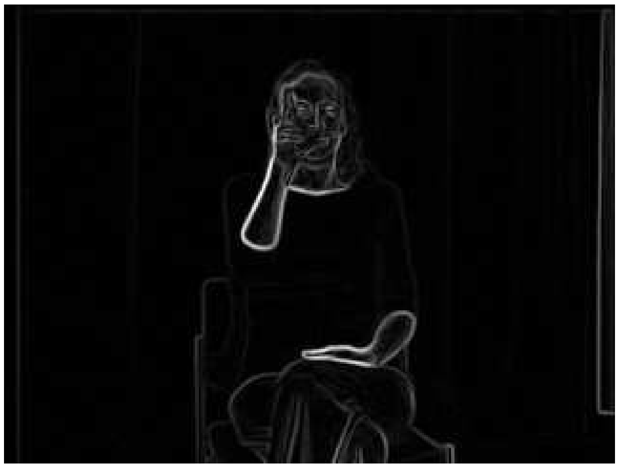


Figure 3.12: Gradient score map of the image shown in Figure 3.3.

hypothesized bounding box B is calculated as the average of the sum of gradient indicator values in the bounding box, which can be expressed in the following equation:

$$\mathrm{F}_{\text{gradient}} = \frac{1}{wh} \sum_{\substack{x\in[l,l+w-1]\\ y\in[t,t+h-1]}} G(x,y) \tag{3.8}$$

Residue feature. The calculation of the residue score map is introduced in Section 3.2.2, so we will not repeat it here again. Let R denote the residue score map, and the residue feature of the hypothesized bounding box B is calculated using the following equation:

$$\mathrm{F}_{\text{residue}} = \frac{1}{wh} \sum_{\substack{x\in[l,l+w-1]\\ y\in[t,t+h-1]}} R(x,y) \tag{3.9}$$

Chains model feature. The intuition of the chains model [13] is to detect a difficult part (hand) by extending from an easy reference part (face). We use the source code provided by Karlinsky et al. to compute the hand posterior probability image (indicated as C), which is introduced in section 3.2.4. The chains model feature of the hypothesized bounding box B is calculated using the following equation:

$$\mathrm{F}_{\text{chain}} = \frac{1}{wh} \sum_{\substack{x \in [l, l+w-1] \\ y \in [t, t+h-1]}} C(x, y) \tag{3.10}$$

Hand shape and context features. Let B_{shape} denote the hand bounding boxes set returned by the hand shape detector. In order to calculate the hand shape feature of the hypothesized bounding box B, we first compute the overlap between B and every box in the set B_{shape} and then the hand shape feature is assigned as the score of the box in B_{shape} with the maximum overlap with B. The calculation of the hand context feature is the same as the hand shape feature.

3.3 Dataset Description

The **ASL1113x3** dataset includes three components: **G1113**, **L1113** and **T1113**. Each component contains same 1113 distinct signs and each sign has one video example. All of them include two kinds of signs: one-handed signs and two-handed signs (Figure 3.14 and Figure 3.13). **L1113** and **T1113** are single signer datasets while **G1113** is a multi-signers dataset. Annotations for hand regions and face regions are also available for all videos.

We randomly selected 100 signs from the **G1113** dataset to generate the **G100** dataset. We also picked the same 100 signs from the **L1113** and **T1113** datasets to create the **L100** and **T100** datasets respectively. **G100**, **L100** and **T100** are combined together to generate **ASL100x3**. Information about the number of videos and the number of frames for these subsets, separated into one-handed and two-handed instances, can be found in Table 3.1.

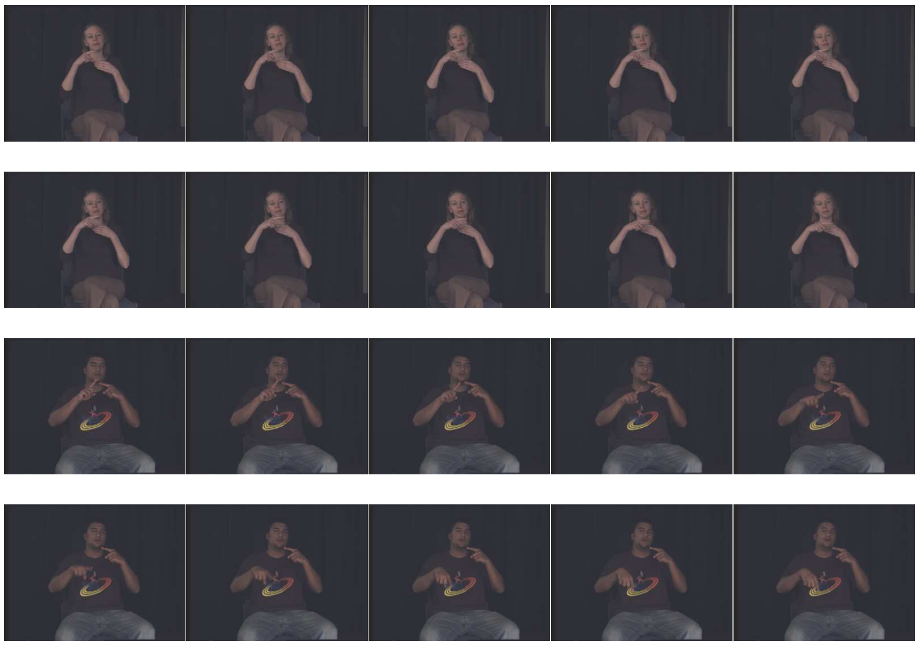


Figure 3.13: Video frames of two two-handed signs from **L100** (top two rows) and **T100** (bottom two rows).

3.4 Hand Detection Experiments

3.4.1 Experimental Protocol

We evaluated five methods on the **ASL100x3** dataset. The first one is the multiple proposals method [2], which combines a skin detector, context detector, and hand shape detector. We use the model which is trained by the authors on an external dataset to do the test.

Table 3.1: Description of datasets

	G100	**L100**	**T100**
# of one-handed videos	42	42	42
# of one-handed frames	902	1276	1197
# of two-handed videos	48	48	48
# of two-handed frames	1337	1945	1735

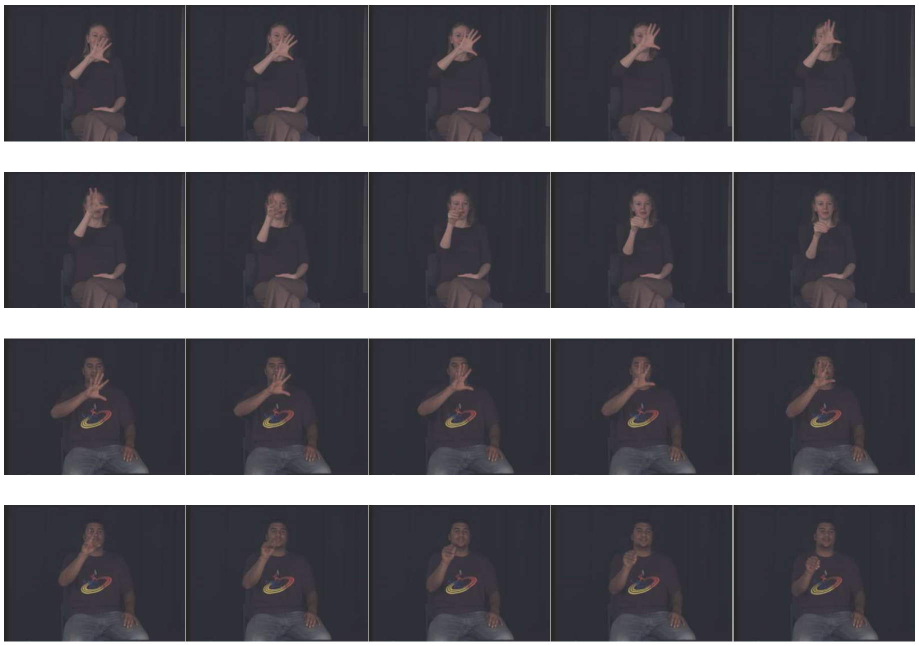


Figure 3.14: Video frames of two one-handed signs from **L100** (top two rows) and **T100** (bottom two rows).

The second method is the chains model method [13], which detects the hand by linking a path from the face to the hand. Each component in **ASL100x3** is tested separately, using one of the others as a training set. **T100** was used to train the model for both the **G100** and **L100** datasets while **L100** was used as training dataset for **T100**.

The skin and motion detector and residue detector do not require a training step, so all videos are used for testing.

The combination of four methods is tested with two experimental settings. The first one is called user-dependent setting where the training and test come from the same dataset. This user-dependent setting can be further categorized into two groups, combination20 and combination50 based on the number of sign videos used as training. Take combination20 as an example, to test dataset **G100** we randomly choose 20 signs from **G100** and the rest is used for test. The second one is called user-independent setting. The test dataset is trained on the other two datasets; for example, to test dataset **G100**, the training datasets consist of **L100** and **T100**.

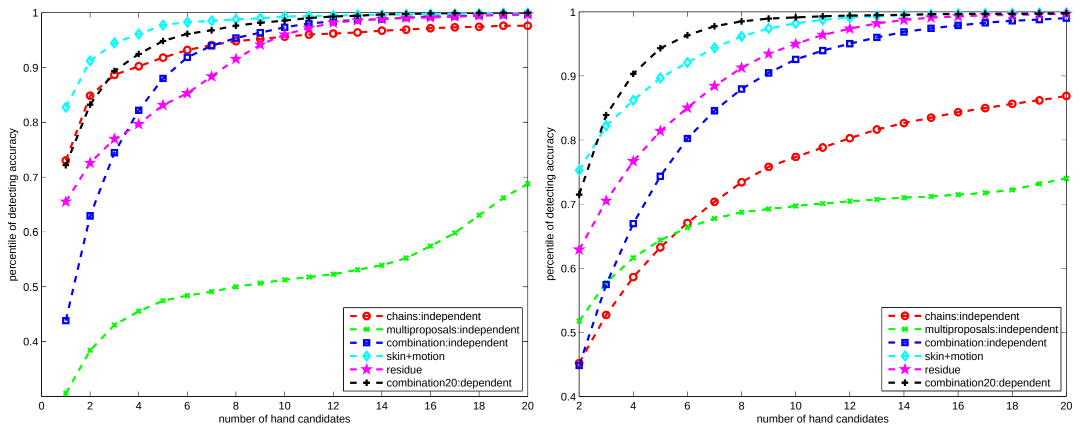


Figure 3.15: Accuracy vs Top k bounding boxes retrieval on the **ASL100x3** dataset using combination20 and other competitive methods. The left side is the result of one-handed signs and the right side of two-handed signs. The x-axis corresponds to the top k hand candidates, while the y-axis corresponds to detection accuracy.

3.4.2 Measure of Accuracy

The hand detection is considered to be correct if it is within half-face width from the ground-truth location of the hand. We report the detection performance within the top k (k is from 1 to 20 for one-handed cases and from 2 to 20 for two-handed cases) hand candidates per frame. So if the ground truth is within half the face width of one of the top k candidates, it is considered accurately located. The one-handed and two-handed test cases are examined separately.

Figure 3.15, 3.16, and 3.17 show the evaluation results on the **ASL100x3** dataset. The left side is the result of one-handed sign videos, and the right side is the result of two-handed sign videos.

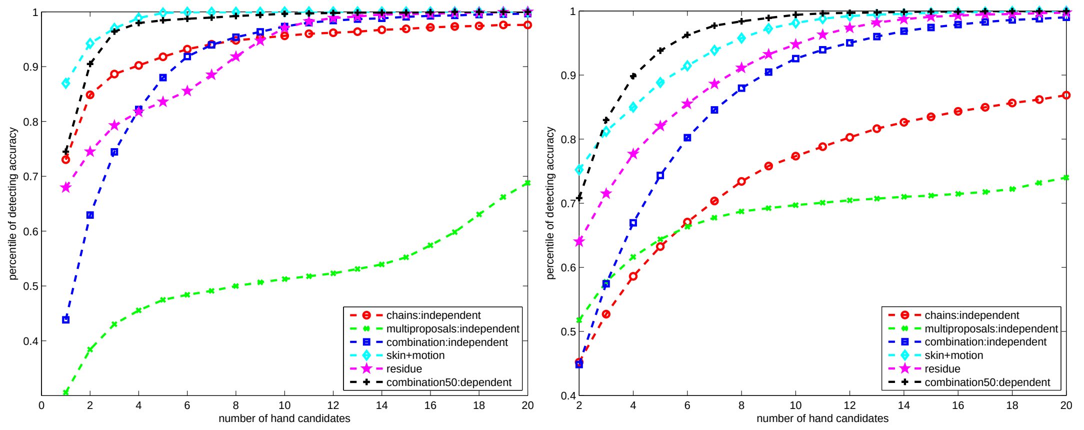


Figure 3.16: Accuracy vs Top k bounding boxes retrieval on the **ASL100x3** dataset using combination50 and other competitive methods. The left side is the result of one-handed signs and the right side of two-handed signs. The x-axis corresponds to the top k hand candidates, while the y-axis corresponds to detection accuracy.

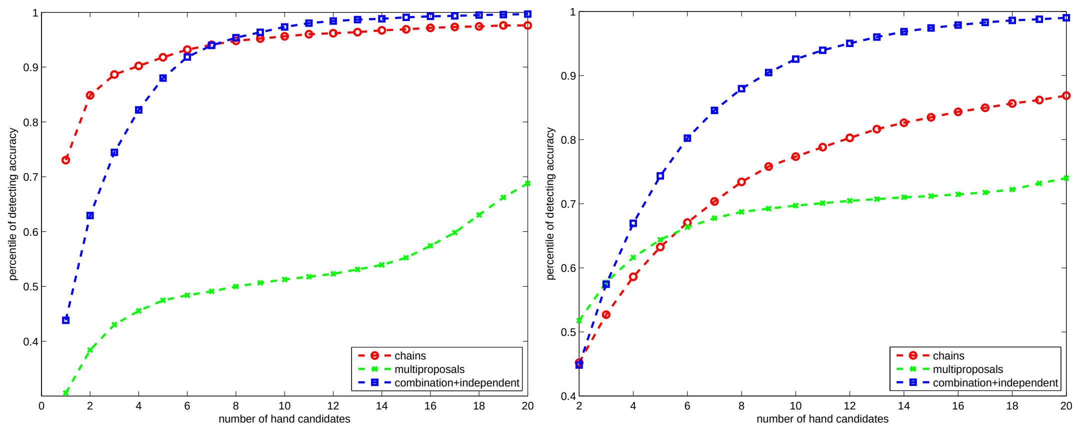


Figure 3.17: Accuracy vs Top k bounding boxes retrieval on the **ASL100x3** dataset using chains model, multiple proposals, and combination method with user independent setting. The left side is the result of one-handed signs and the right side of two-handed signs. The x-axis corresponds to the top k hand candidates, while the y-axis corresponds to detection accuracy.

Table 3.2: p-value between the method of combination20 and skin plus motion

number of candidate	p-value of combination20 vs skin+motion	
	one-handed signs	two-handed signs
1	2.9e-28	5.2e-51
2	2.9e-24	1.9e-10
3	5.7e-16	8.6e-4
4	4.4e-12	2.9e-21
5	1.7e-10	3.5e-35
6	6.6e-8	1.5e-38
7	3.2e-6	6.1e-34
8	4.1e-4	8.4e-24
9	7.1e-3	8.4e-15
10	5.4e-3	1.1e-8
11	1.3e-1	1.1e-4
12	3.8e-1	1.4e-2
13	4.5e-1	2.0e-1
14	6.5e-1	7.0e-1
15	6.2e-1	7.5e-1
16	2.5e-1	6.0e-1
17	9.6e-2	4.5e-1
18	1.0e-1	8.8e-2
19	8.3e-2	1.6e-1
20	1.6e-1	2.5e-1

In figure 3.15 and 3.16, we compare the combination methods with the experimental settings of combination20 and combination50 with other competitors respectively. Both of them show that the combination methods with user-dependent setting (combination20 and combinatino50) work worse than skin and motion detector on one-handed signs but better than skin and motion detector on two-handed signs. Table 3.2 show the p-values between the methods of combination20 and skin plus motion, while table 3.3 is about combination50 and skin plus motion.

We have three methods with user-independent training in Figure 3.17. These three methods are: chains model, multiple proposals, and the combination method

Table 3.3: p-value between the method of combination50 and skin plus motion

number of candidate	p-value of combination50 vs skin+motion	
	one-handed signs	two-handed signs
1	1.7e-19	7.3e-59
2	7.2e-5	2.5e-11
3	3.9e-1	2.0e-3
4	4.7e-2	1.1e-21
5	7.2e-5	5.6e-29
6	9.3e-5	6.6e-35
7	4.6e-4	2.4e-31
8	3.9e-3	1.9e-20
9	2.0e-2	9.5e-13
10	2.5e-2	1.1e-11
11	4.5e-2	6.8e-8
12	8.3e-2	8.1e-5
13	1.6e-1	1.7e-3
14	1.6e-1	1.5e-3
15	1.6e-1	2.2e-2
16	3.2e-1	1.1e-1
17	NaN	2.1e-2
18	NaN	1.0e-1
19	NaN	1.8e-1
20	NaN	3.2e-1

with user-independent setting. Our combination method gives best result on both one-handed sign videos and two-handed sign videos.

3.5 DSTW Experiments

Sign lookup can be treated as a video database retrieval problem. When the user encounters an unknown sign, the user provides a video example of that sign as a query, so as to retrieve the most similar signs in the database. A necessary component of such a sign lookup system is a similarity measure for comparing sign videos. Given a query video of a specific sign, the similarity measure should assign high similarity values to videos from the same sign, and low similarity values to videos from other signs. This paper evaluates a state-of-the-art video-based similarity measure called Dynamic Space Time Warping (DSTW) for the purposes of sign retrieval. Dynamic Space-Time Warping (DSTW) is an extension of Dynamic Time Warping (DTW), designed explicitly to work with multiple candidate hand locations per frame.

