UNIVERSITY OF CALIFORNIA
RIVERSIDE

Automated Video-Based Fall Detection

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Alex Daniel Edgcomb

June 2014

Dissertation Committee:

Dr. Frank Vahid, Chairperson
Dr. Eamonn Keogh
Dr. Victor Zordan
Dr. Tony Givargis

The Dissertation of Alex Daniel Edgcomb is approved:

_____

_____

_____

_____
Committee Chairperson

University of California, Riverside

ACKNOWLEDGMENTS

Many people, both on and off campus, have given me support and guidance during my graduate studies.

My greatest thanks go to my closest supporters: my wife, Juliet Beni Edgcomb; my mom, Carol Ann Edgcomb; my dad, Daniel Leuty Edgcomb; and, my Nana, JoEllen Acampora.

I truly appreciate the guidance, support, confidence-building, and refined-thinking skills from my Ph.D. advisor, Frank Vahid. I would also like to thank my dissertation committee, including Professors Eamonn Keogh, Tony Givargis, and Victor Zordan.

I appreciate the fantastic academic environment that enables students to focus on studies of the University of California, Riverside, especially the Computer Science and Engineering Department.

Lastly, I would thank the Association of Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE) organizations for providing constructive criticisms and opportunities for publishing my works.

DEDICATION

To my grandparents, Ken and Helen McLaughlin, and JoEllen and Wayne Acampora.

ABSTRACT OF THE DISSERTATION

Automated Video-Based Fall Detection

by

Alex Daniel Edgcomb

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, June 2014
Dr. Frank Vahid, Chairperson

Automatically detecting falls is a desired part of caring for a live-alone senior. Researchers have developed various video-based fall detection methods, including moving-region-based 3D-projection-based methods. We introduce a video-based fall detection method that is simpler and more efficient than previous methods, while being equally or more accurate. The method is based on the moving-regions, represented as a minimum bounding rectangle (MBR) around the person in video. The method uses fall detectors that use a particular feature of the MBR, such as height or width, to contribute a fall likelihood score. Many fall likelihood scores can be combined to produce a single-camera fall score. Multiple cameras can be combined to produce a multi-camera fall score. We evaluated our method on a commonly used video data set featuring a middle-

aged, male actor performing falls and in-home activities. We report accuracy as sensitivity and specificity, and efficiency as frames per second (FPS). The method for a single-camera achieved 0.960 sensitivity and 0.995 specificity, and for 2 or more cameras achieved at least 0.990 sensitivity and at least 0.990 specificity. The method runs at 32.1 FPS while single-threaded on a 3.30 GHz Xeon processor. Our method was more accurate than the state-of-the-art MBR-based methods, while being equally efficient. Also, our method was about 10x more efficient than the state-of-the-art projection-based algorithms, while being more accurate with 3 cameras and equally accurate with 4+ cameras.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1. INTRODUCTION OF DISSERTATION

1.1 VIDEO-BASED FALL DETECTION

INTRODUCTION

Researchers [2][17][31][32][39][57][58] have developed efficient video-based fall detection algorithms using the minimum bounding rectangle (MBR) around the moving-region. To improve accuracy, researchers [3][4][49] developed computationally-expensive 3-dimensional projection-based algorithms.

Falls contribute to the injuries of elderly people [48]. Automatically detecting a fall can enable rapid response that in turn can reduce additional complications from a long period in a fallen position. Even when a fall did not result in injury, automatic detection can alert caregivers of the need for preemptive measures such as hazard elimination, physical conditioning, etc.

A fall is characterized by a person beginning with normal behaviors, such as sitting or walking, followed by the person rapidly descending then laying on the ground for an extended period of time. The fall characterization is a sequence of time-ordered events. State machines (SMs) excel at describing time-ordered behaviors; a fact of which this dissertation takes advantage.

The following chapters consist of:

- A description of our moving-region and state machine-based fall detection algorithm and a comparison of that algorithm to the state-of-the-art video-based fall detectors.
- An examination of how to further improve fall detection accuracy by analyzing the efficacy of 3D and 2D head tracking.
- An examination of how fall detection fits into a more complete assistive monitoring environment.
- An analysis of the privacy perceptions of real-people, including evaluating the efficacy of human's to detect falls on various privacy-enhanced video.

1.2 BACKGROUND ON VIDEO-BASED FALL DETECTION

Researchers have developed fall detection algorithms using features of the MBR [2][17][31][32][39][57][58]. Anderson's algorithm [2] used the MBR width to height ratio along with the off-diagonal term from a covariance matrix as features to detect falls. A threshold for each feature determines whether a fall event has occurred. The result of the threshold is the input to a Hidden Markov Model (HMM) that determines whether a fall is likely to have occurred. Anderson did not report the sensitivity or specificity of the fall detector. Miaou's algorithm [39] used a ceiling-mounted camera with a 360 degree view and extracted the person silhouette to produce an MBR. The height to width ratio of the MBR was a feature, and if that ratio exceeded a threshold, then a fall was said to have occurred. Miaou also used the person's body mass index (BMI) to adjust the threshold,

achieving 78% sensitivity and 60% specificity without BMI adjustment and 90% sensitivity and 86% specificity with BMI adjustment. Miaou's study included 20 people and a total of 60 recordings. Williams [58] developed an algorithm using the width to height ratio of the MBR. If the ratio exceeded a threshold then the person was said to be standing, otherwise the person had fallen. Whether the person was sitting or had fallen was the input of a support vector machine (SVM) that determined whether a fall had occurred. Williams evaluated the algorithm with 40 images (not videos) and observed only 1 false positive and 0 false negatives. The sensitivity and specificity were not reported and we were unable to reconstruct them from the results given in the paper. Cucchiara [17] developed an occlusion resistant MBR algorithm using multiple cameras, then input the width to height ratio of the MBR to an HMM to generate a fall likelihood. Cucchiara did not report the sensitivity or specificity of the fall detector. Thome's [57] algorithm computed the angle between the height and width of the MBR to predict the pose of the person, which was passed to a Layered HMM that output the probability that a fall had occurred. Thome evaluated the algorithm with 50 fall videos and 50 walking videos, achieving 82% sensitivity and 98% specificity with 1 camera, and 98% sensitivity and 100% specificity with 2 cameras. Hung's algorithm [31][32] combined two orthogonal cameras by multiplying the width of the MBR from each camera, calling the result the occupied area. The algorithm also determined whether a person was standing, sitting, or laying using the height and occupied areas as features. If the person was laying and the occupied area exceeded a threshold, then a fall may have occurred. If the person remained laying for an extended period of time, then a fall was said to have occurred. The

algorithm was evaluated with the University of Montreal video data set [5]. Hung's algorithm ran in real-time on a desktop computer and achieved a 95.8% sensitivity and 100% specificity using cameras 2 and 5 of the data set.

Rougier developed an algorithm [51] that tracked the human shape by using line detection on the moving region. A Procrustes distance [19] was measured between the observed human shape and a database of normal activity human shapes. The distance was input to a Gaussian Mixture Model (GMM) that output the likelihood of a fall. If the likelihood of a fall exceeded a threshold, then a fall was said to have occurred. Multiple cameras voted to determine whether a fall occurred. A three-fourths majority in favor of a fall was required to determine that a fall had occurred. Rougier evaluated her algorithm with the University of Montreal video data set and achieved with a single camera an area under curve (AUC) between sensitivity and specificity ranging from 0.964 - 0.978. The algorithm achieved with four cameras an AUC of 0.997.Sensitivity and specificity were not provided. In a previous study with the same algorithm, Rougier [51] reported a sensitivity of 0.955 and specificity of 0.964 with a single camera.

Researchers have used features of the multi-dimensional projection of the MBR [3][4][49]. Rougier [49] also developed an algorithm for 3D head tracking to detect falls with a single camera. The algorithm requires the calibration of the camera's characteristics, such as lens deformation effects on the observed world, and the pose of the ground's plane relative to the camera. The head was tracked as an ellipsoid with particle filters using the moving region, color, and body coefficients as features. If the vertical velocity of the head exceeded a threshold, then a fall was said to have occurred.

Rougier evaluated the algorithm with 10 fall recordings (and no non-fall recordings), achieving a 100% sensitivity. Anderson's algorithm [3] extracted person silhouettes from multiple cameras to build a voxel person, then used fuzzy logic to determine whether the person was in an upright state, an on-the-ground state, or in-between states. A fall was declared when the voxel person was in the on-the-ground state for 5 seconds. The algorithm achieved 100% sensitivity and 93.75% specificity with videos of 14 falls and 32 non-falls.

Auvinet's algorithm [4] built a vertical volume distribution based on the silhouette of a person using multiple cameras. The vertical volume would be heavily distributed near the floor when the person was laying on the floor, whereas the vertical volume would be evenly distributed when the person was standing. Auvinet evaluated the algorithm with the University of Montreal video data set and the leave-one-out method, achieving 80.6% sensitivity and 100% specificity with 3 cameras and 100% sensitivity and specificity with 4+ cameras. The results of 2 cameras were not reported. Auvinet also added artificial full occlusions by deleting contributions of one camera resulting in a 94.7% sensitivity and 100% specificity with 3 cameras, and 100% sensitivity and specificity with 4+ cameras. The 3 cameras' result improved with full occlusions over no occlusions because 1/3 of the time the camera that caused the misclassification was excluded. With eight cameras, Auvinet's algorithm ran at 0.34 frames per second (FPS) on a 2.4Ghz Xeon processor, meaning one camera would run at 2.68 FPS. Auvinet accelerated his algorithm with a GPU implementation that improved to 9.5 FPS for eight cameras.

1.3 COMMON TERMS USED IN THIS DISSERTATION

Video-based fall detection algorithms often used the *moving-region* in the video as a predictor for a person. A *minimum bounding rectangle*, or *MBR*, is often fit around the moving-region.

We report accuracy as two numbers: *sensitivity* and *specificity*. Sensitivity is the ratio of correct fall detections over actual falls, e.g. if 23 falls were correctly detected but there were 25 total falls, then sensitivity was 23/25 = 0.92. Specificity is the ratio of correct non-fall reports over actual non-falls, e.g., if 24 non-falls were reported but there were 25 non-falls, then specificity was 24/25 = 0.96.

Chapter 2. ACCURATE AND EFFICIENT CAMERA-BASED FALL DETECTION USING MOVING-REGIONS

2.1 INTRODUCTION TO MOVING-REGION-BASED FALL DETECTION

A fall is characterized by a person beginning with normal behaviors, such as sitting or walking, followed by the person rapidly descending then laying on the ground for an extended period of time. A fall can be described as a time-ordered sequence of behaviors or events. State machines (SMs), widely used in digital circuit design and increasingly in software design, excel at describing time-ordered behaviors. Additionally, SMs promote the capturing of specific, modular behaviors. The modular behaviors allow for a plug-and-play framework for fall detection. Therefore, we used state machines to describe the fall detection method presented in this chapter. In the future work section of this chapter, we discuss expansions on the state machine model.

As shown in Figure 2-1, our method uses multiple fall detectors that contribute to the final decision of whether a fall has occurred, e.g., camera 1 and 2 both feed their own height-based and width-based fall detectors. Each fall detector contributes a fall likelihood score. The method combines the fall likelihood scores to produce a single camera's combined fall score. Multiple cameras can be combined to produce a single multi-camera fall score.

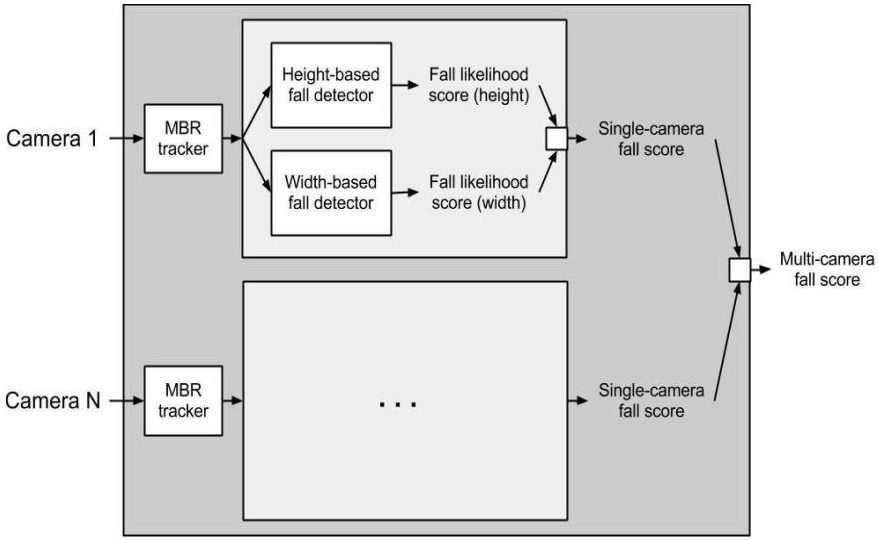


Figure 2-1: Each camera has multiple SM-based fall detectors that produce a fall likelihood score, which can be combined to produce a single-camera fall score, and further combined to a multi-camera fall score.