3.5.1 Hand Detection and Feature Extraction

The DSTW algorithm has been designed to accommodate multiple hypotheses for the hand location in each frame. Therefore, we can afford to use a relatively simple hand detection scheme. The hand detectors used in this experiments are introduced in Section 3.2. They are: chains' model, multi-proposals, skin and motion, residue, and the combination of previous four methods. Each hand detector generates K hand candidates for each frame. K is set as 5 in this experiment.

As described above, for every frame j of the query sequence, the hand detector identifies K candidate hand regions. For every candidate k in frame j a 2D feature vector $Q_{jk} = (x_{jk}, y_{jk})$ is extracted. The 2D position (x, y) is the region centroid. To account for differences in translation and spatial scale between the query video and the matching training videos, we use a face-centric coordinate system. We use the face detector provided by OpenCV to detect the face of each signer at the first frame of the sign. The coordinate system is defined so that the center of the face is at the origin, and the diagonal of the face bounding box has length 1.

3.5.2 DSTW Algorithm

Let $M = (M_1, ..., M_m)$ be a model sequence in which each M_i is a feature vector. In our experiment, M_i is the 2D centroid position of the dominant hand. Let $Q = (Q_1, ..., Q_n)$ be a query sequence. In the regular DTW framework, each Q_j would be a feature vector, of the same form as each M_i. However, in dynamic space-time warping (DSTW), we want to model the fact that we have multiple candidate feature vectors in each frame of the query. For example, if the feature vector consists of the position of the hand in each frame, and we have multiple hypotheses for hand location, each of those hypotheses defines a different feature vector. Therefore, in DSTW, Q_j is a set of feature vectors: $Q_j = Q_{j1}, ..., Q_{jK}$, where each Q_{jk}, for $k \in 1, ..., K$, is a candidate feature vector. K is the number of feature vectors extracted from each query frame. In our algorithm we assume K is fixed, but in principle K may vary from frame to frame. The detailed algorithm for calculating the similarity between Q with M can be found at [57].

3.5.3 Measure of Accuracy

The **ASL100x3** dataset includes three subsets: **G100**, **L100** and **T100**. Each subset contains 100 signs and each sign has one video example. To test the sign in the subset **G100**, the **L100** and **T100** datasets are used as database videos. The same procedure applies to testing the **L100** and **T100** subsets.

Table 3.4, 3.5 and 3.7 show the results. In these tables, the first column specifies different values of rank of correct sign. For each such value, x, for each of the three hand detectors, we show the percentage of test signs for which the correct sign is among the top x matches. We choose for each class its highest-ranking exemplar. We then rank classes according to the score of the highest ranking exemplar for each class. Table 3.6 shows the p-values between combination20 and skin plus motion and combination50 and skin plus motion.

3.6 Discussion and Future Work

This chapter has compared four hand detection methods [11, 12, 2, 13] on three sign language datasets. Comparing these four popular hand detection methods

Table 3.4: Comparisons between combination20 and alternatives

rank of correct sign	percentage of queries		
	combination+combination20	skin+motion	residue
1	**27.9**	27.0	17.9
4	**54.2**	53.8	37.9
10	**72.5**	72.1	55.4
20	**89.6**	88.8	75.0

Accuracy statistics for using the combination method with the experimental setting of combination20, skin and motion, and residue. The first column specifies values of rank of correct sign. For each such value, x, for each of the three hand detectors, we show the percentage of test signs for which the correct sign class is among the top x matches.

Table 3.5: Comparisons between combination50 and alternatives

rank of correct sign	percentage of queries		
	combination+combination50	skin+motion	residue
1	33.3	**37.3**	25.3
4	64.0	**65.3**	42.0
10	**86.0**	84.7	69.3
20	**97.3**	96.7	83.3

Accuracy statistics for using the combination method with the experimental setting of combination50, skin and motion, and residue. The first column specifies values of rank of correct sign. For each such value, x, for each of the three hand detectors, we show the percentage of test signs for which the correct sign class is among the top x matches.

Table 3.6: p-value table of DSTW

rank of correct sign	p-value	
	comb20 vs sm + dstw	comb50 vs sm + dstw
1	7.5e-1	2.4e-1
4	8.8e-1	7.4e-1
10	8.6e-1	6.6e-1
20	6.6e-1	6.6e-1

Table 3.7: Comparisons between the combination method with user-independent setting and alternatives

rank of correct sign	percentage of queries		
	combination+user-independent	skin+motion	residue
1	**15.0**	14.3	12.7
4	**33.3**	32.0	25.0
10	**52.0**	48.7	35.7
20	**68.3**	67.7	48.3

Accuracy statistics for using the combination method with the user-independent setting, skin and motion, and residue. The first column specifies values of rank of correct sign. For each such value, x, for each of the three hand detectors, we show the percentage of test signs for which the correct sign class is among the top x matches.

show that the skin and motion based method provides the best results on our sign language datasets. It is also clear, however, that its performance on two-handed signs drops considerably. We can also tell that the multi-proposals method gives the worst results from the comparisons. The possible reason is that the models used for the multi-proposals method are trained on an external dataset. The proposed hand detector, which combines these four methods, achieved significantly better detection accuracy than all of its components under user-dependent experimental protocol. In future work, we will use our sign language datasets to train the models for the multi-proposals method. At the same time, we will utilize tracking algorithms to ensure hand candidate temporal consistency across frames, rather than relying on single frame detection.

CHAPTER 4

FALL DETECTION USING A SINGLE, UNCALIBRATED RGB-D CAMERA

4.1 Introduction

Over the past few years, several researchers have focused on designing robust algorithms to understand and recognize human actions in videos. This is a challenging research area, and progress in that area can be directly applicable in real-world applications. As an example, smart monitoring systems that recognize human actions and events can be helpful for improving the safety of elderly people and patients living unaccompanied at home. A useful functionality of such a home monitoring system is fall detection, i.e., the ability to automatically detect cases where a human falls and may have been injured.

In this paper, we propose a statistical method to detect falls based on depth images captured using a Kinect [58] camera. Using the depth images, we detect and track the person in both image coordinates and world coordinates. From every frame, features are extracted based on the position and velocity of the person. A Bayesian classifier is built on top of those features. At each frame, the system uses this Bayesian classifier to decide whether a fall event has just concluded.

Modeling fall events using a camera-independent world coordinate system allows our method to be viewpoint invariant, and to be quite insensitive to the choice of position and viewing direction for the camera. Using observations from a single camera, and requiring minimal effort to adjust the system once the camera has been moved, are key differentiating features from most existing computer vision methods for fall detection, which are typically either viewpoint-dependent, or require multiple cameras and a careful and time-consuming calibration process. Especially for calibrated multi-camera systems, we should have in mind that the calibration process needs to be repeated every time a single camera is intentionally or accidentally removed.

A key feature of our experiments is that we record all training data from a specific camera position, and all test data from another specific camera position, different from the position used to record the training data. Our results show that moving the camera to a new position does not affect accuracy. Several methods have been proposed in the literature that use 3D features, and thus in theory such methods are viewpoint invariant. However, we are not aware of other methods that have been explicitly evaluated using such a protocol, where the viewpoint (or overall multi-camera setup, for multi-camera systems) for the test data is ensured to be different than viewpoint used for the training data. While 3D features are theoretically viewpoint-invariant, training and testing using the same viewpoint does not allow identification of systematic errors that may occur when the viewpoint or camera arrangement changes. Such systematic errors may be caused by discrepancies in calibration, ground-floor estimation, or any other parameters that need to be re-estimated when the viewpoint or multi-camera arrangement changes. Such errors will not be detected when the same camera placements are used for both training data and test data, and when calibration parameters do not need to be recomputed for the test data.

Our experiments include simulated falls by several subjects. They are performed in a user-independent setting, where, for each test video, the subject appearing in the test video does not appear in any of the training data used to detect falls in that video. Another key feature in our experiments is that our data include several confounding events, such as lying down on the floor, picking up an object from under the bed, tying shoelaces, etc. Overall, the experiments demonstrate robust performance, with good accuracy on detecting falls and minimal false alarms.

4.2 Related Work

Several approaches have been proposed for fall detection. We refer the reader to [59, 60, 61, 62] for some recent reviews. With respect to the method proposed in this paper, existing methods can be categorized as using (exclusively or partially) sensors, or being exclusively vision-based. Furthermore, vision-based methods can be characterized based on 3D-based methods using multiple calibrated cameras, appearance-based methods using one or more (but uncalibrated) color cameras, and 3D methods using depth cameras.

Several existing methods use a variety of non-vision sensors to make a decision [63, 64, 65, 66, 67]. The most commonly used sensor is the accelerometer. Chen et al. [68] use low-cost and low-power MEMS accelerometers to detect falls. Zhang et al. [67] embed a tri-axial accelerometer in a cellphone, whereas [65] set accelerometers behind the person's ear.

In [63, 64], the authors propose fall detection algorithms based on thresholds on both signals from a tri-axial accelerometer and a biaxial gyroscope. In [69], three different sensors are used: a 3D Time-Of-Flight range camera, a wearable MEMS accelerometer and a microphone. In Luo et al. [70], the input comes from waist-mounted accelerometry. Lai et al. [71] use several triaxial acceleration sensor devices for joint sensing of injured body parts, when an accidental fall occurs. Li et al. [72] use both accelerometers and gyroscopes to detect falls. Kangas et al. [73] show that triaxial accelerometric measurements at waist and head have potential to distinguish between falls and normal daily activities. Hwang et al. [74] use an accelerometer to measures kinetic force and tilt sensor and a gyroscope to estimate body posture.

They design rules based on signals from these sensors to determine whether a fall has occurred. Lee et al. [75] implement a wireless accelerometer sensor module and an algorithm to determine the wearer's posture, activity, and occurrence of fall events.

The methods described above, that use wearable sensors, have all made valuable contributions by demonstrating the accuracy and reliability of such technology for fall detection. At the same time, requiring subjects to wear sensors can also have potential disadvantages. People may not be willing to wear such sensors, and even if the subjects are willing, it may be difficult to ensure that they wear these sensors throughout the day. Wearing these sensors at night can also be an issue, as falls can occur in the dark during a night visit to the restroom or to the kitchen for a drink of water. The key differentiating aspect of our method compared to wearable sensor methods is that our input device is a camera. Once the camera is placed, the subjects can be oblivious to the camera's functioning, and thus they do not need to actively cooperate with the system by accepting to wear/carry sensors or altering in any other way their daily routine.

An alternative to both wearable sensors and cameras is proposed by Rimminen et al. [76], where falls are detected using a floor sensor based on near-field imaging. The floor sensor detects the locations and patterns of people by measuring impedances with a matrix of thin electrodes under the floor.

Similar to our method, several other vision-based methods have been proposed for fall detection. A first categorization of vision-based methods is based on the type and number of cameras that are used, and the need or not for external camera calibration. We first consider multi-camera calibrated systems.

An example of a method using multiple cameras is the method of Cucchiara et al. [77], that is used for detecting and tracking people and recognizing fall events. Multiple cameras are used to cover different rooms and the camera handoff is treated by warping the person's appearance in the new view by means of homography. Auvinet et al. [78, 79] use a network of multiple calibrated cameras. Fall events are detected by analyzing the volume distribution along the vertical axis, and an alarm is triggered when the major part of this distribution is abnormaly near the floor. Anderson et al. [80] use multiple cameras and a hierarcy of fuzzy logic to detect falls. Overall, using multiple cameras offers the advantage of allowing 3D reconstruction and extraction of 3D features for fall detection. [81] propose using the measures of humans' heights and occupied areas to distinguish three typical states of humans: standing, sitting and lying. Two relatively orthogonal views are utilized, in turn, simplifying the estimation of occupied areas as the product of widths of the same person, observed in two cameras. At the same time, setting up those cameras requires careful and time-consuming external camera calibration. When a single camera is moved, whether by accident or intentionally, the system needs to be recalibrated. The method proposed in this paper detects falls using observations from a single camera, and robust performance is demonstrated in cases where the camera is moved.

Another category of vision-based methods for fall detection is appearance-based methods, which can be applied even with a single camera, but can also benefit from multiple cameras so that they cover larger areas. Nait-Charif et al. [82] use ceiling-mounted, wide-angle cameras with vertically oriented optical axes. Miaou et al. [83] uses an omni-camera and identifies falls based on the width to height aspect ratio of the person. Tao et al. [84] use a computer vision method to detect and track moving people and an event-inference module to parse observation sequences of people features for possible falling behavior signs. Similar to [83], the width to height aspect ratio of the person is the feature used for identifying falls. Nater et al. [85] present an approach for unusual event detection based on a tree of trackers.

Each tracker is specialized for a specific type of activity. Falls are detected when none of the specialized trackers for "normal" activities can explain the observation. [86] use variations in silhouette area that are obtained from only one camera and SVM classifier to detect fall. [87] use adaptive background Gaussian mixture model (GMM) to segment object. An ellipse shape has been built from the segmented object for body modeling. And then five features are extracted from this ellipse model and fed into two Hidden Markov Models(HMM) to classify fall and normal activities.

Rougier et al. [88] use a shape matching technique to track the person's silhouette along the video sequence. The shape deformation is then quantified, and falls are detected from normal activities using a Gaussian mixture model. Zweng et al. [89] detect falls using multiple cameras. Each of the camera inputs results in a separate fall confidence (so no external camera calibration is needed). These confidences are then combined into an overall decision. Fu et al. [90] use an asynchronous temporal contrast vision sensor which features sub-millisecond temporal resolution. A lightweight algorithm computes an instantaneous motion vector and reports fall events.

One attractive feature of appearance-based methods is that they do not require camera calibration. The disadvantage is that they are viewpoint-dependent, and moving a camera to a different viewpoint (especially a different height from the floor) would require collecting new training data for that specific viewpoint. In contrast, our method does not require new training data for each viewpoint.

Lustrek et al. [91] use an infrared-based motion capture system. The fall detection system considers locations of body parts and the angles between adjacent body parts. They compare eight machine learning algorithms, and the highest classification accuracy of over 95% is achieved by Support Vector Machine used on the reference attributes and angles. Use of a motion capture system provides highly informative information about articulated 3D body pose, but such a solution can be prohibitively expensive for adoption by consumers.

Depth cameras provide 3D information about the scene, thus allowing extraction of viewpoint-invariant features without the need to externally calibrate multiple cameras. Diraco et al. [3] use a wall-mounted Time-Of-Flight 3D camera to monitor the scene. The system identifies a fall event when the human centroid gets closer than a certain threshold to the floor, and the person does not move for a certain number of seconds after getting close to the floor. In a related approach, Leone et al. [4] employ a 3D range camera. A fall event is detected based on two rules: (1) the distance of the person's center-of-mass from the floor plane decreases below a threshold within a time window of about $900ms$; (2) the person's motion remains negligible within a time window of about $4s$. Rougier et al. [5] use a Kinect camera to obtain depth images. Thresholds on the human centroid height relative to the ground and the body velocity are used to determine if a fall has occurred.

Those three fall detection methods using a single depth camera [3, 4, 5] are the most related to our method, which also detects falls using observations from a single depth camera. There are three main differences between our method and those three methods. The first difference is that in all three methods [3, 4, 5] the decision is made by applying thresholds to individual features. In our method, decisions are made probabilistically, which allows all information to be considered before making a hard decision. Furthermore, the probability threshold for making a decision is a tunable parameter in our method, allowing different trade-offs between sensitivity and false alarms. Another difference is that our method proposes additional features in addition to distance from the floor and acceleration. We also use the duration of the candidate fall event, the total change in height between the beginning and the end of the candidate event, and the percentage of frames in the candidate event where the person's height decreases. Another contribution of our method compared to the three related methods of [3, 4, 5] is, as mentioned earlier, our experimental protocol, whereby all training data are collected from a specific viewpoint, and all the test data are collected from another viewpoint, several meters away from the training viewpoint. We believe that this new evaluation protocol is a useful approach for measuring the robustness of the system to displacements of the camera.

In our own prior work, we proposed a method for identifying falls using a single color camera [92], and using a single RGB-D (Kinect) camera [6, 93]. This paper is an extended journal version of [6]. At the same time, this paper contains significant improvements compared to [6]. First, the method in [6] was not fully automatic, as it required the user to click on two points on the depth image to specify the vertical orientation in world coordinates. That had to be done each time the Kinect camera moved. Our current method estimates the vertical orientation automatically. Second, the evaluation dataset is significantly expanded, including over 220,010 frames, and 300 examples of falls, compared to 10,400 frames and 12 examples of falls in the experiments of [6]. Instead of just concluding frontal and lateral fall directions, the expanded dataset has more varied fall directions. There are eight fall directions in total in the expanded dataset, which is named as **EDF** in this paper. Third, in addition to creating the **EDF** dataset, we also make available three partially occluded fall datasets. A partially occluded fall refers to a fall where part of the body is occluded by a certain object when the person performs the fall action. We use this dataset to verify that the proposed method is robust to partial occlusions. Four, in this paper we evaluate our approach against several existing methods [3, 4, 5], whereas no such comparisons were included in [6].

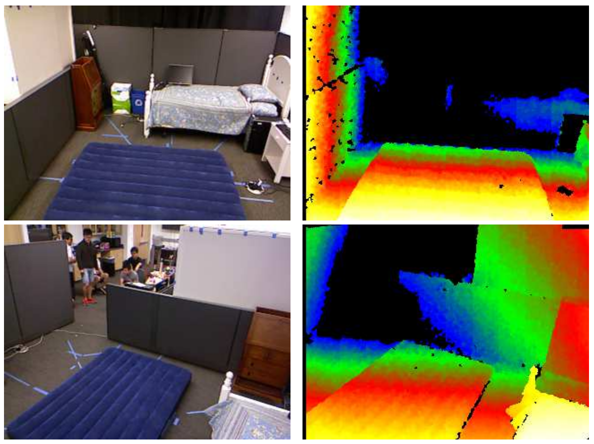


Figure 4.1: Our simulated apartment, seen from the two viewpoints that is used to make **EDF** dataset. For each viewpoint we show a color image and a depth image. Depth images are color-coded so that: white indicates small depth values, and yellow, orange, red, green, blue indicate progressively larger depth values. Black indicates invalid depth.