2.2 RECORDINGS

We used two data sets to evaluate our fall detection algorithm: the University of Montreal (U of M) video data set [5] and our own recordings.

The U of M data set included 24 recordings with 8 cameras each running at 30 FPS with 720x480 pixel resolution. Each recording had a length between 52 seconds and 3 minutes. Figure 2-2 shows a moment in one recording from each camera's perspective.

Figure 2-2: Same moment captured from each camera of the University of Montreal video data set [5]. Camera's numbered 1-8 from left-to-right (first row: 1 2).

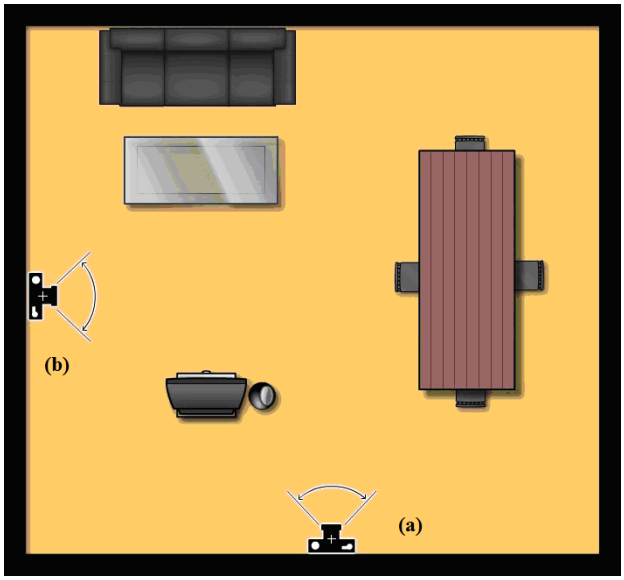


Figure 2-3: Floor plan of our in-home environment including the position and direction of the (a) main camera and (b) supplementary camera. (Floor plan created with http://floorplanner.com)

Our recordings took place at a UC Riverside laboratory in-home environment, shown in Figure 2-3. Two cameras recorded at 15 frames per second at a resolution of 352x240 pixels. The cameras came from a popular in-home 8-camera monitoring set made by Q-See and sold by Costco for around $400. The cameras were positioned and had a viewing direction as shown in Figure 2-3. The cameras were 182cm above the floor.

Our recordings included 1 male actor, age 27, for 69 recordings. The recordings contained 35 falls, including standing, tripping, and slipping falls described in [48], and 34 non-falls, including walking, watching TV, and cleaning. Each recording was one minute long, starting the first 5 seconds without the actor so that the MBR tracking algorithm could learn the background. The falls occurred between 7 and 37 seconds into the video.

Of the 35 fall recordings, we recorded 10 falls with only the main camera. We recorded the other 25 falls with both the main and supplemental camera. Of the 34 non-fall recordings, we recorded 12 non-falls with only the main camera. We recorded the other 24 non-falls with both the main and supplemental camera.

2.3 MBR TRACKER

The MBR tracker performs foregrounding, or foreground-background segmentation, which is the splitting of moving objects in a video's foreground from the video's static background. Typical moving objects in a home are people, especially in the home of a live-alone elderly person being monitored for falls. Many algorithms for accurate foreground-background segmentation exist [36][56][63]. Such algorithms typically detect

non-changing portions of an image as representing the image background, and then subtract the background image from video frames, thus leaving only the moving objects.

We used the established foregrounding algorithm by Zivkovic [63] (implemented in OpenCV 2.2.0 library [43] as BackgroundSubtractorMOG2) to extract a foreground image, shown in Figure 2-4(c). Using the output foreground image, we developed an algorithm [22] to place an MBR around the largest moving object in the foreground image, as shown in Figure 2-4 (d). The MBR tracker is designed to track only one moving object, as would be common in an elderly live-alone home.

(a) **(b)** **(c)** **(d)**

Figure 2-4: Foregrounding via foreground-background segmentation takes as input (a) a background image and (b) a video frame, and outputs (c) a foreground. (d) An MBR is fit around the foreground.

2.4 HEIGHT-BASED FALL DETECTOR

This section describes one of our MBR-based fall detection methods, based on height. The height-based fall detector, shown in Figure 2-5**,** uses the height of the MBR to detect falls with 4 state machines (SMs): suspected fall event, orientation, OK-to-lay, and fall sense. The period of each SM is the same as the frames per second of the recording. The suspected fall event SM uses the height of the MBR to calculate a

suspected fall score that grows proportional to the amount and speed of the increase in the height of the MBR. The orientation SM uses the height-to-width ratio of the MBR to determine whether the person is off-screen, sitting, standing, or laying. The OK-to-lay regions SM is preloaded with a list of regions in the video that are considered safe to lay on, such as a couch. The fall sense SM produces a fall likelihood score based on the maximum suspected fall score and the amount of time the person remains laying in a not safe region.

The training videos used to develop the state machines and thresholds included 9 falls and 5 non-falls from our recordings. The fall recordings included falling from a standing position by tripping, slipping, and loosing consciousness, and backwards, forwards, and rightwards, falling while sitting in a chair, falling while reaching for a lamp from the couch, and falling while cleaning. The non-falls recordings included a couch nap, reading on the couch, walking around the room, cleaning the room, and watching TV.