4.3 Scene Setup

The experimental data for this paper come from experiments run in the Heracleia Human Centered Computing Laboratory at the University of Texas at Arlington. In that lab, a simulated apartment has been set up. Two Kinect cameras were set up at two corners of the apartment, and were set to monitor the apartment. The reason of setting up two Kinects is that the range of the Kinect's depth sensor is from $0.5m$ to $4m$, which means that one Kinect is not enough to cover the whole apartment. In addition, this setup allows us to validate our claim that the proposed method is to a large extent viewpoint-invariant, and automatically adapts when moving the camera from one place to another. Figure 4.1 and Figure 4.2 show the scene setups for recording the eight directions dataset (**EDF**) and the partial occlusion datasets respectively.

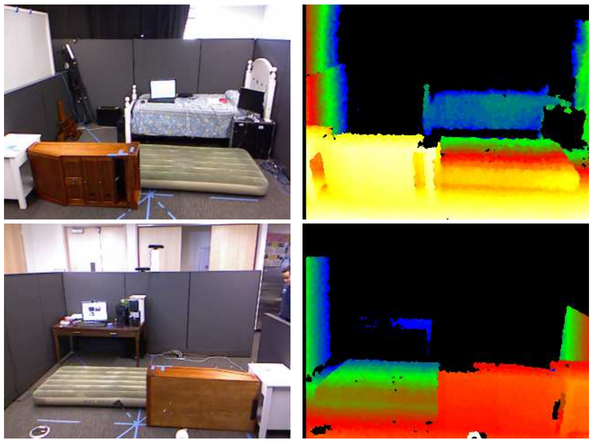


Figure 4.2: Our simulated apartment, seen from the two viewpoints that is used to collect partially occluded videos. For each viewpoint we show a color image and a depth image. Depth images are color-coded so that: white indicates small depth values, and yellow, orange, red, green, blue indicate progressively larger depth values. Black indicates invalid depth.

4.4 Image and World Coordinates

In this section we describe how to compute the transformation from image coordinates to world coordinates.

4.4.1 Image and Kinect Coordinates

Let us consider a pixel i on a depth image obtained by the Kinect camera. The pixel's image coordinates are denoted as $[x_i, y_i]$, and the depth value at that location is z_i. We combine the image coordinates and depth value into vector $\mathbf{x_i} = [x_i, y_i, z_i]$.

We define a Kinect-centered world coordinate space, whose origin point is the center of the Kinect sensor array. The xy-plane of that space is parallel to the depth image plane and the z axis points to the camera view direction. Figure 4.3 provides an illustration of this coordinate system.

Let $[w, h]$ be respectively the width and height of the image in pixels, and m_x and m_y be camera calibration values describing scaling factors along the x and y

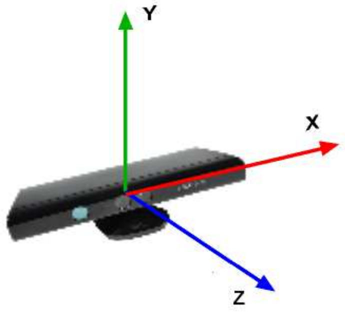


Figure 4.3: Kinect-centered world coordinate space. The origin is the center of the Kinect sensor array. The xy-plane of that space is parallel to the depth image plane and the z axis points to the camera view direction.

axis respectively (m_x and m_y are specified in the Kinect documentation provided by Microsoft). Then, the Kinect-centered world coordinates $[x_k, y_k, z_k]$ corresponding to image coordinates $[x_i, y_i]$ and depth value z_i can be computed by Equation 4.1 as follows:

$$\begin{aligned} x_k &= (\frac{x_i - 1}{w - 1}) * m_x * z_i \\ y_k &= (\frac{y_i - 1}{h - 1}) * m_y * z_i \\ z_k &= z_i \end{aligned} \tag{4.1}$$

4.4.2 World Coordinates

The Kinect coordinates refer to the 3D world, but the y axis in those coordinates is not necessarily the world vertical axis (which points up and down in the real world). In our setting, it is important to compute the real-world y value corresponding to each pixel in the depth image. Thus, we define a world coordinate system, where:

- The y axis points up and down.
- The zero value on the y axis corresponds to the floor.

- Positive values on the y axis corresponds to locations above the floor.

In order to calculate the real-world y value for depth image pixels, we first need to estimate the floor plane of the scene in Kinect-centered world coordinates. Then, the real-world y value of a certain pixel can be computed as a simple point-to-plane distance.

4.4.2.1 Calculating the floor plane in Kinect-centered world coordinates

To compute the floor plane, we must estimate parameters A, B, C in the following equation:

$$Ax + By + Cz + D = 0 \tag{4.2}$$

For the correct values of A, B, C, D, any floor point $[x, y, z]$ (expressed in Kinect coordinates) satisfies the above equation. We obtain parameters A, B, C, D directly using the Microsoft Kinect SDK, which has built-in floor estimation functionality. Furthermore, the Kinect SDK provides normalized values for parameters A, B, C, D so that the physical interpretation of D is the height of the camera from the floor. Figure 4.4 shows four examples of floor detection. The pixel is classified as floor if its height value is less than 2cm in our method.

4.4.2.2 Calculating the world y-axis of a certain pixel

Given a pixel's Kinect coordinates, $[x_k, y_k, z_k]$, the distance from this point to the floor ground can be directly obtained by a simple point-to-plane distance calculation:

$$y_w = \frac{|Ax_k + By_k + Cz_k + D|}{\sqrt{A^2 + B^2 + C^2}} \tag{4.3}$$

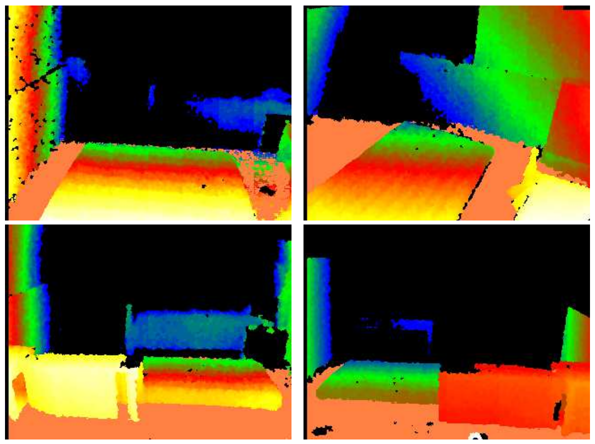


Figure 4.4: The floor plane, which is indicated by orange color, is detected by Microsoft Kinect SDK in four scenes.

4.5 A Statistical Method to Detect Falls

In this section, we describe our statistical method for fall detection. First, we describe how to detect the person in every frame, and then how to extract features from a sequence of depth video frames. Once we obtain features from training examples of falls and non-falls, we can train positive (fall) models and negative (non-fall) models from these features. Finally, we describe how to make a decision to determine whether the current frame corresponds to the end of a fall event or not.

4.5.1 Person Detection

We assume that the only moving object in the scene is the person. To detect the person, we use background subtraction method. The background can be identified by the following two methods.

4.5.1.1 Static Background Model

The static background map B is obtained by averaging the last few frames where no significant motion was observed.

4.5.1.2 Adaptive Background Model

The algorithm proposed by Stauffer and Grimson [51] is the representative adaptive background model method which uses a mixture of Gaussian (MoG) distributions to model a multimodal background image sequence. For each pixel, each normal distribution in its background mixture corresponds to the probability of observing a particular intensity or color in the pixel.

We choose the static background model in this paper because: (1) it is simpler than the adaptive background model; (2) it gets better detection result on the depth videos which is shown in the experiment section.

Let $D_n(x, y)$ denote the value at pixel coordinates (x, y) for the n-th depth frame. Through comparing $D_n(x, y)$ with the background depth map $B(x, y)$, the moving parts of the current frame can be extracted easily. In particular, using a system-specific threshold for T ($T = 200$ in our implementation), a pixel is considered to belong to a moving object if its depth value differs from the corresponding background value by more than T:

$$M_n(x, y) = \begin{cases} 1 & if \quad |D_n(x, y) - B(x, y)| > T \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

Once $M_n(x, y)$ has been computed for all pixels of frame n, connected component analysis is used to identify the largest connected component with values $M_n(x, y) = 1$. We call that component the *person region.*

An example result of this simple person detection method is shown on Figure 4.5. Overall, we have found that this method produces near-perfect results.

We should note that the proposed method would not work well in cases where multiple people are visible. However, we consider that a fall detector is most needed when a person is alone in the apartment, as that is the case when an automated system

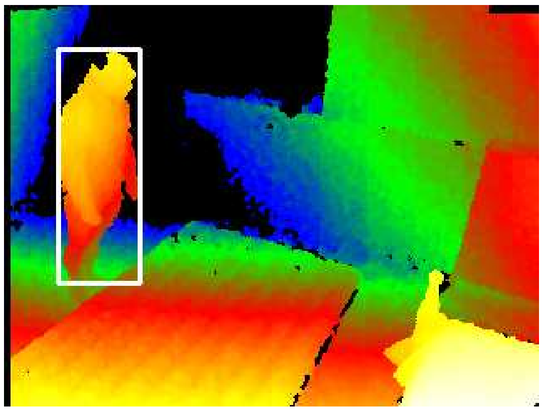


Figure 4.5: Person detection using depth map. The white rectangle indicates the bounding box of the person region.

is most needed to call for help. We also note that the background in an apartment may change every now and then, for example due to motion of chairs and desks. Our method can easily accommodate that, because it defines the background image as the average of the last few frames where no significant motion was observed. Thus, the background model is quickly updated when furniture configurations change.

4.5.2 Computing the Height of the Person

We have found that an important feature for fall detection is the distance from the top of the person to the floor. We use the term "top" instead of the term "head", because in various cases the head is not the part of the body that is the farthest from the floor, e.g., when a person is tying their shoelaces.

To estimate the distance from the top of the person to the floor, we first calculate the world y-axis value of every pixel in the person region. These world y-axis values are sorted in descending order. The 2.5 percentile world y-axis value is considered as the height of the person's top.

We note that, instead of using the top of the person as a feature, it is also possible to use the height of the centroid of the person region [3, 4, 5]. We have

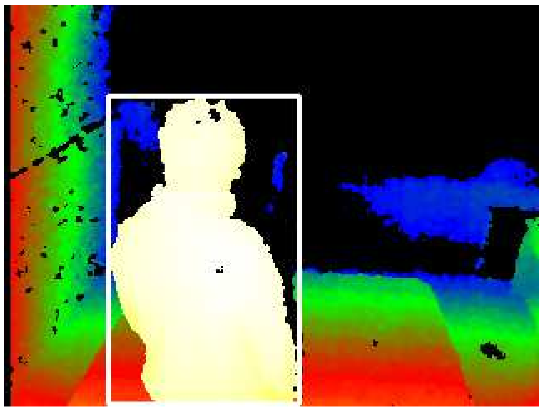


Figure 4.6: Occlusion example, the person's location is indicated by a white rectangle in the depth image. It can easily be seen that the centroid of the detected person region is not the actual centroid of the person.

found, however, that using the centroid may not be accurate in cases where the person is partially occluded, as shown on Figure 4.6.

4.5.3 Features Extracted from Fall Event

When there is a fall, the y-axis value of the top of the person decreases rapidly over a few frames. There are several possible features one can use to distinguish between fall events and non-fall events. In this section, we describe the features that we found to be the most useful.

Given a specific start frame i and end frame j , we treat the video subsequence between frame i and frame j as a candidate fall event. Let the sequence of person height in every frame be denoted as $[h_i, ..., h_j]$. In order to decide if this candidate is indeed a fall or not, we extract from the video subsequence the following five features:

Duration. The duration of a fall in frames, denoted as k, is defined simply as: $k = j - i + 1$.

Smallest person height. Denoted as l, this is defined simply as: $l = \min(h_i, ..., h_j)$.

Maximum speed. This is the largest drop in person height, from one frame to the next, denoted as m, and defined as: $m = \max_{n \in \{i+1,\ldots,j\}}(h_{n-1} - h_n)$.

Total person height decrease. This is the total change of person height during the fall, denoted as t, and defined as: $t = h_i - h_j$.

Fraction of frames where person height drops. Denoted as p, this is simply the percentage of frames, between $i + 1$ and j, where the height has a smaller value than in the previous frame.

In [92] , we did not use feature p, i.e., the percentage of frames where the y value of the height has decreased. Instead, we force the fall event to be a process where the person height must decrease frame-by-frame. We have found that condition to be too strict, leading occasionally to false negatives, where a true fall event fails to get detected, due to the fact that computing the person height may lead to a noisy result. This is why, in the method described here, we opted to remove this hard constraint, and replace it with this new feature p.

4.5.4 Training Positive Models and Negative Models

We perform training in a user-independent and camera independent manner. We split our recorded videos into two big sets: view 1, which has been recorded from a specific viewpoint, and view 2, which has been recorded from another specific viewpoint, different to that of view 1. In each view, the data is split into sixteen sets, where each set includes all videos from a specific subject. This way, for each view, each human subject appears in only one of the sets. Then, each set in one view is used, in turn, as a test set. To detect falls on that test set we apply the statistical model constructed using as training data the remaining fifteen sets (i.e., all sets that do not include the human subject appearing in the test set) from the other view. Fall events(start and end frames are manually annotated in all videos, and this information is used during training and also to measure the accuracy on the test data.

Our system makes a decision at each frame of each test video. The decision is on whether that frame is the last frame of a fall event. If it is determined that a fall event has occurred, the start frame for the fall event is also determined. At a later stage, fall detection events that overlap in time are merged as a single event.

At each frame n, we go back from ten to fifty frames. This choice was made based on the time it takes for a person to fall. We note that our videos are recorded at about 25fps, and thus ten frames correspond to about 0.4 seconds and fifty frames correspond to about 2 seconds. We did not see any need to consider shorter or longer intervals. In this way, we get 41 candidate fall events at each frame n, each such candidate event starting at frame $n - i$ (i ranging from ten to fifty) and ending at frame n.

For every candidate fall event, five features are extracted, as described in Section 4.5.3. We assume that the five features are conditionally independent, given the class of the event, where the class can be positive (fall) or negative (non-fall). For the positive examples, we model each feature distribution as a Gaussian, as we have few positive samples. For the negative examples, we model each feature distribution nonparametrically as a histogram, since we have tens of thousands of negative examples, a number that is quite adequate for estimating a nonparametric one-dimensional distribution. While assuming conditional independence is less realistic, it allows us to build statistical models using much fewer training examples than we would need to account for conditional dependencies.

4.5.5 How to Make a Decision

As discussed in Section 4.5.3, at each frame n we consider 41 frame intervals of length 10 to 50 frames, ending at frame n, and we decide separately for each interval if it constitutes a fall event or not. Here we describe how to make that decision. For the interval in question, we compute a feature vector $[k, l, m, t, p]$. Using Bayes rule, the probability of a fall having occurred at this interval can be computed as:

$$P(f \mid k,l,m,t,p) = \frac{P(k,l,m,t,p \mid f)P(f)}{P(k,l,m,t,p)} \tag{4.5}$$

In Equation 4.5, all quantities can be computed based on the training data and the probability distributions that we have constructed. $P(f)$ is the prior probability of the candidate frame interval being a fall, which is easily computed from the frequency of falls in the training data. To reduce the complexity of the probability distributions, we assume that each feature is conditionally independent of all other features given the true label (fall or non-fall) of the event. Then, $P(k,l,m,t,p) = P(k \mid f)P(l \mid f)P(m \mid f)P(t \mid f)P(p \mid f)P(f) + P(k \mid \neg f)P(l \mid \neg f)P(m \mid \neg f)P(t \mid \neg f)P(p \mid \neg f)P(\neg f)$. All these quantities can be determined using $P(f)$ and the conditional probability distributions we have computed for each feature for the positive and negative samples. Therefore, Equation 4.5 can be rewritten as:

$$\begin{aligned} P(f \mid k,l,m,t,p) = {} & \frac{P(k \mid f)P(l \mid f)P(m \mid f)}{P(k,l,m,t,p)} * \\ & \frac{P(t \mid f)P(p \mid f)P(f)}{P(k,l,m,t,p)} \end{aligned} \tag{4.6}$$

To determine whether the interval constitutes a fall event, we simply compare:

$$P(f \mid k,l,m,t,p) \geq \theta \tag{4.7}$$

Setting different values of θ we get different rates of fall detections and false positives, as shown in the experiments. This way, by adjusting θ the system can be easily tuned to trade off between detecting more fall events and reducing the number of false alarms.

If the system determines that a fall has occurred for two or more intervals that are temporally overlapping, then only one detection is reported, that corresponds to the interval with the highest probability of fall. The other detections are suppressed. More specifically, if an interval $(n-i, \ldots, n)$ is classified as a fall event, it is likely that highly overlapping frame intervals will have similar features and will also be classified as fall events. To account for that, we cluster the neighboring candidate falls and then reject all the candidates in a cluster aside from the one with the highest score.

4.6 Experiments

4.6.1 Dataset Description

We use Kinect for Xbox 360 and the Microsoft Kinect for Windows SDK Beta to build the recording system and the programming language is `C#`. All the experiment datasets are collected by this system at a frame rate of about 30fps.

The first created dataset, named **EDF**, includes eight fall directions. Figure 4.7 illustrates the setup of the two cameras (viewpoints) for recording this dataset. The blue arrows indicate the fall directions (8 directions in total). The two viewpoints were recorded at the same time, and thus every event was recorded simultaneously from both viewpoints. Each of the 10 subjects performed two falls along each direction in each viewpoint in the **EDF** dataset. So, there are 160 falls in each viewpoint and 320 falls in total. Figure 4.8 and 4.9 show one fall example in each direction in the viewpoint 1 and viewpoint 2 respectively.