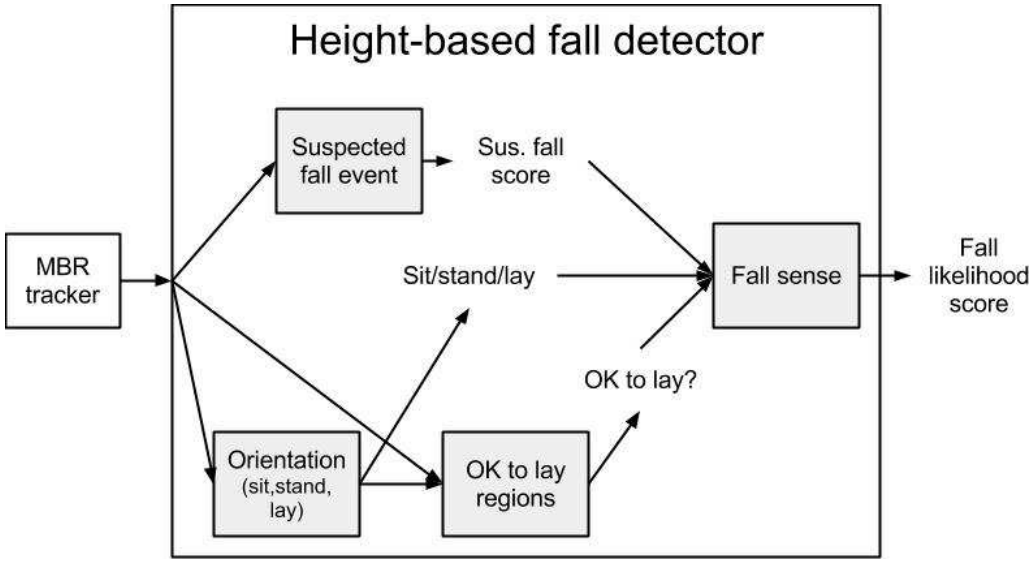


Figure 2-5: Height-based fall detector that includes 4 state machines, uses the MBR from the MBR tracker, and outputs a fall likelihood score.

2.4.1. SUSPECTED FALL EVENT STATE MACHINE

The suspected fall event SM, shown in Figure 2-6, maintains a four second buffer of the height of the MBR and the time of the MBR sampling. The buffer is ordered with the oldest sample at the front and newest sample at the back. The SM initially fills the buffer then transitions to the "check for fall" state.

If the height of MBR element in the buffer is more than 1.0 times the oldest height of MBR in the buffer (indicating a fall), then the SM transitions to the "suspected fall event detected" state. Otherwise, the SM transitions to the "check for fall" state.

The "fill buffer" and "check for fall" states output a suspected fall score of 0. The "suspected fall event detected" state outputs a suspected fall score of 2.

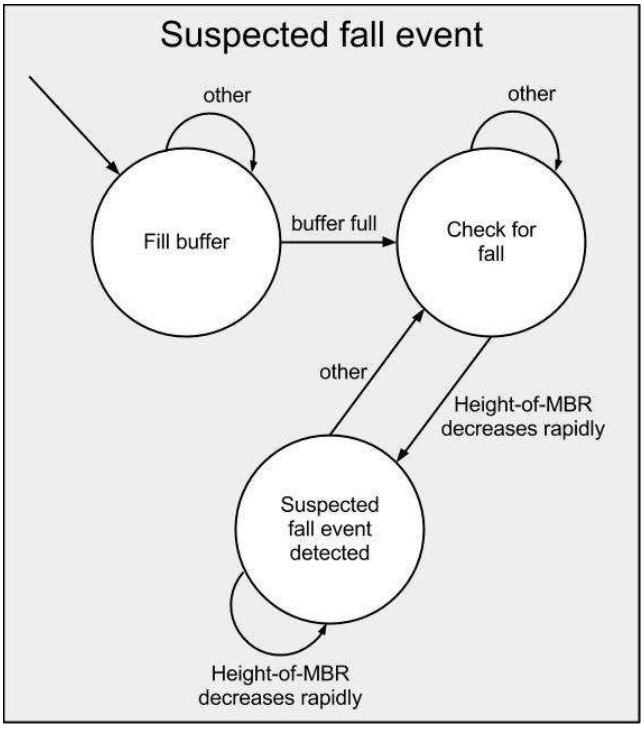


Figure 2-6: Suspected fall event SM checks for falls by looking for large decreases in the height of the MBR.

2.4.2. ORIENTATION STATE MACHINE

The orientation SM uses the height and width of the MBR to output whether the person is in one of four states: off-screen, standing, sitting, or laying. The person is said to be off-screen if the width or height of the MBR is 0 pixels. Otherwise, if the height-to-width ratio exceeds 1.75, the person is said to be standing. Otherwise, if the height-to-width ratio exceeds 0.9, the person is said to be sitting. Otherwise, the person is said to be laying. The height-to-width ratio thresholds of 1.75 and 0.9 were determined by observing the actor's ratios in the training recordings when the actor was standing, sitting, and laying.

2.4.3. OK-TO-LAY REGION STATE MACHINE

The OK-to-lay region SM determines whether the person is laying in a safe place, such as a couch, bed, or chair. The SM contains a list of regions that are safe to lay down. The list was created during setup by manually adding a rectangular box around the couch in our recordings.

The OK-to-lay region SM has three states: not-laying, safe-laying, and not-safe-laying. The SM takes as input the MBR and the output of the orientation SM. If the orientation SM outputs that the person is not laying, then the OK-to-lay region SM transitions to the not-laying state. Otherwise, if one of the OK-to-lay regions contains 90% of the MBR, then the SM transitions to the safe-laying state. Otherwise, the SM transitions to the not-safe-laying state.

The OK-to-lay region SM outputs true only when in the safe-laying state. Otherwise, the SM outputs false.

2.4.4. FALL SENSE STATE MACHINE

A fall includes a suspected fall event followed by a period of laying down in an unsafe region. The fall sense SM, shown in Figure 2-7, takes input from the suspected fall event SM, orientation SM, and OK-to-lay regions SM. The fall sense SM outputs a fall likelihood score as a positive value or 0. The larger the score, the more likely that a fall occurred. A score of 0 means that a fall did not occur.

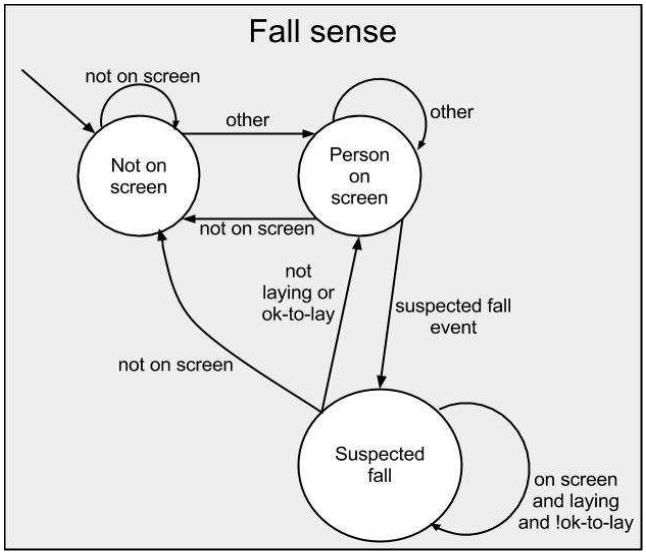


Figure 2-7: Fall sense SM outputs a fall likelihood score based on the maximum suspected fall score since transitioning to the "suspected fall" state and the length of time the person remains laying down in a not ok-to-lay region.

The fall sense SM initializes in the "not on screen" state. If the orientation SM is not in the "not on screen" state, then the fall sense SM transitions to the "person on screen" state. Otherwise, the fall sense SM transitions to the "not on screen" state.

If the orientation SM is in the "not on screen" state, then the "person on screen" state transitions to the "not on screen" state. Otherwise, if the suspected fall score exceeds 0, then the "person on screen" state transitions to the "suspected fall" state. Otherwise, the "person on screen" state transitions to itself.

The "suspected fall" state transitions to "not on screen" state if the orientation SM is in the "not on screen state". Otherwise, if the orientation SM is not in the "laying" state or the OK-to-lay region SM outputs true, then the "suspected fall" state transitions to the "person on screen". Otherwise, the "suspected fall" state transitions to itself.

The fall likelihood score is 0 when in the "not on screen" and "person on screen" states. The "suspected fall" state maintains a maximum-seen suspected fall event score that is set to 0 when the "suspected fall" state is left. The maximum-seen score divided by one-fifth the sample rate of MBR is added to the fall likelihood score once per period. A fall likelihood score can reach 100 in no faster than 2 seconds.

2.5 HEIGHT AND WIDTH-BASED FALL DETECTORS

This section describes an improvement to our method that uses both height and width. A weakness of height-based fall detection occurs when the fall occurs from the sitting position because the height does not decrease rapidly. However, the width may increase rapidly.

As shown in Figure 2-8, we added width-based fall detection to the height-based fall detection (shown in Figure 2-5). The orientation and OK-to-lay regions SMs used in the width-based fall detector are identical to the SMs used in the height-based fall detector. The width-based fall detector has modified suspected fall event and fall sense SMs. Lastly, the fall likelihood scores for height and width are averaged to produce a single-camera fall score.

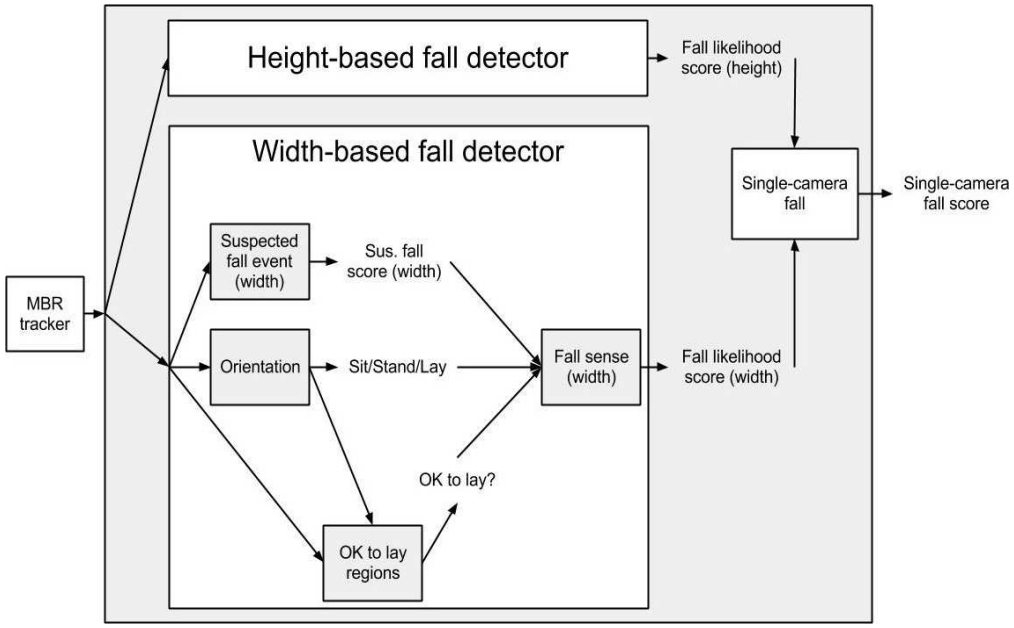


Figure 2-8: Height and width-based fall detectors for a single camera. The fall likelihood for height and width are averaged to produce a single-camera fall score.

2.5.1. SUSPECTED FALL EVENT STATE MACHINE (WIDTH)

The suspected fall event SM (width) has the same definition as the suspected fall event SM (height) described in Section 0, except the buffer contains width of MBR instead of height of MBR, and the threshold values are tuned for width-based fall detection.

If the newest width of MBR in the buffer is more than 1.5 times the oldest width of MBR in the buffer (indicating a fall), then the SM transitions to the "fall event detected" state. Otherwise, the SM transitions to the "check for fall" state. The "fill buffer" and "check for fall" states output a suspected fall score of 0. The "fall event detected" state outputs a suspected fall score of 2. A suspected fall score (width) of 2 was used, instead of up to 10 like with the suspected fall score (height), because the suspected fall event

SM (width) often confuses bending down or leaning over with a fall. The lower value causes a slower growth potential (up to 5 times slower) of the fall likelihood score (width) than the fall likelihood score (height).

2.5.2. FALL SENSE STATE MACHINE (WIDTH)

The fall sense SM (width) has the same definition as the fall sense SM (height), shown in Figure 7, except that fall sense SM (width) replaces input from suspected fall score (height) with input from suspected fall score (width). Also, the output is fall likelihood score (width).

2.5.3. SINGLE-CAMERA FALL STATE MACHINE

The single-camera fall SM averages the fall likelihood scores from the height and width-based fall sense SMs. The effect of averaging is that when the fall detectors disagree, then twice the length of time must pass before a fall can be declared. For example, we may assume that a single-camera fall score of 100 is needed to declare that a fall has occurred. If only one of the fall sense SMs is outputting a fall likelihood score above 0, then the single-camera fall score will require twice the length of time to declare a fall to have occurred. However, if both the height and width fall sense SMs are outputting fall likelihood scores above 0, then the single-camera fall score will reach 100 in about the same amount of time that either fall likelihood score would reach 100.