To verify that the proposed method is robust to occlusions, we have collected three partial occlusion datasets: **O25**, **O50** and **O75**. In the **O25** dataset, 25% percent of the person's body is occluded by a certain object when the person falls down. In the **O50** dataset, 50% percent of the person's body is occluded by a certain object when the person falls down. In the **O75** dataset, 75% percent of the person's body is occluded by a certain object when the person falls down. Each of the 5 subjects performed five partially occluded falls in each viewpoint in the **O25**, **O50** and **O75** datasets respectively. There are also two viewpoints in these partial occlusion datasets. Each viewpoint was recorded at separate times from the other viewpoint, and thus we had no instances where the same events were recorded simultaneously from both viewpoints. Figure 4.10 show one fall example in each viewpoint in the **O25**, **O50** and **O75** datasets respectively.

Table 4.1 gives the summary of each dataset. The **EDF** dataset includes 110,774 frames and 160 real falls in videos from the first viewpoint, and 109,236 fra mes and 160 real falls in videos from the second viewpoint. The **O25** dataset includes 17,402 frames and 25 real falls in videos from the first viewpoint, and 17,061 frames and 25 real falls in videos from the second viewpoint. The **O50** dataset includes 18,613 frames and 25 real falls in videos from the first viewpoint, and 17,264 frames and 25 real falls in videos from the second viewpoint. The **O75** dataset includes 16,699 frames and 25 real falls in videos from the first viewpoint, and 15,992 frames and 25 real falls in videos from the second viewpoint.

In the **EDF** dataset, our subjects also performed a total of 100 actions that tend to produce features similar to those of a fall event, namely: 20 examples of picking up something from the floor, 20 cases of sitting on the floor and 20 examples of lying down on the floor, 20 examples of tying shoelaces and 20 examples of doing plank. Each partial occlusion dataset includes a total of 40 actions that tend to produce

Table 4.1: Summary of each dataset

	# of subjects	# of frames		# of falls		# of non-fall actions	
		viewpoint 1	viewpoint 2	viewpoint 1	viewpoint 2	viewpoint 1	viewpoint 2
EDF	10	110,774	109,236	160	160	50	50
O25	5	17,402	17,061	25	25	20	20
O50	5	18,613	17,264	25	25	20	20
O75	5	16,699	15,992	25	25	20	20

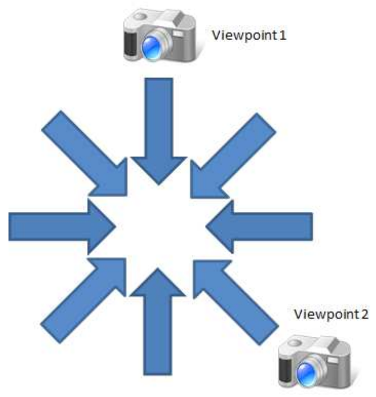


Figure 4.7: Kinect cameras setup for recording the **EDF** dataset

features similar to those of a fall event, namely: 10 examples of picking up something from the floor, 10 examples of sitting on the floor, 10 examples of tying shoelaces, and 10 examples of lying down on the bed.

Our entire dataset and annotations are publicly available at: `https://sites.google.com/site/kinectfalldetection/`.

4.6.2 Experimental Protocol

Our experiments are both user-independent (to recognize the actions of the user, no training data from the same user is used) and viewpoint-independent (to recognize actions observed from a specific viewpoint, no training data from the same viewpoint is used). We divide our videos into two big groups (one per viewpoint), and each big group includes one small group per subject. Each small group is used as a test set, for which we apply models learned using only videos from the other viewpoint and other subjects (different than the subject in the test set).

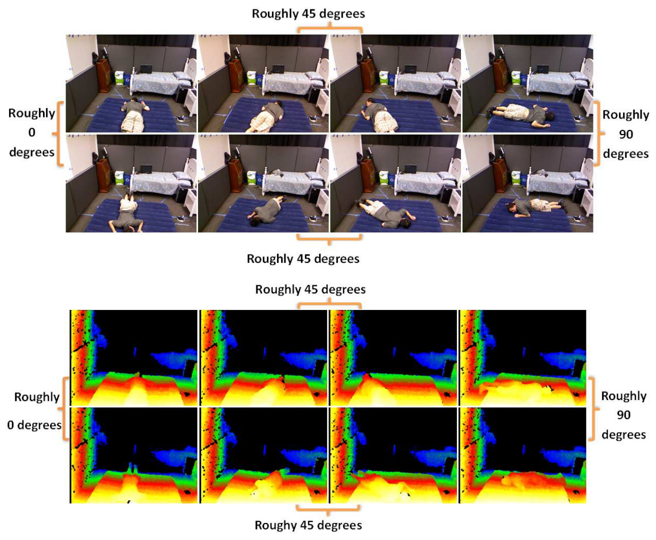


Figure 4.8: A person falls down along eight different directions in viewpoint one. The top is for color images while the bottom is for depth images.

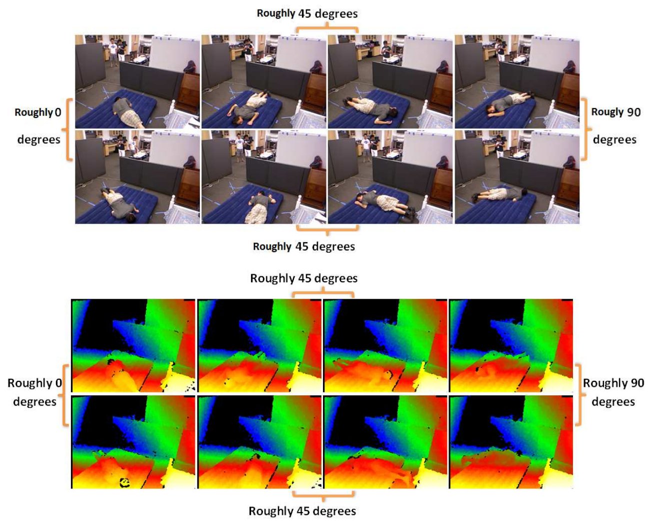
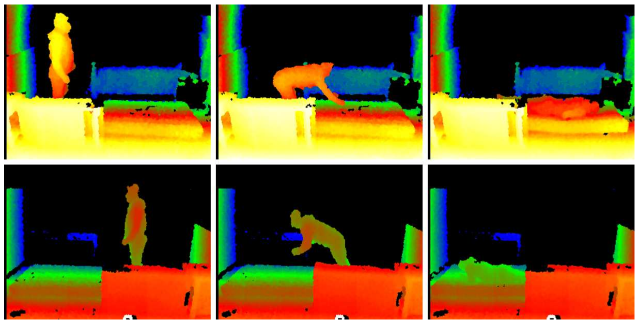
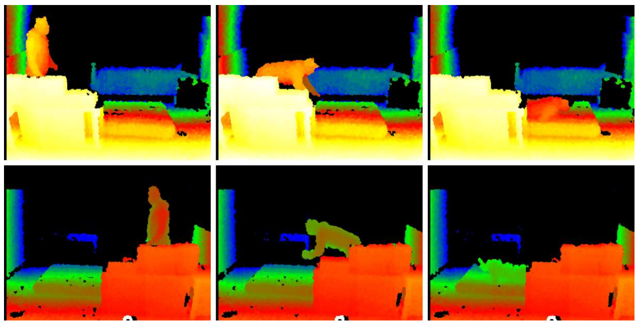


Figure 4.9: A person falls down along eight different directions in viewpoint two. The top is for color images while the bottom is for depth images.


(a) Two 25% occluded fall examples. The top images are from viewpoint 1 while the bottom images are from viewpoint 2. The left is the first frame, the middle is the middle frame and the right is the last frame.


(b) Two 50% occluded fall examples. The top images are from viewpoint 1 while the bottom images are from viewpoint 2. The left is the first frame, the middle is the middle frame and the right is the last frame.

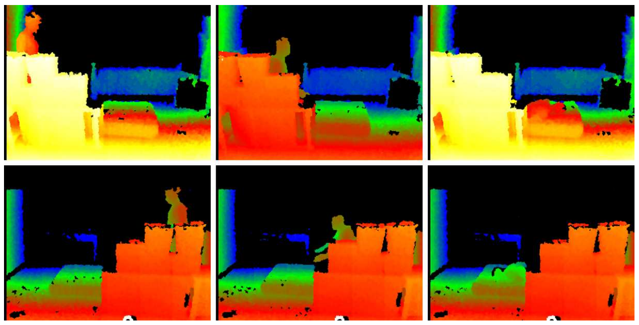

(c) Two 75% occluded fall examples. The top images are from viewpoint 1 while the bottom images are from viewpoint 2. The left is the first frame, the middle is the middle frame and the right is the last frame.

Figure 4.10: Partial occlusion fall examples

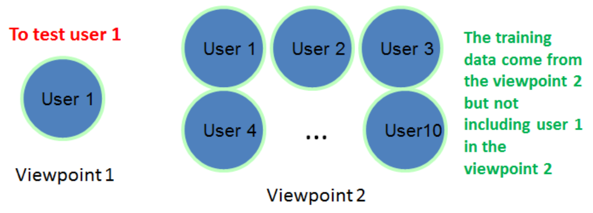

To test user 1
User 1
Viewpoint 1
User 1
User 2
User 3
User 4
...
User10
Viewpoint 2
The training data come from the viewpoint 2 but not including user 1 in the viewpoint 2

Figure 4.11: The training and test procedure for one subject in the **EDF** dataset.

Figure 4.11 illustrates the training and test procedure for one subject in the **EDF** dataset. To test subject 1 in the viewpoint 1, the training data come from all subjects but not subject 1 in the viewpoint 2. This training and test procedure is also applied to the occlusion datasets.

4.6.3 Comparisons Between Two Background Modeling Methods

The MoG based method (adaptive background model method) [51] has been implemented in the computer vision toolbox of Matlab. We use this Matlab implementation. The following are four parameters used in the implementation.

Number of initial video frames for training background model. This parameter is the number of training frames at the start of the video sequence and it is set to 150 in this experiment.

Learning rate for parameter updates. This parameter, which is set to 0.005, controls how quickly the model adapts to changing conditions.

Number of Gaussian modes in the mixture model. This parameter specifies the number of Gaussian modes in the mixture model and it is set to 3.

Threshold to determine background model. This parameter represents the minimum of the apriori probabilities for pixels to be considered background values and it is set to 0.7 in this experiment.

4.6.3.1 Measure of Accuracy

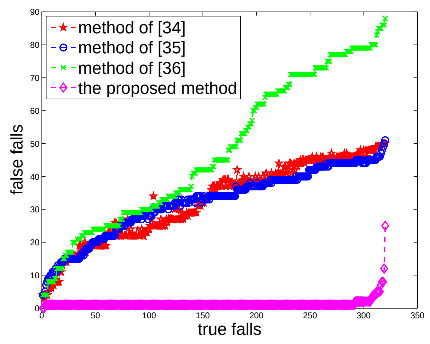
We randomly choose 281 frames from a depth image sequence and a color image sequence. An aligned rectangle is used to annotate the person in each frame. The detection is considered to be correct if the overlap ratio between the annotated rectangle and the detected rectangle is larger than a certain threshold. The adaptive background model method is tested on both color and depth image sequences while the static background model method is only tested on depth image sequence. Table 4.2 shows the result. In table 4.2, *method1* represents using adaptive background model on the color image sequence, *method2* represents using adaptive background model on the depth image sequence, and *method3* represents using static background model on the depth image sequence.

Table 4.2: Result of person detection

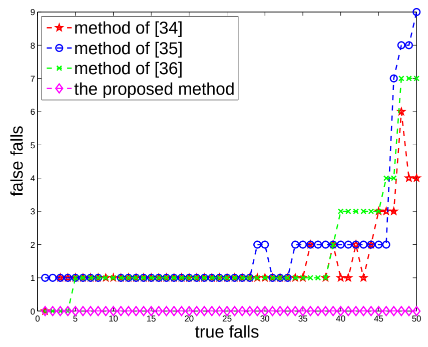
	method1	*method2*	*method3*
>0.3	46.26%	41.99%	100%
>0.4	38.08%	37.01%	93.24%
>0.5	30.60%	28.11%	81.14%

The adaptive background model method works worse than the static background model method for the following two reasons: (1) The background scene does not change in the video, which is the case for all of our test videos. The static background model method creates a background map from couple of starting frames of the video, where there is no moving object (the person). (2) The adaptive background model method suffers from the temporal background clutter problem while the static background model method does not. If a person stays in a place for a while, then this person will be considered as background in the following couple of frames, which leads to poorly background subtraction results for the following several frames. In contrast, the static background model method does not suffer from this issue.

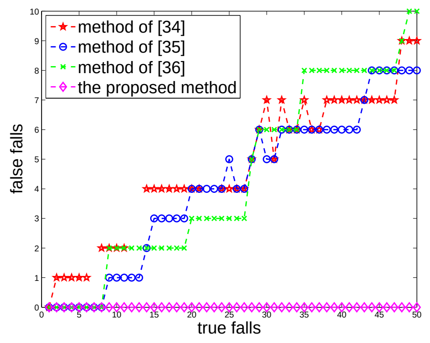
4.6.4 Comparisons with Existing Fall Detection Methods



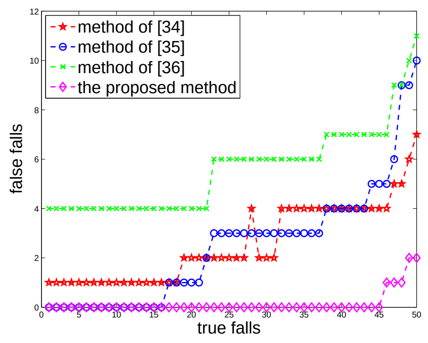
(a) comparison result on the **EDF** dataset



(b) comparison result on the **O25** dataset



(c) comparison result on the **O50** dataset



(d) comparison result on the **O75** dataset

Figure 4.12: Results of comparing the proposed method with the three most directly related existing methods [3, 4, 5] on the **EDF** dataset, **O25** dataset, **O50** dataset and **O75** dataset.

Here we compare the proposed method with three recent fall detection methods that are the most directly related to our method, as they also use a single RGB-D camera [3, 4, 5].

- In method [5], the human height relative to the ground, indicated as T_1, is used to detect (non-occluded or partially occluded) falls. When the end of an action is totally occluded by furniture, an analysis of the 3D body velocity prior to occlusion allows detection of the fall. All the datasets used in this paper do not include completely occluded cases. So the 3D body velocity feature does not help in this paper.
- The method of [3] uses two rules to describe a fall: the distance of the top of the person to the floor plane, which is indicated as T_2, being lower than a prefixed value, and an unchangeable situation (negligible movements of the person), which is indicated as U_2, lasting for at least 4 seconds.
- The method of [4] is also based on two rules: the distance of the top of the person to the floor plane, which is indicated as T_3, decreasing 0.40m within a time window of about 900ms (We refer the length of the time window as W_3), and the person's silhouette movement remaining negligible for a time window of about 4s (the length of the time window for negligible movements is referred as U_3).

We have implemented all these three methods. Since the thresholds provided by the authors may not be the best choice for our data, we searched and identified the best thresholds for each method. Figure 4.12 shows the results. We see that the proposed method is significantly more accurate than those three related methods on all datasets.

Take the **EDF** dataset (Figure 4.12a) an example, to detect 306 of the actual falls (about 96 percent of all falls in our dataset), our method produces 2 false positives, compared to 45 false positives by the method of [4], which is the best of the competitors here. The corresponding thresholds for the proposed method and the method of [4] are $\theta = -11.3265$ (the logarithm of the probability) and

$T_3 = 490mm, W_3 = 160(frames), U_3 = 140(frames)$ respectively. A false positive, which is reported by the method of [4] while rejected by the proposed method, in the **EDF** dataset is shown on Figure 4.13. Figure 4.13 shows a person doing plank on the floor. The height of the person in this false positive keeps low for a long time so that the competitive method, which relies on the height of the person and how long the height is lower than a threshold, can not make the correct decision. However, in addition to using the feature of the lowest person's height, the proposed method also considers other features, like the maximum speed, which help rejecting these two false positives.

Based on Figures 4.12b, 4.12c and 4.12d, we can see that to detect all the true falls, the proposed method reports zero false positives on the **O25** and **O50** datasets and 2 false positives on the **O75** dataset. These results indicate that the proposed method is robust to partial occlusions and also achieves better accurate than other competitors on these partial occlusion datasets. A false positive, which is reported by the method of [3] while rejected by the proposed method, in the **O75** dataset is shown on Figure 4.14. Figure 4.14 shows a person lying down on the floor. The reason of the proposed method beating the competitive method on the case of lying down on the floor is the same as doing plank introduced in the previous paragraph.

We should note that the proposed method would not work well in cases where the end of the fall is completely occluded by a certain object, like a bed. We refer these cases as complete occlusions. A possible solution to the complete occlusions is to set up multiple cameras so that there is no hidden area in the apartment.

4.6.5 Evaluating Individual Features

We analyze the contribution of each descriptor for detecting falls on the **EDF** dataset in this section. We also study the contribution of each descriptor versus

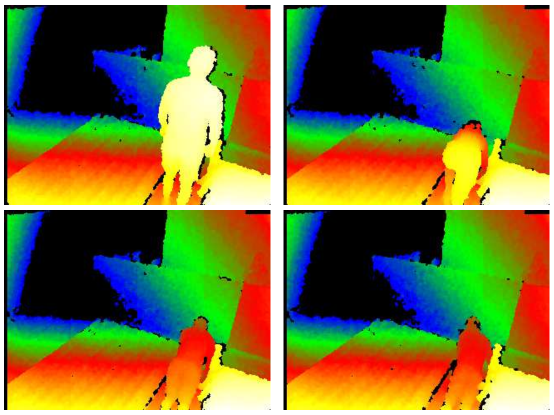


Figure 4.13: Example frames from a false alarm, showing a person doing plank.

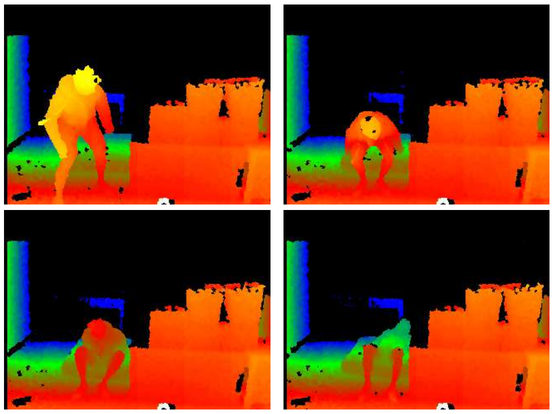


Figure 4.14: Example frames from a false alarm, showing a person lying down on the floor.

the kind of a fall. All falls in the EDF dataset are further categorized into three subgroups: falls with direction of 0 degrees, direction of 45 degrees and direction of 90 degrees. Figure 4.8 and 4.9 illustrate how to generate the subgroups. We name these three subgroups as: **EDF0**, **EDF45** and **EDF90**. The **EDF0**, **EDF45** and **EDF90** datasets include 80, 160, and 80 falls in total respectively, and also inherit all other non-fall actions from the **EDF** dataset.

In this section, we performed two sets of experiments on the dataset **EDF**, **EDF0**, **EDF45** and **EDF90**. In the first set, we evaluated the accuracy we obtain by leaving each feature out and using the remaining four features. In the second set of experiments, we evaluated the accuracy we obtain by using each of the five features by itself.

The positive and negative models, which are used in this section, are trained on the original **EDF** dataset using a user-independent and camera independent protocol introduced at section 4.5.4 and 4.6.2.

The results on Figure 4.15, 4.17, 4.19, 4.21 illustrate that all five features contribute somewhat towards the accuracy of the overall system. Features k, m and p make a relatively small contribution, and leaving each of them out does not change significantly the overall accuracy. Feature l, indicating the smallest height of the person during the video subsequence, reduces significantly the number of false detections that we obtain when we set the detection threshold low enough to detect all actual fall events. Leaving out feature t leads to the largest drop in accuracy.

The results on Figure 4.16, 4.18, 4.20, and 4.22 illustrate that features k, l, m, p used in isolation do not give very good performance. Feature t gives the best performance, and used in isolation is still comparable to the methods of [3, 4, 5] which are shown on Figure 4.12a. At the same time, using all five features instead of using just t eliminates several false positives for a broad range of detection thresholds.

4.7 Discussion and Future Work

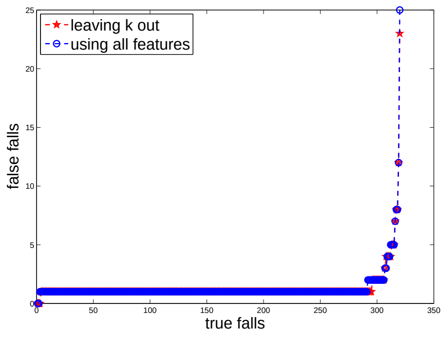
The proposed method produces good results in the experiments, and offers significantly better accuracy than the most directly related methods for this problem. At the same time, it is worth emphasizing that the proposed method is not meant to be a standalone system for fall detection, but rather to be a computer vision module that is part of a more general system. The overall fall detection system may contain additional modules, both to improve accuracy, and to include additional functionality, such as actually sending an alert about the detected fall. We believe that including sound processing and speech recognition would help significantly towards obtaining a robust system. Sound processing may produce additional features to be used for classifying a candidate fall event. Speech recognition can be used so that the system initiates a dialog with the subject, in the case where a fall has been detected. For example, the system can ask “Are you OK?” and the user can respond to indicate that there was no actual fall and no need to issue an alert. However, the focus of this paper has not been in proposing such an end-to-end system, but rather proposing a computer vision method that can be part of such an end-to-end system.

We should also note that a single RGB-D camera is clearly not sufficient to monitor the area of an entire apartment. For a real system, we anticipate that several cameras would be deployed, so as to cover most or all of the apartment area. In that scenario, we believe that the proposed method (that detects falls using observations from a single RGB-D camera) holds significant advantages over existing multi-camera systems, that detect falls using observations simultaneously recorded from multiple cameras, and require careful calibration each time a camera is moved. We note that, to cover an entire apartment, this calibration process would need to be even more complicated, or repeated several times, compared to using multiple cameras to observe a single part of the apartment. At the same time, the proposed method would also be more convenient to use compared to existing viewpoint-dependent appearance-based methods, because in our method training data does not need to be collected separately from each viewpoint.

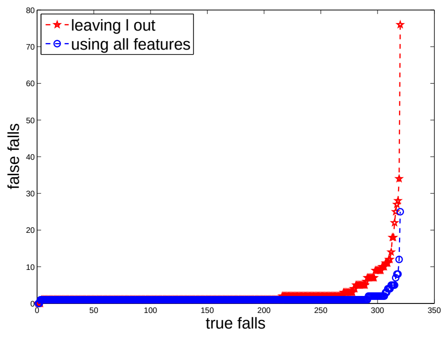
Regarding the use of computer vision in an end-to-end system, an interesting direction is to explore the use of a mobile camera-carrying robot. Such a robot can be dispatched when a fall is detected, to approach the person and observe from a close range to verify the status of the person. Such a robot could make additional visual observations about the position and pose of the person, that can be used to increase confidence that a fall has occurred. At the same time, successful deployment of such a robot would require advances not only in computer vision, but also in robotics, regarding accurate and safe navigation of robots in a domestic environment.

4.8 Summary

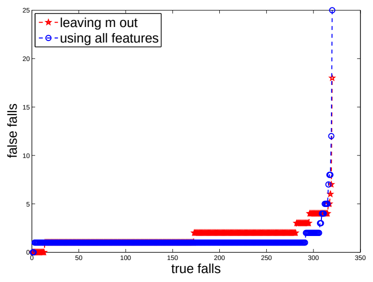
In this chapter, we have proposed a method that detects falls based on observations from a single RGB-D camera. In our experiments, the proposed method has produced significantly better accuracy than existing methods that also use a single RGB-D camera. At the same time, using a single RGB-D camera allows our method to be viewpoint-invariant, without requiring the type of time consuming and complicated calibration process that existing multi-camera viewpoint-invariant methods require. We also show that the proposed method is robust the partial occlusions. We believe that the proposed method can be useful as the basis of a computer vision module for a broader end-to-end fall detection system, in conjunction with other sensors and modules. For example, even the system primarily relies on wearable accelerometers for fall detection, the computer vision module can still serve to improve accuracy, and also to be a fallback option for the case where the subject is not actually wearing the required sensors. The proposed method can thus be combined with such non-vision methods, that use wearable sensors, sound, and/or smart floors, for even higher accuracy and robustness.



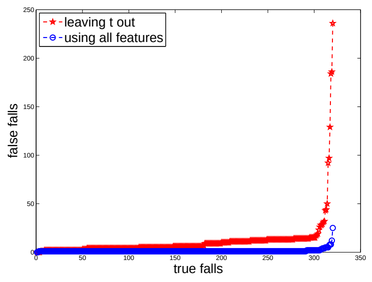
(a) leaving k out



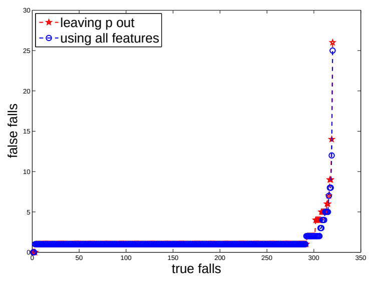
(b) leaving l out



(c) leaving m out

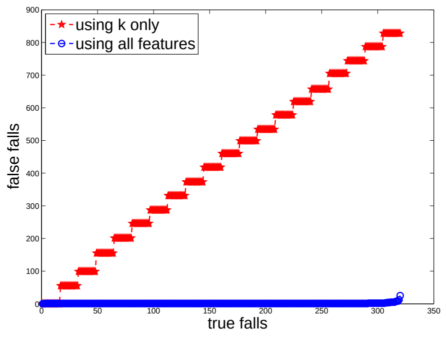


(d) leaving t out

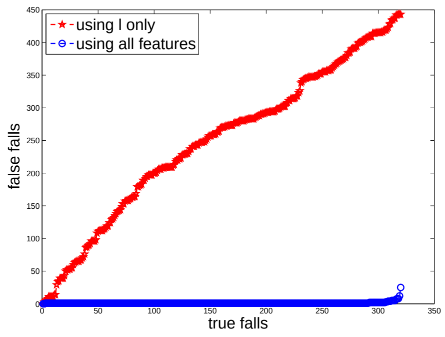


(e) leaving p out

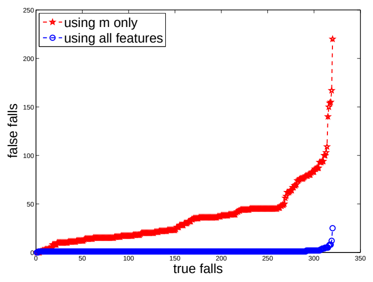
Figure 4.15: Evaluating individual features on **EDF** dataset by measuring the effects of leaving each feature out, and using only the remaining four features.



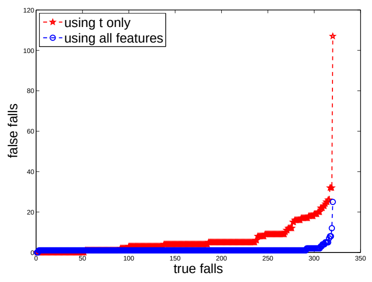
(a) using k only



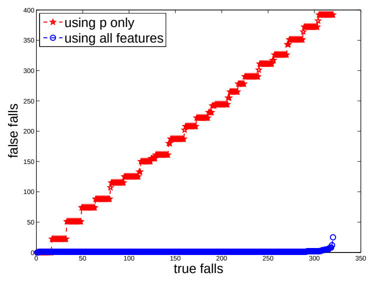
(b) using l only



(c) using m only

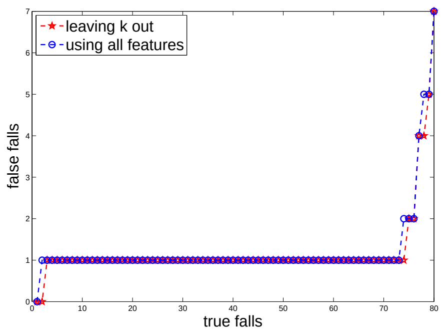


(d) using t only

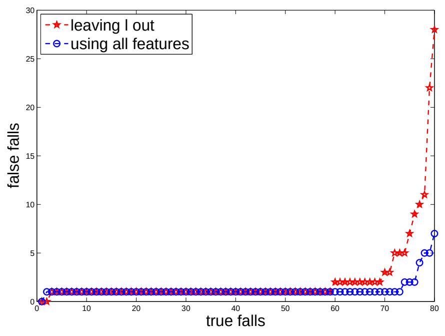


(e) using p only

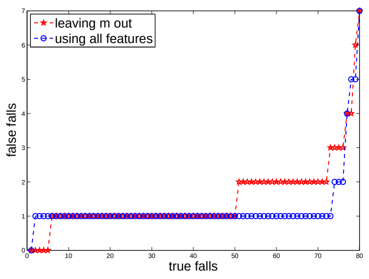
Figure 4.16: Evaluating individual features on **EDF** dataset by measuring the result using each single feature in isolation, compared to using all five features together.



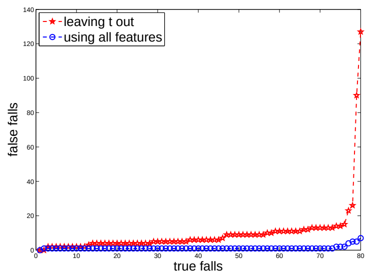
(a) leaving k out



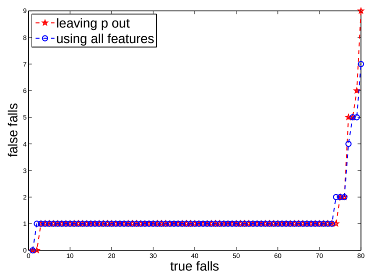
(b) leaving l out



(c) leaving m out

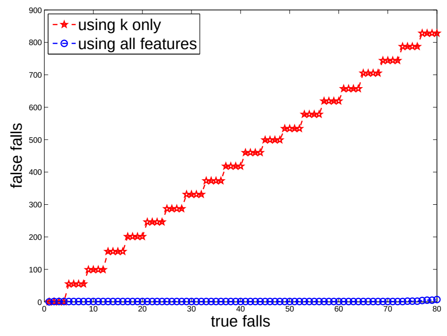


(d) leaving t out

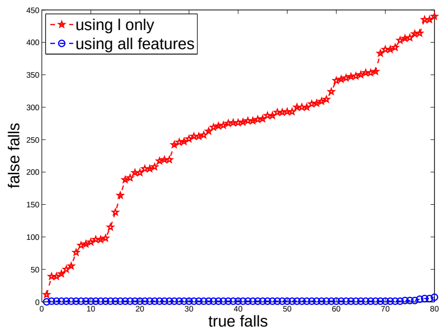


(e) leaving p out

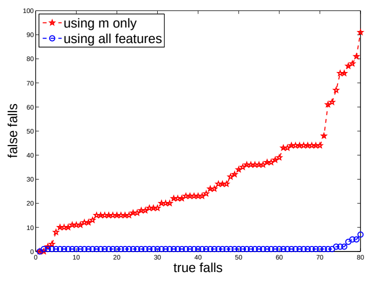
Figure 4.17: Evaluating individual features on **EDF0** dataset by measuring the effects of leaving each feature out, and using only the remaining four features.



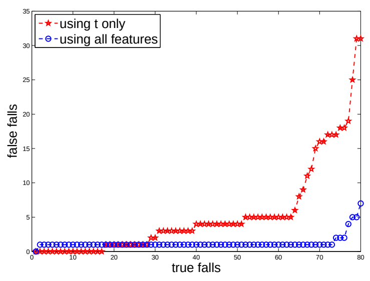
(a) using k only



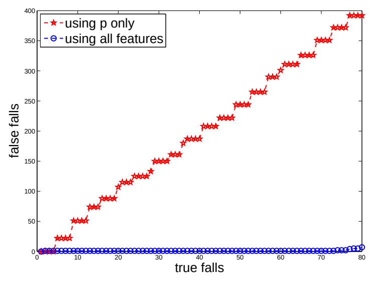
(b) using l only



(c) using m only

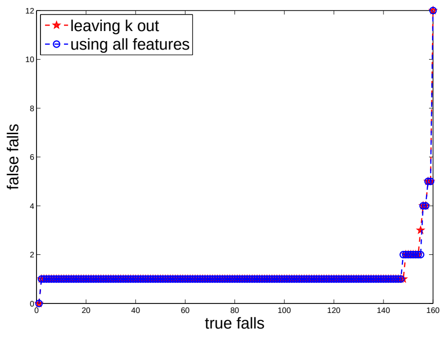


(d) using t only

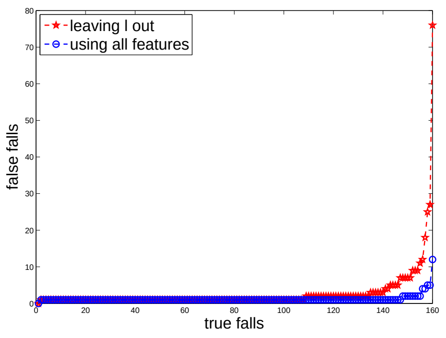


(e) using p only

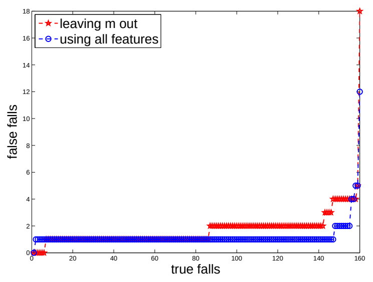
Figure 4.18: Evaluating individual features on **EDF0** dataset by measuring the result using each single feature in isolation, compared to using all five features together.



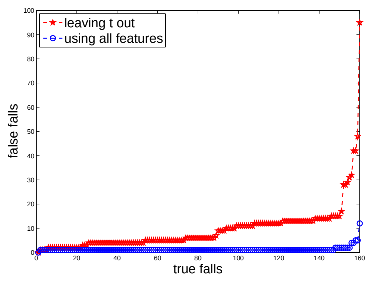
(a) leaving k out



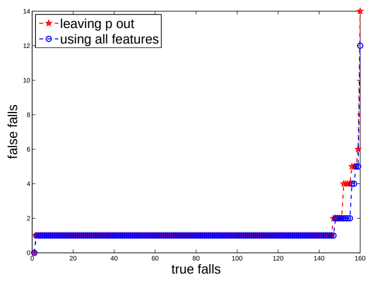
(b) leaving l out



(c) leaving m out

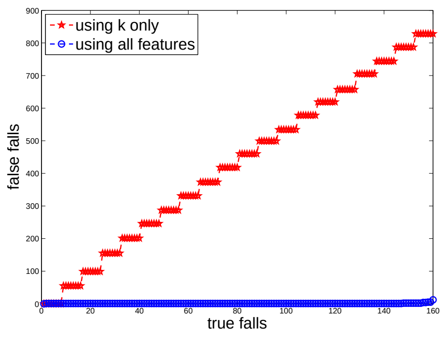


(d) leaving t out

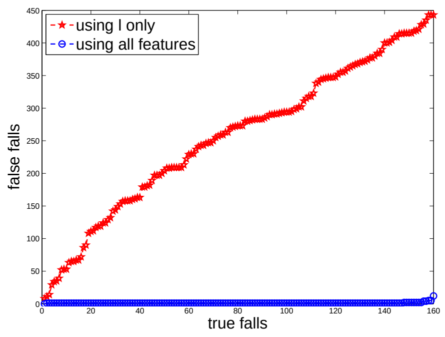


(e) leaving p out

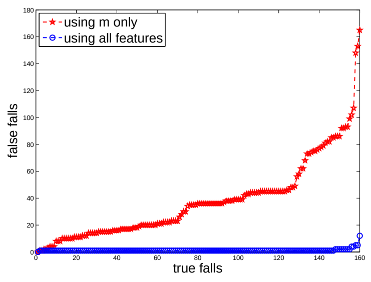
Figure 4.19: Evaluating individual features on **EDF45** dataset by measuring the effects of leaving each feature out, and using only the remaining four features.