2.5.4. EXTENDING TO MULTI-CAMERA FALL DETECTION

The multi-camera fall detection, shown in Figure 2-9, averages the outputs of the single camera fall detection, which could be the single-camera fall score described in Section 2.5.3 or the fall likelihood score described in Section 2.4. The multi-camera fall SM averages the single-camera fall score for cameras with an MBR (indicating that a person is on screen). The average is output as the multi-camera fall score.

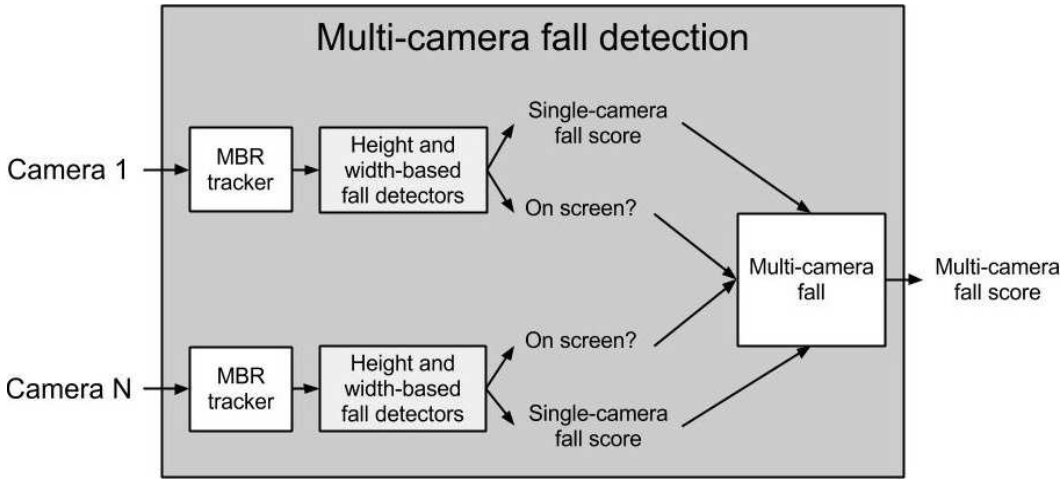


Figure 2-9: Each camera contributes a single-camera fall score. The single-camera fall scores of cameras with an MBR (indicating person on screen) are averaged to produce a multi-camera fall score.

2.6 EVALUATION OF OUR FALL DETECTORS

2.6.1. FALL DETECTION ACCURACY

We evaluated the accuracy of our fall detection algorithm on the recordings on the University of Montreal video data set [5].

We required our algorithm's multi-camera fall score to meet or exceed 100 to declare that a fall had occurred. If only a single camera was used, then the multi-camera fall score reflected that camera's single-camera fall score.

Hung's [32], Auvinet's [4], and Rougier's [52] fall detection algorithms were evaluated on the University of Montreal video data set. All three algorithms required the person to remain on the floor for at least 5 seconds after the fall in order to determine that a fall had occurred. The problem is that the actor did not remain on the floor for at least 5 seconds after some falls. The researcher's solution was to extend the last frame until 5 seconds after the fall. We employed the same technique extending the last frame until our fall detector decided whether a fall had occurred. The recording ending within 5 seconds of the fall does not allow for additional confounding events, such as the person getting back up or moving on the ground. However, the data set was sufficient to get a sense of our fall detection algorithm's performance and to compare to other fall detection algorithms on a common data set.

We excluded cameras 4 and 5 from the evaluation because the MBR of the person standing and laying are nearly the same. These two cameras were on opposite sides of the room, viewing the long direction of the room, which was 9.0m. The actor typically fell toward/away from these cameras. Future work is to improve our orientation SM to better handle falls toward/away from the camera, perhaps by using information provided by each camera.

We excluded recordings 23 and 24, as did Auvinet's experiments [4], because those videos had other moving objects that cause the MBR tracker to fail to track the person. In

these recordings, the actor moved various inanimate objects, including boxes, a jacket, brooms, and trash bins, which confused the MBR tracker to the extent that nearly the entire room was included in the MBR. Future work is to improve our MBR tracker (more details in Section 0). To see if excluding such inanimate objects would improve our method, we performed human-assisted MBR extraction for the chair fall recordings from our recordings, which then yielded perfect fall detection for height and width fall detectors.

We developed the state machines and thresholds described in Sections 2.4 and 2.5 using our video recordings. To account for changes in camera position and angle between our recordings and the University of Montreal's recordings, we re-trained the orientation SM's sit-lay threshold using recording number 5 (chosen arbitrarily) for each camera of the University of Montreal video data set. The method of training was to select the smallest threshold (to the tenths place) that yielded perfect sensitivity and specificity. We tested on the remaining 21 recordings of the University of Montreal data set. For multi-cameras, we evaluated all combinations of cameras.

The OK-to-lay SMs were not used during the evaluation. We would expect adding the OK-to-lay SMs to improve the specificity in scenarios in which the actor sits or lays on a couch but is determined to have fallen.

# of cameras	Our method	Hung [32]	Auvinet [4]	Rougier [51]	Anderson [3]	Miaou [39]	Thome [57]
1	**0.960**	-	-	0.955	-	0.900	0.820
2	0.990	0.958	-	-	**1.000**	-	0.980
3	**0.998**	-	0.806	-	-	-	-
4	**1.000**	-	0.997	-	-	-	-
5	**1.000**	-	0.999	-	-	-	-
6	**1.000**	-	**1.000**	-	-	-	-
7	-	-	**1.000**	-	-	-	-
8	-	-	**1.000**	-	-	-	-

Table 2-1: Fall detection *sensitivity* comparing our method to the state-of-the-art video-based fall detection algorithms. A dash (-) means unreported or not applicable, such as Hung's algorithm that uses exactly two cameras.

# of cameras	Our method	Hung [32]	Auvinet [4]	Rougier [51]	Anderson [3]	Miaou [39]	Thome [57]
1	**0.995**	-	-	0.964	-	0.860	0.980
2	**1.000**	**1.000**	-	-	0.938	-	**1.000**
3	**1.000**	-	**1.000**	-	-	-	-
4	0.995	-	**0.998**	-	-	-	-
5	0.993	-	**1.000**	-	-	-	-
6	**1.000**	-	**1.000**	-	-	-	-
7	-	-	**1.000**	-	-	-	-
8	-	-	**1.000**	-	-	-	-

Table 2-2: Fall detection *specificity* comparing our method to the state-of-the-art video-based fall detection algorithms. A dash (-) means unreported or not applicable, such as Hung's algorithm that uses exactly two cameras.