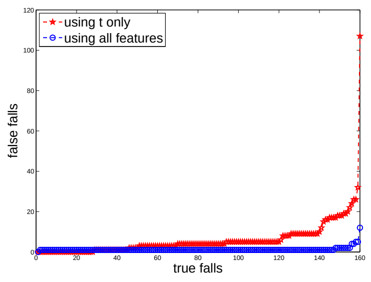
(a) using k only



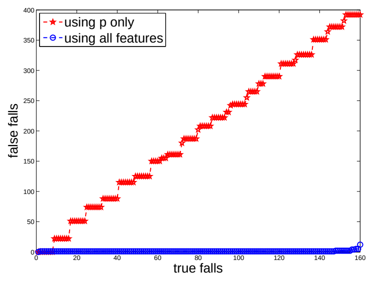
(b) using l only



(c) using m only

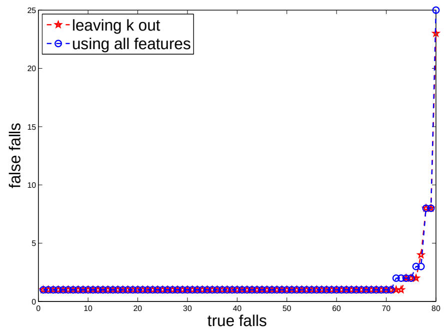


(d) using t only

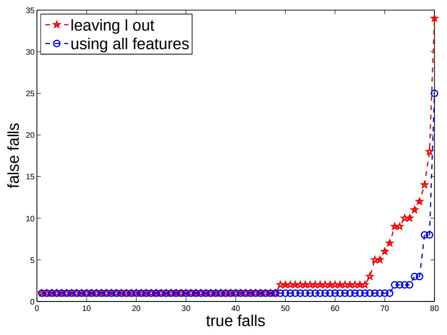


(e) using p only

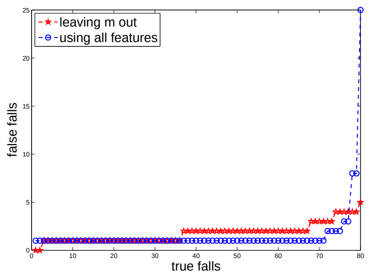
Figure 4.20: Evaluating individual features on **EDF45** dataset by measuring the result using each single feature in isolation, compared to using all five features together.



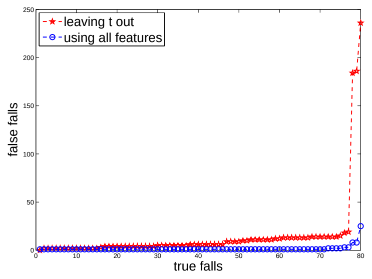
(a) leaving k out



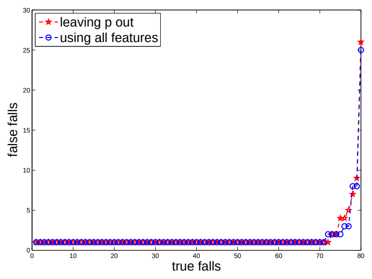
(b) leaving l out



(c) leaving m out

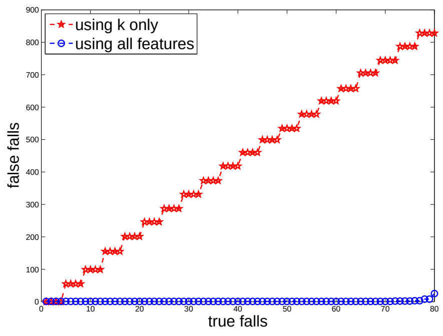


(d) leaving t out

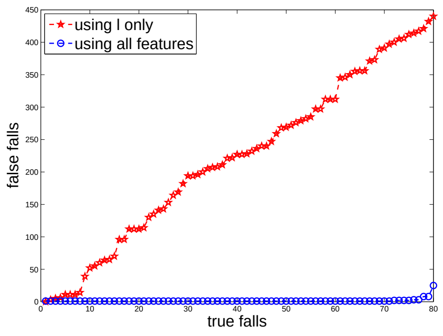


(e) leaving p out

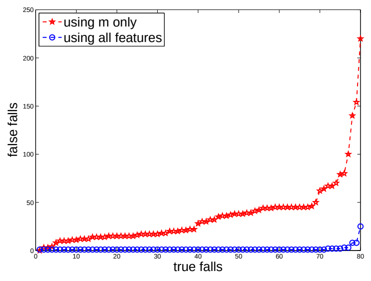
Figure 4.21: Evaluating individual features on **EDF90** dataset by measuring the effects of leaving each feature out, and using only the remaining four features.



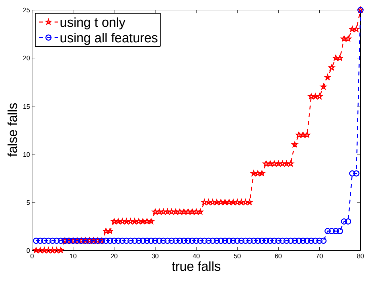
(a) using k only



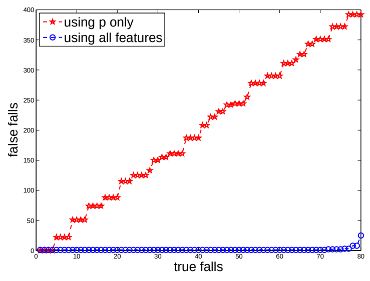
(b) using l only



(c) using m only



(d) using t only



(e) using p only

Figure 4.22: Evaluating individual features on **EDF90** dataset by measuring the result using each single feature in isolation, compared to using all five features together.

CHAPTER 5

EVALUATING DEPTH-BASED COMPUTER VISION METHODS FOR FALL DETECTION UNDER OCCLUSIONS

5.1 Introduction

The growing population of seniors would benefit from systems that include fall detection functionality. To automatically detect falls in a home environment can help ensure the safety of elderly people living alone. In this work we focus on the task of vision-based occluded fall detection. An occluded fall occurs when an object, such as a bed, blocks the sensor's view of the end of the fall, and thus the entire fall is not visible. These occlusions frequently occur at a home environment since a room contains furniture and objects that could be placed between the subject and the camera. Kinect cameras are used to capture the occluded fall benchmark dataset, and the detection is thus based on depth videos.

Several sensor-based approaches have been proposed for fall detection, some recent reviews include [59, 61]. An accelerometer [71, 69, 72] is the most commonly used sensor, and it is often combined with other devices such as gyroscopes [72] and microphone [69]. These approaches do not suffer from the occlusion problem. However, these methods require subjects to actively cooperate by wearing the sensors, which can be problematic and possibly uncomfortable (e.g., wearing sensors while sleeping, detecting falls during a night trip to the restroom).

Several vision-based methods have been proposed for fall detection. They can be broadly classified into two categories: 2D-based and 3D-based. [85, 88, 92, 89] use 2D appearance-based features to detect falls. These methods use a single camera and are viewpoint-dependent. Moving a camera to a different viewpoint (especially to a different height from the floor) would require collecting new training data for that specific viewpoint. The 3D features for fall detection can be extracted from a calibrated multi-camera system [80, 77, 94] or from depth cameras [6, 3, 4, 5]. Multi-camera calibrated systems require time-consuming external camera calibration while depth cameras based systems do not.

Most of these methods did not report any information about the robustness of the system to occlusions aside from [94] and [5]. Auvinet et al. [94] used multiple cameras to cover the whole space so that no occlusion occurred for at least one of these cameras. Thus, the total volume information in the scene can be reconstructed based on the multiple-cameras network. This solution, however, is expensive and difficult to set up (multiple calibrated and synchronized cameras are needed). Rougier et al. [5] classified each frame into two classes: the person is totally occluded or not. If it is an occluded case, the velocity of the person just before the occlusion occurs is used to determine whether a fall has occurred. In the non-occluded case, the height of the person's top relative to the floor plane is used to make a decision.

The focus of this paper is on the evaluation of existing depth-based computer vision approaches for fall detection in the case of occlusions. In particular, we make two contributions:

- A new, extensive and annotated dataset of depth videos, including fall events performed by several individuals, without occlusions and with occlusions. The dataset also includes several non-fall events that can lead to false alarms, such as tying shoelaces or lying on the floor. This dataset is freely available online, together with annotations marking the beginning and end of each fall event (`http://sites.google.com/site/occlusiondataset`).

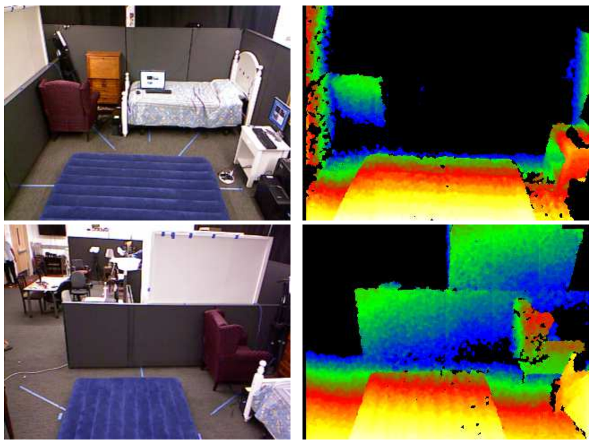


Figure 5.1: Our simulated apartment, seen from the two viewpoints that is used to collect non-occluded fall videos. For each viewpoint we show a color image and a depth image. Depth images are color-coded so that: white indicates small depth values, and yellow, orange, red, green, blue indicate progressively larger depth values. Black indicates invalid depth.

- An evaluation of four recent methods for fall detection using depth cameras [6, 5, 3, 4] on this new dataset, using a user-independent and viewpoint-independent experimental protocol (All training data are collected from a specific viewpoint, and all the test data are collected from another viewpoint). We compare the performance of these methods separately for unoccluded observations and for occluded observations.

5.2 Scene Setup

In our experimental setup, two Kinect depth cameras are set up at two corners of a simulated apartment. The reason for using two Kinects is simply to collect data from two different viewpoints so that the viewpoint-independent experimental protocol can be conducted. Figure 5.1 and Figure 5.2 show the scene setups for recording non-occluded and occluded fall dataset respectively. We can easily see that the two viewpoints in Figure 5.1 are different from the two viewpoints in Figure 5.2.

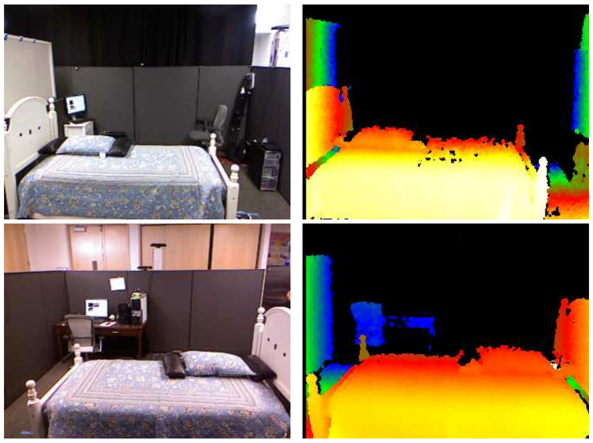


Figure 5.2: Our simulated apartment, seen from the two viewpoints that is used to collect occluded fall videos. For each viewpoint we show a color image and a depth image. Depth images are color-coded so that: white indicates small depth values, and yellow, orange, red, green, blue indicate progressively larger depth values. Black indicates invalid depth.

5.3 Four Existing Fall Detection Methods

In this section, we briefly introduce four existing fall detection methods [6, 5, 3, 4] that use a single depth camera. Since the distance of the person's top relative to the floor plane is the common feature used in all of these methods, we describe how to calculate this height first. To compute the distance between the top of the person and the floor plane, we first need to estimate the equation of the floor plane, and it can be used to compute the distance between a 3D point of the scene and the floor plane.

5.3.1 Floor Plane Estimation

The floor plan can be represented as the following equation:

$$Ax + By + Cz + D = 0 \tag{5.1}$$

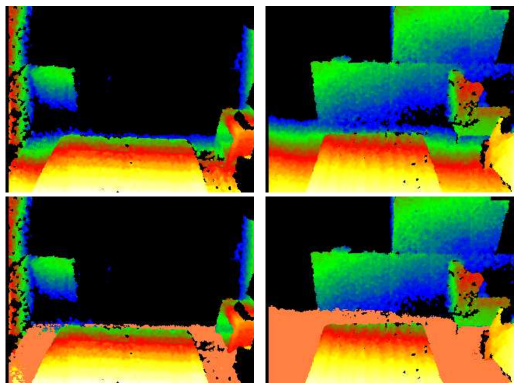


Figure 5.3: Result of floor level detection. Top: a background depth image. Bottom: the same image as on the top, with an orange color superimposed on pixels with height value less than 2cm.

We obtain parameters A, B, C, D directly using the Microsoft Kinect SDK, which has built-in floor estimation functionality. Furthermore, the Kinect SDK provides normalized values for parameters A, B, C, D so that the physical interpretation of D is the height of the camera from the floor. Figure 5.3 shows two examples of floor detection. The pixel is classified as floor if its height value is less than 2cm. It is worth noting that the floor plane detection method in [6] is not fully automatic and requires the user to click on two points in the depth image to specify the vertical orientation in world coordinates.

5.3.2 Person Detection and Height Calculation

In our environment, we can assume that the only moving object is the person. To detect moving objects, we use background subtraction on depth images. The background depth map B is obtained by using the average of the last few frames where no significant motion was observed.

Let $D_n(x, y)$ denote the value at pixel coordinates (x, y) for the n-th depth frame. By comparing $D_n(x, y)$ with the background depth map $B(x, y)$, the moving parts of the current frame can be easily extracted. In particular, using a system-specific threshold T ($T = 200$ in our implementation), a pixel is determined to belong to a moving object if its depth value differs from the corresponding background value by more than T:

$$M_n(x,y) = \begin{cases} 1 & if \quad |D_n(x,y) - B(x,y)| > T \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

Once $M_n(x,y)$ has been computed for all pixels of frame n, connected component analysis is used to identify the largest connected component with values $M_n(x,y) = 1$. We call that component the *person region*.

To estimate the distance from the top of the person to the floor, we first identify the top region of the person as seen in the depth image. Once the person region has been detected in the image, the pixels of that region are sorted based on their row coordinate, and the value at the highest 5 percentile is selected as the "top row" of the region. We denote that "top row" as $T_i^y(n)$. By examining pixels in the person region that are located at the "top row", we identify the median column for those pixels and denote it as $T_i^x(n)$. We can then read the depth value at pixel $[T_i^x(n), T_i^y(n)]$ and denote it as $T^z(n)$.

Given $[T_i^x(n), T_i^y(n), T^z(n)]$, we compute the Kinect-centered world coordinates for that point using Equations 5.3, and from those coordinates, we compute the distance from the top of the person to the ground plane using Equation 5.4, where $[w, h]$ is the width and height of the depth image in pixels, and m_x (1.12032) and m_y (0.84024) are fixed parameters specified in the Kinect documentation provided by Microsoft.

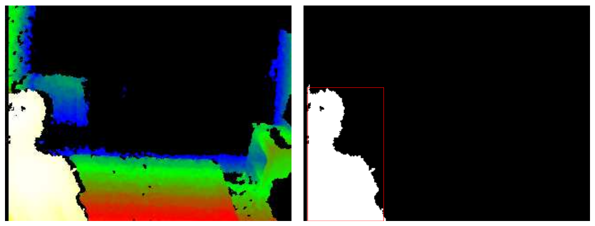


Figure 5.4: The left side shows an example of partial occlusion, while the right side shows the person detection result with a red rectangle indicating the person's location in the depth image. It can easily be seen that the centroid of the silhouette is not the actual centroid of the person.

$$\begin{aligned} T_k^x(n) &= (\frac{T_i^x(n)-1}{w-1}) * m_x * T^z(n) \\ T_k^y(n) &= (\frac{T_i^y(n)-1}{h-1}) * m_y * T^z(n) \\ T_k^z(n) &= T^z(n) \end{aligned} \tag{5.3}$$

$$y_w = \frac{|AT_k^x(n) + BT_k^y(n) + CT_k^z(n) + D|}{\sqrt{A^2+B^2+C^2}} \tag{5.4}$$

5.3.3 Fall Detection

The authors of [5, 3, 4] calculate the height as the distance of the centroid of the person's silhouette to the floor plane. The height of this centroid, however, might not be accurate when the person is partially occluded (Figure 5.4). Instead, we calculate the distance of the person's top to the floor plane as described in section 5.3.2.

- In method [5], the human height relative to the ground is used to detect (non-occluded or partially occluded) falls. When the end of an action is totally occluded by furniture, an analysis of the 3D body velocity prior to occlusion allows detection of the fall.
- The method of [3] uses two rules to describe a fall: the distance of the top of the person to the floor plane being lower than a prefixed value; an unchangeable situation (negligible movements of the person) for at least 4 seconds.

- The method of [4] is also based on two rules: the distance of the top of the person to the floor plane, decreasing 0.40m within a window of about 900ms; the people's silhouette movement remaining negligible for a time window of about 4s.
- Zhang et al. [6] extract five features from a candidate event and use a pre-trained naive Bayesian classifier on these features to determine whether the candidate is a fall or not.

5.4 Experiments

5.4.1 Dataset Description

We created a non-occlusion dataset, **EDF**, and an occlusion dataset, **OCCU**, using Kinect cameras for XBOX 360 with the Microsoft Kinect for Windows SDK Beta at a frame rate of about 30fps. Each of the 5 subjects performed a non-occluded fall along eight directions in each viewpoint in the **EDF** dataset. Each of the 5 subjects performed six occluded falls in each viewpoint in the **OCCU** dataset. Figure 5.5 shows one fall example in every direction. In the **OCCU** dataset, the end of the fall is completely occluded by a bed. Figure 5.6 shows an example of an occluded fall in each viewpoint.