Hung's algorithm [32] required two orthogonal cameras and used camera 2 and camera 5 of the University of Montreal video data set. Auvinet's algorithm [4] required 3 or more cameras. Anderson's algorithm [3] required two cameras. Anderson [3], Miaou [39], and Thome [57] did not evaluate using the University of Montreal video data set, instead each used his own recordings.

Table 2-1 shows the fall detection sensitivity, comparing our algorithm to the state-of-the-art video-based fall detection algorithms. Our algorithm outperformed the other algorithms for one, three, four, and five cameras. For two cameras, Anderson's algorithm

achieved perfect 1.000 sensitivity, while ours achieved 0.990. For six cameras, both our and Auvinet's algorithm achieved 1.000 sensitivity.

Table 2-2 shows the fall detection specificity, comparing our algorithm to the state-of-the-art fall detection algorithms. For a single camera, our algorithm outperformed Miaou's and Thome's algorithms. For two cameras, our algorithm, Hung's, and Thome's algorithm achieved a perfect 1.000 specificity. Our method matched Auvinet's method for 3 and 6 cameras with perfect specificity but performed worse by less than 0.008 for 4 and 5 cameras.

The performance of each algorithm may be due in part to that algorithm's coverage of a fall's typical time-ordered behaviors, including suspecting a fall, waiting to see whether the person is OK or got up, then after an extended period of time notifying the caregiver. As shown in Table 2-3, the state-of-the-art methods are missing one or more of the time-ordered behaviors, whereas our method includes each of the time-ordered behaviors.

We also evaluated the accuracy of our fall detection algorithm on our recordings described in Section 2.2. We developed the state machines described in Sections 2.4 using the training data described in the recording's section. The accuracy was 1.000 for sensitivity and specificity when using either 1 or 2 cameras.

Time-ordered behaviors	**Our method**	**Hung [32]**	**Auvinet [4]**	**Rougier [51]**	**Anderson [3]**	**Miaou [39]**	**Thome [57]**
Suspected fall	✓			✓			
Orientation	✓	✓	✓	✓	✓	✓	✓
Fall sense	✓	✓	✓		✓		✓

Table 2-3: Time-ordered behaviors by fall detection algorithm. Our method includes each time-ordered behavior, whereas the state-of-the-art methods are missing one or more behavior.

2.6.2. FALL DETECTION EFFICIENCY

We evaluated computational efficiency of our MBR tracker by determining the processing rate in frames per second (FPS) with the University of Montreal video data set [5], which recorded at 30 FPS. The evaluation included camera 1 and all 24 recordings. We ran the evaluation 10 times, calculating the mean and standard deviation of FPS. The computer used during evaluation had a 3.30 GHz Intel Xeon processor with 8GB of memory and ran 64-bit Windows 7. Our MBR tracker was run single-threaded and achieved an average of 32.1 FPS with 0.1 FPS standard deviation, which is comparable to other MBR-based fall detection algorithms, such as Hung's algorithm running in real-time [32].

Recently, Rougier has shown that 3D head tracking, shown in Figure 1(a), can be used to predict falls [49][51], but at the higher CPU cost of 7.7 frames per second (FPS) [49]. Also, Auvinet's 3D human tracking algorithm ran at 2.7 FPS per camera with a 2.4Ghz Xeon processor. Since Avuinet's algorithm was run on a lower frequency processor than ours, we can somewhat compensate his performance by multiplying his evaluated FPS by (3.3/2.4), or 1.375, to yield an adjusted FPS of 3.7 FPS. Although the adjusted FPS is an estimate of projection-based fall detection efficiency, the projection-based efficiency is still about 10x slower than MBR-based fall detection efficiency.

More than 99.9% of our run-time was spent on just MBR tracking. In contrast, Auvinet's algorithm [4] spent more time computing projection than MBR tracking as evidenced by an increase from 0.88 FPS, when both MBR tracking and projection were

run on the CPU, to 10.2 FPS, when the projection was moved to a GPU implementation but MBR tracking remained on CPU.

2.6.3. TRADE-OFF BETWEEN FALL DETECTION ACCURACY AND EFFICIENCY

We analyzed the trade-off between fall detection accuracy, using a combined accuracy score, and efficiency, using frames per second processing rate, on our method and the state-of-the-art methods. The combined accuracy score is the sensitivity multiplied by the specificity.

We assume the efficiency of the other MBR-based methods by Hung [32], Miaou [39], and Thome [57] were equivalent to our method because those methods computationally were not significantly different from ours, and the MBR tracking was 99.9% of our run-time, as discussed in Section 2.6.2. Auvinet's method [4] achieves approximately 3.7 FPS per camera as discussed in Section 2.6.2. Rougier [51] stated that her algorithm "can run in real time at 5 frames/s", but likely executes closer to 15 FPS because her method uses MBR tracking along with histogram analysis and shape analysis, in which MBR tracking is likely as computationally expensive as histogram analysis plus shape analysis. Anderson evaluated his method [3] using 3 FPS video but does not discuss computational expenses. Anderson's algorithm included MBR tracking and voxel intersection, which were likely the most computationally expensive aspects of his fall detection algorithm, with MBR tracking being twice as expensive as voxel

intersection; therefore, we assume Anderson's algorithm would execute at 22.5 FPS per camera.

Figure 2-10 shows the trade-off between the combined accuracy score and efficiency when a single camera is used. The trade-off included all state-of-the-art methods that with reported results for single camera fall detection accuracy. Figure 2-11 shows for two camera used, Figure 2-12 for 3 camera use, and Figure 2-13 for four camera use. For both metrics of accuracy and efficiency, a higher score is better, thus being closer to the top-right of the graph is better. Our method was closest to the top-right in each figure.