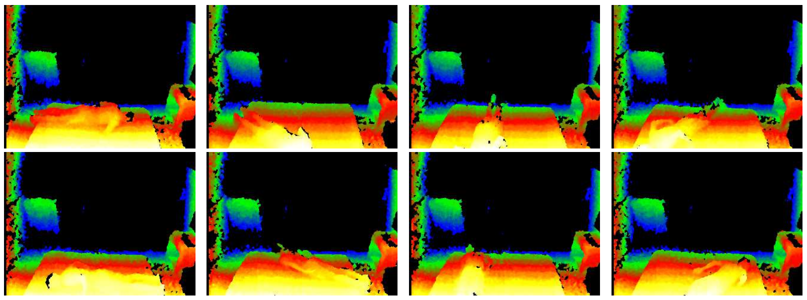


Figure 5.5: A person falls down along eight different directions

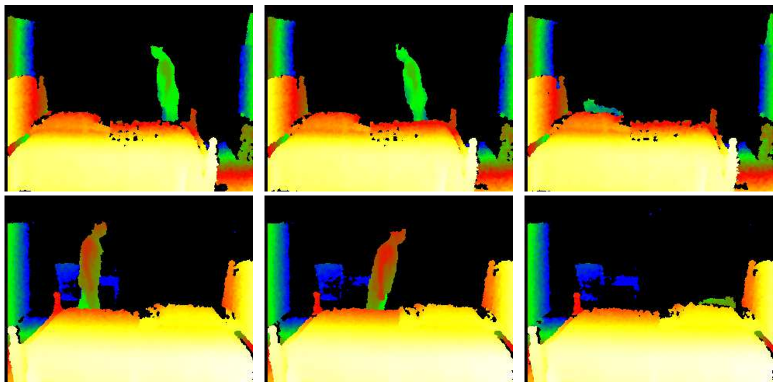


Figure 5.6: Examples of the occluded fall. The top row shows an occluded fall in the first viewpoint while the bottom row shows an occluded fall in the second viewpoint.

The **EDF** dataset is comprised of 25,881 frames and 40 real falls in videos from the first viewpoint, and 24,497 frames and 40 real falls in videos from the second viewpoint. The two viewpoints were recorded at the same time, and thus every event was recorded simultaneously from both viewpoints. Our subjects also performed a total of 30 actions that tend to produce features similar to those of a fall event, namely: 10 examples of picking up something from the floor, 10 cases of sitting on the floor and 10 examples of lying down on the floor.

The **OCCU** dataset includes 25,618 frames and 30 totally occluded falls in videos from the first viewpoint, and 23,703 frames and 30 totally occluded falls in videos from the second viewpoint performed by the same subjects. Each viewpoint was recorded at separate times from the other viewpoint, and thus we had no instances where the same events were recorded simultaneously from both viewpoints. Our subjects also performed a total of 80 actions that tended to produce features similar to those of a fall event, namely: 20 examples of picking up something from the floor (all of them are non-occluded), 20 examples of sitting on the floor (all of them are non-occluded), 20 examples of tying shoelaces (all of them are non-occluded), and

21 examples of lying down on the floor(all of them are totally occluded at the end frame).

The **OCCU1** dataset is a subset of **OCCU**. To generate the **OCCU1** dataset, we removed all examples of lying down on the floor from **OCCU** and created **OCCU1** from the remaining examples. As the primary use of this application is intended for elderly people, and it is unlikely that an elderly person would all of a sudden lie down on the floor, we found it is necessary to conduct a separate evaluation on the dataset by excluding the ”lying down on the floor” events.

5.4.2 Results

As mentioned in Section 5.1, our experiments are both user-independent (to recognize the actions of a user, no training data from that same user is used) and viewpoint-independent (to recognize actions observed from a specific viewpoint, no training data from the same viewpoint is used). Also, because the fall action is a continuous process, if a fall is detected in the current frame, then no fall would be reported in the next 250 frames.

Here we evaluate four depth-based fall detection methods on the three datasets [6, 5, 3, 4]. We have implemented these competitive methods ourselves. Since the thresholds provided by the authors may not be the best choice for our data, we searched and identified the best thresholds for them. The methods of [5, 6] need a training step while the methods of [3, 4] do not, since they are rule based. The experiments using the methods of [5, 6] strictly follow the user-independent and viewpoint-independent protocols.

The methods of [6, 3, 4] did not discuss how to handle occluded falls. When the person is totally occluded by the bed in our simulated environment, the height of the person’s top can not be calculated. For these cases, we simply assign the height

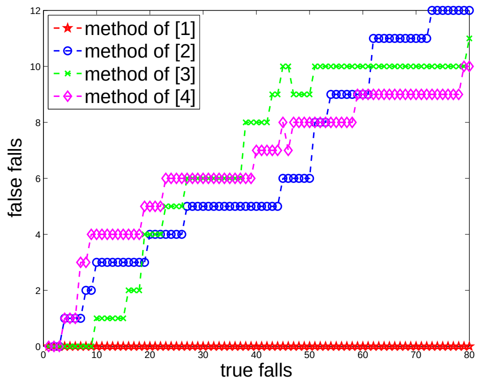


Figure 5.7: Results of four evaluated methods [6, 5, 3, 4] on **EDF** dataset.

value of these frames to the last frame where the person is visible. Rougier et al. [5] classified every frame into two categories: whether the person is visible in the current frame or not. When the person is visible, the human top height relative to the ground is used to detect (not occluded or partially occluded) falls. When the end of an action is totally occluded by the bed, an analysis of the 3D body velocity prior to occlusion allows detection of the fall.

5.4.2.1 Results on **EDF** dataset

Five subjects appear in the videos collected from each viewpoint in **EDF** dataset and there are two viewpoints in total. To train the classifier that will be used for a specific subject and viewpoint, we use as training data only videos of other subjects from the other viewpoint. Thus, to classify a subject's actions as seen from a specific viewpoint, the system does not use training data from the same subject or from the same viewpoint.

Figure 5.7 shows the results on dataset **EDF**. The method proposed by Zhang et al. is significantly more accurate than those of the competitors. If we choose thresholds for each method so as to detect all of the actual falls, the method of

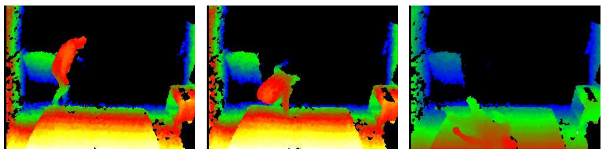


Figure 5.8: Example frames from a false alarm, showing a person lying on the floor.

[6] produces zero false positive, compared to ten false positives by the best of the competitors [4]. A false alarm reported by [4] is shown on Figure 5.8. In Figure 5.8, the person lies down on the floor.

5.4.2.2 Results on **OCCU** dataset

We use the **EDF** dataset to train the models for methods [5, 6] for tests on the **OCCU** dataset. We ensure user independence and viewpoint independence by using different subjects and viewpoints in **EDF** and **OCCU**.

Two parameters are used in the method of [5]: a threshold for the height of the person's top, T, and a threshold for the body velocity, V. If there is no occlusion, the height of the person's top relative to the ground is used to detect a potential fall; if an occlusion occurs, the body velocity just before the occlusion occurs is analyzed to try to determine whether a fall has occurred. Since there are no non-occlusion falls in the **OCCU** dataset, we set a low enough value for T to eliminate all non-occlusion false falls caused by such events as sitting down on the floor, tying the shoelaces, etc., and do not need to worry about missing the detection of any true fall (all of the true falls are totally occluded and thus would be handled by velocity checking). In this way, we favor method [5] over other competitive methods, since they treat every candidate event, either occluded fall or non-occluded fall, equally.

The result is shown in the left side of Figure 5.9. The method of [5] is significantly better than other methods. But in detecting all the actual falls, it also

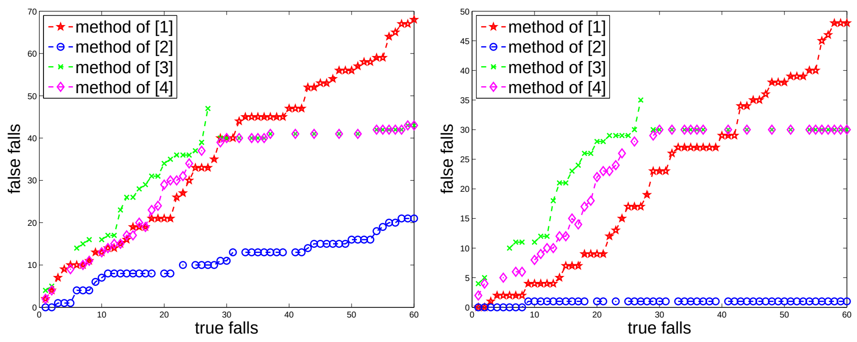


Figure 5.9: Results of four evaluated methods [6, 5, 3, 4] on **OCCU** dataset and **OCCU1** dataset respectively. The left side is about **OCCU** dataset while the right side is for **OCCU1** dataset

reports 21 false positives. Most of these false positives are lying down on the floor. Figure 5.10 shows an example of this action. This action has similar features to a true occluded fall, and thus it is also hard to be correctly classified by all of other evaluated methods. The methods of [3, 4, 6] also fail on other actions, like picking up something from the floor and tying shoelaces (Figure 5.11).

5.4.2.3 Results on **OCCU1** dataset

The training protocol and parameters selection are the same as the experiment conducted on the **OCCU** dataset (section 5.4.2.2). The result is shown in the right side of Figure 5.9. The method of [5] is still significantly better than other methods. To detect all the actual falls, only one false positive is reported. This verifies the fact that most of the false positives produced by the method of [5] on the dataset **OCCU** come from the event of lying down on the floor. By comparing the experimental results on **OCCU** and **OCCU1** dataset for other evaluated methods [3, 4, 6], we can clearly see that methods [3, 4, 6] also fail in most cases of lying down on the floor.

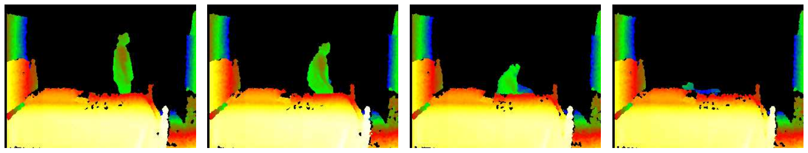


Figure 5.10: Example frames from a false alarm, showing a person lying down on the floor.

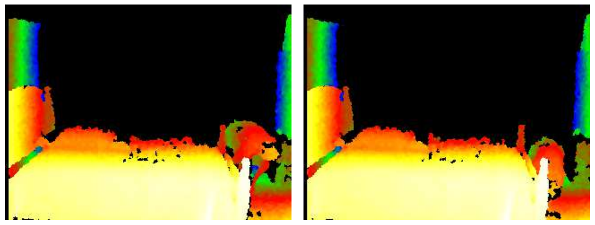


Figure 5.11: The left side is an example of tying shoelaces and the right side is an example of picking up something from the floor.

5.5 Discussion and Conclusion

For the non-occluded falls, the method proposed by Zhang et al. [6] achieves the best result, while for the occluded falls, the method proposed by Rougier et al. [5], which first classifies a candidate event into occluded case or non-occluded case and then checks it using the specific feature of its corresponding class, is better than other methods. We can conclude that it is necessary to handle occluded and non-occluded falls separately. Although the method of [5] is significantly better than the other three methods, we note that the velocity feature adopted by [5] is not adequate for distinguishing lying down from falling down.

For future work, we plan to explore more representative features, like acceleration, to make the method capable of distinguishing occluded fall from other similar occluded false events. At the same time, due to the low cost of the Kinect, it is practical to deploy several depth cameras, so as to cover most or all of the apartment area. In that scenario, the decisions from all the cameras are aggregated into a final decision for a candidate event. By doing so, we believe that the occluded cases can be handled more accurately.

CHAPTER 6

DISCUSSION AND FUTURE WORK

6.1 Discussion of Contributions

This thesis investigated methods for vision-based ASL recognition and fall detection for assistive environments. First, Chapter 2 introduced a semi-automatic vision-based ASL recognition system. This system includes two major parts: hand trajectory extraction and sign search. The hand trajectory extraction is achieved by the user's interaction which makes the system semi-automatic. Sign search is formulated as a template matching problem. The hand trajectory of the query sign is compared with each pre-computed model in a large database of sign videos. The similarity measure between the hand trajectory of the query sign and the model is calculated by a well-known similarity measure algorithm, Dynamic Time Warping (DTW). In Chapter 3, towards making the recognition system more automatic, we first evaluated four recently published representative hand detectors on three sign datasets. Experimental results showed that to include real hands, a large number of false detections are also reported, which means a lot of manual work must be involved to do the corrections if directly using these hand detection methods as the hand detector for the proposed sign recognition system. A natural improvement is to combine these four methods. We proposed a two-stage hand detector. The first stage uses a skin and motion detector to generate hand bounding boxes. Several features are calculated for each box, and a second stage pre-trained SVM classifier is employed to compute a final score for each box using these features. Experimental results demonstrate that the combination significantly improved the detection accuracy under the user-dependent experimental protocol.

On the topic of fall detection for assistive environments, Chapter 4 proposed a statistical method to detect falls based on depth images captured by a Kinect camera. Using the depth images, we detect and track the person in both image coordinates and world coordinates. From every frame, features are extracted based on the position and velocity of the person. A Bayesian classifier is built on top of these features. For each frame, the system uses this Bayesian classifier to decide whether a fall event has just occurred. Modeling fall events using a camera-independent world coordinate system allows our method to be viewpoint invariant and not to be so sensitive to the choice of position and viewing direction for the camera. Using observations from a single camera and requiring minimal effort to adjust the system once the camera has been moved are key differentiating features from most existing computer vision methods for fall detection, which are typically either viewpoint-dependent or require multiple cameras and a careful and time-consuming calibration process. Especially for calibrated multi-camera systems, we should keep in mind that the calibration process needs to be repeated every time a single camera is intentionally or accidentally removed. A key feature of our experiments is that we record all training data from a specific camera position and all test data from another specific camera position, different from the position used to record the training data. Experimental results showed that moving the camera to a new position does not affect accuracy. Several methods have been proposed in the literature that use 3D features; thus, in theory such methods are viewpoint invariant. However, we are not aware of other methods that have been explicitly evaluated using such a protocol, where the viewpoint (or overall multi-camera setup, for multi-camera systems) for the test data is ensured to be different than the viewpoint used for the training data. We also contributed several publicly available fall datasets. We showed one non-occlusion fall dataset and three partial occlusion datasets in Chapter 4. Researchers can use them as benchmark datasets to develop their own fall detection methods. As far we know, these are the first publicly available fall datasets captured by Kinect depth cameras.

The fall detection method proposed in Chapter 4 has proven its capability in handling non-occlusion falls. However, for complete occlusion falls, which refer to the end of the fall being totally occluded by a certain object, there is no conclusion yet. Chapter 5 first contributed a complete occlusion fall dataset and made it freely available online. Then we evaluated the proposed fall detection and three other related publications on this occlusion dataset. Although the proposed method does not get the best detection accuracy, the evaluation can serve as a benchmark for other researchers to assess their own fall detectors. And it should be noted that this work is one of the earliest works that discuss the complete occlusion falls.

6.2 Future Work

On the topic of hand detection, hand tracking is especially useful when two hands interact with each other. Figure 6.1 shows two sign examples where one hand interacts with the other hand. In such cases, the hand detection method based on a single frame really does not work since the features cannot even be extracted for the occluded hand. A possible solution is to "guess" the occluded hand's location based on the results of the previous frames and the following frames, which is exactly what tracking methods do.

On the topic of fall detection, the proposed method is not quite good enough to serve as a standalone system for fall detection. The overall fall detection system can contain additional modules, both to improve accuracy and to include additional functionality, such as actually sending an alert about the detected fall to someone

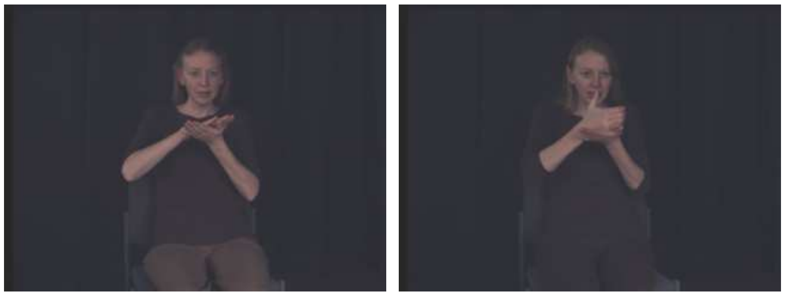


Figure 6.1: Two sign examples where one hand interacts with the other hand.

who cares. Sound processing and speech recognition modules, which would help significantly towards obtaining a robust system, remain as future work. Additional features, which are extracted from sound processing, can be used for classifying a candidate fall event. Speech recognition can be used so that the system initiates a dialogue with the subject in the case when a fall has been detected. For example, the system can ask “Are you okay?” and the user can respond to indicate that there was no actual fall and no need to issue an alert.

REFERENCES

[1] P. Buehler, M. Everingham, D. P. Huttenlocher, and A. Zisserman, “Upper body detection and tracking in extended signing sequences,” *International Journal of Computer Vision*, vol. 95, no. 2, pp. 180–197, 2011.

[2] A. Mittal, A. Zisserman, and P. H. S. Torr, “Hand detection using multiple proposals,” in *British Machine Vision Conference*, 2011.

[3] G. Diraco, A. Leone, and P. Siciliano, “An active vision system for fall detection and posture recognition in elderly healthcare,” in *Design, Automation & Test in Europe Conference & Exhibition*, March 2010.

[4] A. Leone, G. Diraco, and P. Siciliano, “Detecting falls with 3d range camera in ambient assisted living applications: A preliminary study,” *Medical Engineering & Physics*, vol. 33, pp. 770–781, 2011.

[5] C. Rougier, E. Auvinet, J. Rousseau, M. Mignotte, and J. Meunier, “Fall detection from depth map video sequences,” in *International Conference on Smart Homes and Health Telematics*, 2011.

[6] Z. Zhang, W. Liu, V. Metsis, and V. Athitsos, “A viewpoint-independent statistical method for fall detection,” in *Pattern Recognition (ICPR), 2012 21st International Conference on.* IEEE, 2012, pp. 3626–3630.

[7] H. L. R. Hoffmeister and B. Bahan, *A Journey into the Deaf-World.* San Diego, CA: DawnSign Press, 1996.

[8] J. Schein, *At home among strangers.* Washington, DC: Gallaudet U. Press, 1989.

[9] V. Athitsos, C. Neidle, S. Sclaroff, J. Nash, A. Stefan, Q. Yuan, and A. Thangali, "The american sign language lexicon video dataset," in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on.* IEEE, 2008, pp. 1–8.

[10] [Online]. Available: http://www.who.int/ageing/publications/Falls_prevention7March.pdf

[11] Q. Yuan, S. Sclaroff, and V. Athitsos, "Automatic 2d hand tracking in video sequences." in *IEEE Workshop on Applications of Computer Vision*, 2005, pp. 250–256.

[12] J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff, "A unified framework for gesture recognition and spatiotemporal gesture segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 31, no. 9, pp. 1685–1699, 2009.

[13] L. Karlinsky, M. Dinerstein, D. Harari, and S. Ullman, "The chains model for detecting parts by their context," in *IEEE International Conference on Computer Vision and Pattern Recognition(CVPR)*, 2010, pp. 25–32.

[14] e. Clayton Valli, *The Gallaudet Dictionary of American Sign Language.* Washington, DC: Gallaudet U. Press, 2006.

[15] R. A. Tennant and M. G. Brown, *The American Sign Language Handshape Dictionary.* Gallaudet U. Press, 2010.

[16] Z. Zhang, R. Alonzo, and V. Athitsos, "Experiments with computer vision methods for hand detection," in *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments*, 2011, p. 21.

[17] Z. Zhang, C. Conly, and V. Athitsos, "Hand detection on sign language videos," in *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments.* ACM, 2014, p. 26.

[18] W. Liu, Y. Fan, T. Lei, and Z. Zhang, "Human gesture recognition using orientation segmentation feature on random rorest," in *Signal and Information Processing (ChinaSIP), 2014 IEEE China Summit & International Conference on*. IEEE, 2014, pp. 480–484.

[19] C. Conly, Z. Zhang, and V. Athitsos, "An evaluation of rgb-d skeleton tracking for use in large vocabulary complex gesture recognition," in *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments*. ACM, 2014, p. 13.

[20] W. Liu, Y. Fan, Z. Zhang, and Z. Li, "Rgbd video based human hand trajectory tracking and gesture recognition system," *Mathematical Problems in Engineering*, 2015.

[21] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and information systems*, vol. 7, no. 3, pp. 358–386, 2005.

[22] D. Zhang, S. Han, J. Zhao, Z. Zhang, C. Qu, Y. Ke, and X. Chen, "Image based forest fire detection using dynamic characteristics with artificial neural networks," in *Artificial Intelligence, 2009. JCAI'09. International Joint Conference on*. IEEE, 2009, pp. 290–293.

[23] D. Zhang, J. Zhao, J. Zhao, S. Han, Z. Zhang, C. Qu, and Y. Ke, "A new color-based segmentation method for forest fire from video image," in *Future BioMedical Information Engineering, 2008. FBIE'08. International Seminar on*. IEEE, 2008, pp. 41–44.

[24] Z. Zhang, J. Zhao, Z. Yuan, D. Zhang, S. Han, and C. Qu, "Color based segmentation and shape based matching of forest flames from monocular images," in *Multimedia Information Networking and Security, 2009. MINES'09. International Conference on*, vol. 1. IEEE, 2009, pp. 625–628.

[25] H. Cooper and R. Bowden, "Large lexicon detection of sign language," in *Proceedings of the 2007 IEEE international conference on Human-computer interaction*, ser. HCI'07, 2007, pp. 88–97.

[26] A. Farhadi, D. A. Forsyth, and R. White, "Transfer learning in sign language," in *CVPR'07*, 2007.

[27] M. Kölsch and M. Turk, "Robust hand detection," in *IEEE International Conference on Automatic Face and Gesture Recognition(AFGR)*, 2004, pp. 614–619.

[28] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2001, pp. 511–518.

[29] E. J. Ong and R. Bowden, "A boosted classifier tree for hand shape detection," in *Face and Gesture Recognition*, 2004, pp. 889–894.

[30] Z. Zhang, R. Alonzo, and V. Athitsos, "Experiments with computer vision methods for hand detection," in *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments*. ACM, 2011, p. 21.

[31] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Transactions on Computers*, vol. 22, no. 1, pp. 67–92, 1973.

[32] P. Buehler, M. Everingham, D. P. Huttenlocher, and A. Zisserman, "Long term arm and hand tracking for continuous sign language TV broadcasts," in *British Machine Vision Conference(BMVC)*, 2008.

[33] M. P. Kumar, A. Zisserman, and P. H. S. Torr, "Efficient discriminative learning of parts-based models," in *ICCV*, 2009, pp. 552–559.

[34] T. Pfister, J. Charles, M. Everingham, and A. Zisserman, "Automatic and efficient long term arm and hand tracking for continuous sign language tv broadcasts," in *British Machine Vision Conference (BMVC)*, 2012.

[35] H. Trinh, Q. Fan, P. Gabbur, and S. Pankanti, “Hand tracking by binary quadratic programming and its application to retail activity recognition,” in *CVPR*, 2012, pp. 1902–1909.

[36] V. I. Morariu, D. Harwood, and L. S. Davis, “Tracking people’s hands and feet using mixed network and/or search,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 99, no. PrePrints, p. 1, 2012.

[37] H. Zhou and T. S. Huang, “Tracking articulated hand motion with eigen dynamics analysis,” in *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ser. ICCV ’03, 2003.

[38] M. de La Gorce, N. Paragios, and D. J. Fleet, “Model-based hand tracking with texture, shading and self-occlusions,” in *CVPR*, 2008.

[39] B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla, “Model-based hand tracking using a hierarchical bayesian filter,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 9, pp. 1372–1384, 2006.

[40] B.-J. Chen, C.-M. Huang, T.-E. Tseng, and L.-C. Fu, “Robust head and hands tracking with occlusion handling for human machine interaction,” in *IROS*, 2012, pp. 2141–2146.

[41] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, “Efficient Model-based 3D Tracking of Hand Articulations using Kinect,” in *British Machine Vision Conference*, Dundee, UK, 2011.

[42] E. B. Sudderth, M. I. M, W. T. Freeman, and A. S. Willsky, “Visual hand tracking using nonparametric belief propagation,” in *Propagation, IEEE Workshop on Generative Model Based Vision*, 2004, p. 189.

[43] R. Y. Wang and J. Popović, “Real-time hand-tracking with a color glove,” *ACM Trans. Graph.*, vol. 28, no. 3, pp. 63:1–63:8, July 2009.

[44] I. Oikonomidis, N. Kyriazis, and A. Argyros, "Tracking the articulated motion of two strongly interacting hands," in *CVPR 2012*. IEEE, June 2012.

[45] Z. Zhang, J. Zhao, D. Zhang, C. Qu, Y. Ke, and B. Cai, "Contour based forest fire detection using fft and wavelet," in *Computer Science and Software Engineering, 2008 International Conference on*, vol. 1. IEEE, 2008, pp. 760–763.

[46] D. Zhang, C. Qu, J. Zhao, Z. Zhang, Y. Ke, B. Cai, M. Qiao, and H. Zhang, "Eye contour extraction method from monocular image with monkey face," in *Intelligent Information Technology Application Workshops, 2008. IITAW'08. International Symposium on.* IEEE, 2008, pp. 636–639.

[47] D. Zhang, C. Qu, J. Zhao, Z. Zhang, Y. Ke, S. Han, M. Qiao, and H. Zhang, "Extraction and parameterization of eye contour from monkey face in monocular image," in *Advancing Computing, Communication, Control and Management.* Springer, 2010, pp. 182–189.

[48] V. Athitsos, J. Wang, S. Sclaroff, and M. Betke, "Detecting instances of shape classes that exhibit variable structure," in *European Conference on Computer Vision (ECCV)*, 2006, pp. 121–134.

[49] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla, "Shape context and chamfer matching in cluttered scenes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003, pp. 127–133.

[50] M. Jones and J. Rehg, "Statistical color models with application to skin detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1999, pp. I:274–280.

[51] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2, 1999.

[52] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Video-Based Surveillance Systems*, 2002, pp. 135–144.

[53] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 9, pp. 1627–1645, 2010.

[54] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, 2005, pp. 886–893.

[55] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "From contours to regions: An empirical evaluation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[56] J. Zhao, Z. Zhang, S. Han, C. Qu, Z. Yuan, and D. Zhang, "Svm based forest fire detection using static and dynamic features," *Computer Science and Information Systems*, vol. 8, no. 3, pp. 821–841, 2011.

[57] J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff, "Simultaneous localization and recognition of dynamic hand gestures," in *Application of Computer Vision, 2005. WACV/MOTIONS'05 Volume 1. Seventh IEEE Workshops on*, vol. 2. IEEE, 2005, pp. 254–260.

[58] W. Liu, Y. Fan, Z. Zhong, and T. Lei, "A new method for calibrating depth and color camera pair based on kinect," in *Audio, Language and Image Processing (ICALIP), 2012 International Conference on*. IEEE, 2012, pp. 212–217.

[59] M. Kangas, A. Konttila, P. Lindgren, I. Winblad, and T. Jamsa, "Comparison of low-complexity fall detection algorithms for body attached accelerometers," *Gait & Posture*, vol. 28, pp. 285–291, 2008.

[60] N. Noury, A. Fleury, P. Pumeau, A. K. Bourke, G. O. Laighin, V. Rialle, and J. E. Lundy, "Fall detection - principles and methods," in *IEEE Conference on Engineering in Medicine and Biology Society*, 2007, pp. 1663–1666.

[61] T. Shany, S. J. Redmond, M. R. Narayanan, and N. H. Lovell, "Sensors-based werable systems for monitoring of human movement and falls," *IEEE Sensors Journal*, vol. pp, pp. 1–13, 2011.

[62] M. Mubashir, L. Shao, and L. Seed, "A survey on fall detection: Principles and approaches," *Neurocomputing*, vol. 100, pp. 144–152, January 2013.

[63] A. Bourke and G. Lyons, "A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor," *Medical Engineering & Physics*, vol. 30, pp. 84–90, January 2007.

[64] A. Bourke, J. O'Brien, and G. Lyons, "Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm," *Gait & Posture*, vol. 26, pp. 194–199, July 2006.

[65] U. Lindemann, A. Hock, M. Stuber, W. Keck, and C. Becker, "Evaluation of a fall detector based on accelerometers: A pilot study," *Medical and Biological Engineering and Computing*, vol. 43, no. 5, pp. 548–551, October 2005.

[66] N. Noury, P. Barralon, G. Virone, P. Boissy, M. Hamel, and P. Rumeau, "A smart sensor based on rules and its evaluation in daily routines," in *IEEE conference on Engineering in Medicine and Biology Society*, 2003.

[67] T. Zhang, J. Wang, P. Liu, and J. Hou, "Fall detection by embedding an accelerometer in cellphone and using kfd algorithm," *International Journal of Computer Science and Network Security*, vol. 6, no. 10, pp. 277–284, October 2006.

[68] J. Chen, K. Kwong, D. Chang, J. Luk, and R. Bajcsy, "Wearable sensors for reliable fall detection," in *IEEE Conference on Engineering in Medicine and Biology Society*, 2005, pp. 3551–3554.

[69] A.Leone, G.Diraco, C.Distance, P.Siciliano, M.Malfatti, L.Gonzo, M.Grassi, A.Lombardi, G.Rescio, P.Malcovati, V.Libal, J.Huang, and G.Potamianos, "A multi-sensor approach for people fall detection in home environment," in *IEEE Workshop on European Conference Computer Vision for Multi-camera and Multi-modal Sensor Fusioin Algorithms and Applications*, 2008.

[70] S. Luo and Q. Hu, "A dynamic motion pattern analysis approach to fall detection," in *Biomedical Circuits and Systems*, 2004.

[71] C.-F. Lai, S.-Y. Chang, H.-C. Chao, and Y.-M. Huang, "Detection of cognitive injured body region using multiple triaxial accelerometers for elderly falling," *IEEE Sensors Journal*, vol. 11, pp. 763–770, 2011.

[72] Q. Li, J. A. Stankovic, M. A. Hanson, T. Barth, J. Lach, and G. Zhou, "Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information," in *Werable and Implantable Body Sensor Networks*, 2009.

[73] M. Kangas, A. Konttila, I. Winblad, and T. Jamsa, "Determination of simple thresholds for accelerometry-based parameters for fall detection," in *IEEE Engineering in Medicine and Biology Society*, 2007.

[74] J.Y.Hwang, J.M.Kang, Y.W.Jang, and H.C.Kim, "Development of novel algorithm and real-time monitoring ambulatory system using bluetooth module for fall detection in elderly," in *Engineering in Medicine and Biology Society*, 2004.

[75] Y. Lee, J. Kim, M. Son, and M. Lee, "Implementation of accelerometer sensor module and fall detection monitoring system based on wireless sensor network," in *Engineering in Medicine and Biology Society*, 2007.

[76] H. Rimminen, J. Liindstrom, M. Linnavuo, and R. Sepponen, "Detection of falls among the elderly by a floor sensor using the electric near field," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, pp. 1475–1476, 2010.

[77] R. Cucchiara, A. Prati, and R. Vezzani, "A multi-camera vision system for fall detection and alarm generation," *Expert Systems*, vol. 24, pp. 334–345, 2007.

[78] E. Auvinet, F. Multon, A. St-Arnaud, J. Rousseau, and J. Meunier, "Fall detection using body volume recontruction and vertical repartition analysis," in *Proceedings of the 4th international conference on Image and signal processing*, 2010, pp. 376–383.

[79] ——, "Fall detection with multiple cameras: An occlusion-resistant method based on 3-d silhouette vertical distribution," *Information Technology in Biomedicine*, vol. 15, pp. 290–300, 2011.

[80] D. Anderson, R. H. Luke, J. M. Keller, M. Skubic, M. Rantz, and M. Aud, "Linguistic summarization of video for fall detection using voxel person and fuzzy logic," *Computer Vision and Image Understanding*, vol. 113, pp. 80–89, January 2009.

[81] D. Hung and H. Saito, "The estimation of heights and occupied areas of humans from two orthogonal views for fall detection," *IEEJ Transactions on Electronics and Information and Systems*, vol. 133, January 2013.

[82] H. Nait-Charif and S. J. McKenna, "Activity summarisation and fall detection in a supportive home environment," in *Internation Conference on Pattern Recognition(ICPR)*, 2004.

[83] S.-G. Miaou, P.-H. Sung, and C.-Y. Huang, "A customized human fall detection system using omni-camera images and personal information," in *Distributed Diagnosis and Home Healthcare*, 2006.

[84] J. Tao, M. Turjo, M.-F. Wong, M. Wang, and Y.-P. Tan, "Fall incidents detection for intelligent video surveillance," in *Information, Communications and Signal Processing*, 2005.

[85] F. Nater, H. Grabner, T. Jaeggli, and L. V. Gool, "Tracker trees for unusual event detection," in *ICCV Workshop on Visual Surveillance*, 2009.

[86] B. Mirmahboub, S. Samavi, N. Karimi, and S. Shirani, "Automatic monocular system for human fall detection based on variations in silhouette area," *IEEE Transactions on Biomedical Engineering*, vol. 60, pp. 427–436, February 2013.

[87] K. Tra and T. Pham, "Human fall detection based on adaptive background mixture model and hmm," in *International Conference on Advanced Technologies for Communications*, Oct 2013, pp. 95–100.

[88] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Robust video surveillance for fall detection based on human shape deformation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, pp. 611–622, 2011.

[89] A. Zweng, S. Zambanini, and M. Kampel, "Introducing a statistical behavior model into camera-based fall detection," in *Proceedings of the 6th international conference on Advances in visual computing - Volume Part I*, 2010.

[90] Z. Fu, E. Culurciello, P. Lichtsteiner, and T. Delbruck, "Fall detection using an address-event temporal contrast vision sensor," in *Circuits and systems*, 2008.

[91] M. Lustrek and B. Kaluza, "Fall detection and activity recognition with machine learning," *Informatica*, vol. 33, pp. 205–212, 2009.

[92] Z. Zhang, E. Becker, R. Arora, and V. Athitsos, "Experiments with computer vision methods for fall detection," in *Conference on Pervasive Technologies Related to Assistive Environments (PETRA)*, 2010.

[93] Z. Zhang, C. Conly, and V. Athitsos, "Evaluating depth-based computer vision methods for fall detection under occlusions," in *Advances in Visual Computing*. Springer, 2014, pp. 196–207.

[94] E. Auvinet, F. Multon, A. Saint-Arnaud, J. Rousseau, and J. Meunier, "Fall detection with multiple cameras: An occlusion-resistant method based on 3-d

silhouette vertical distribution," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 15, no. 2, pp. 290–300, 2011.

BIOGRAPHICAL STATEMENT

Zhong Zhang was born in Hubei, China, in 1987. He received his B.S. degree from Chongqing University, China, in 2007, his M.S. degree from Wuhan University, China, in 2009 and his Ph.D. degrees from The University of Texas at Arlington in 2015, all in Computer Science. His current research interest includes computer vision and machine learning. His recent work has focused on gesture and sign language recognition, human motion analysis and object detection.