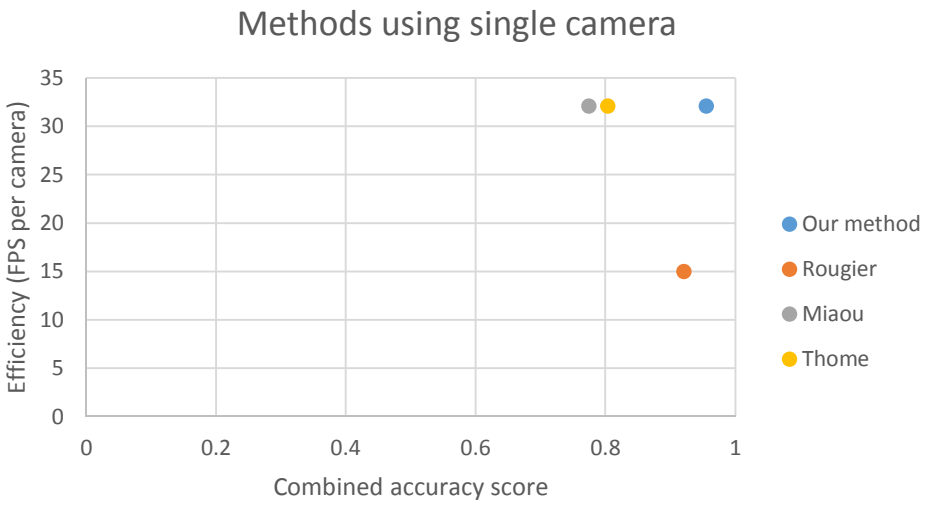


Figure 2-10: The accuracy and efficiency trade-off between method's with reported results for performing fall detection using a single camera. Higher is better; close to top-right is best; therefore, our method is best for single camera use.

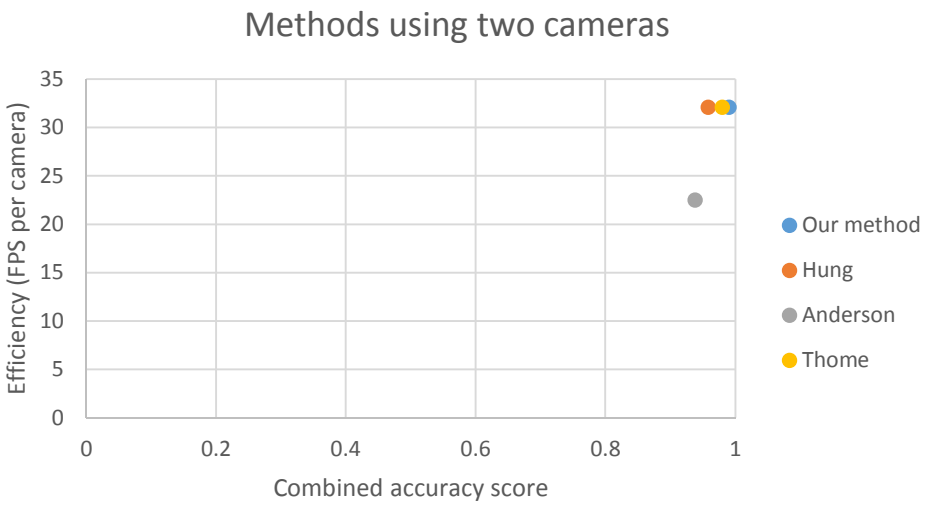


Figure 2-11: Two camera trade-off between accuracy and efficiency. Higher is better; close to top-right is best; therefore, our method is best for two camera use.

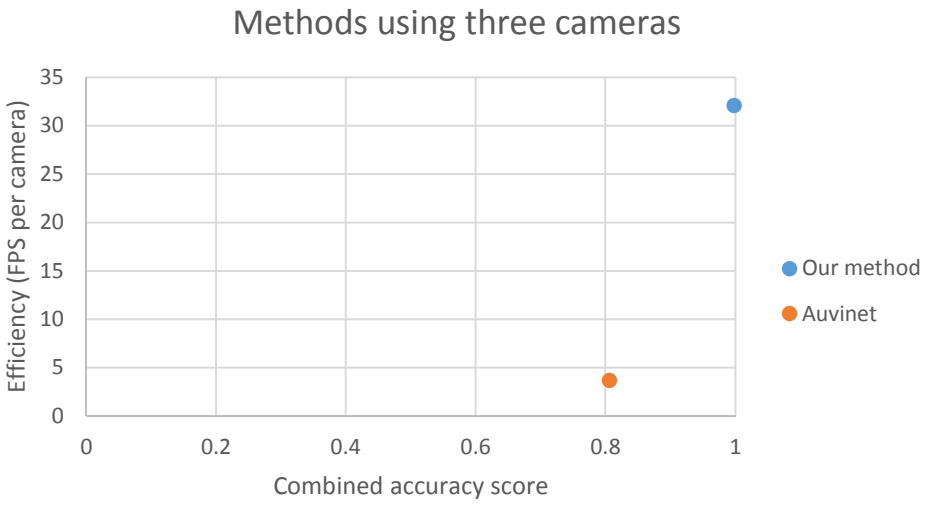


Figure 2-12: Three camera trade-off between accuracy and efficiency. Higher is better; close to top-right is best; therefore, our method is best for three camera use.

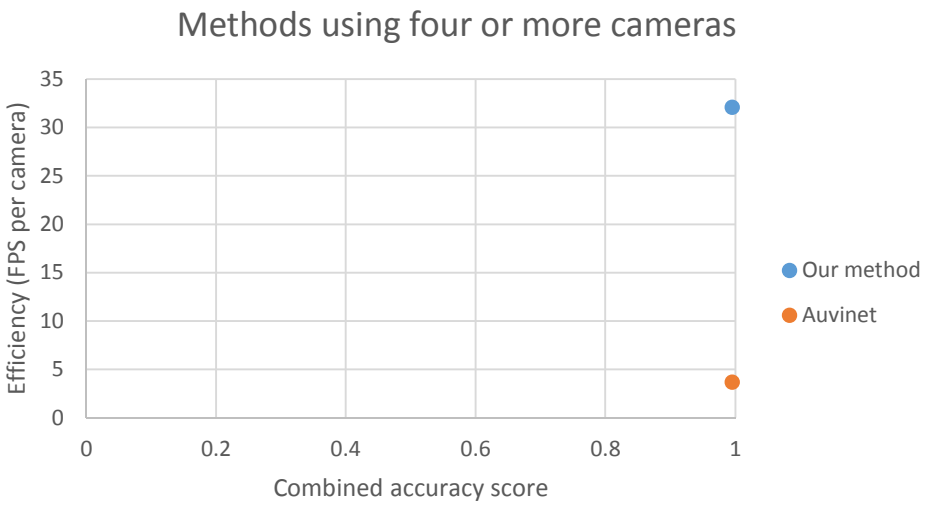


Figure 2-13: Four or more camera trade-off between accuracy and efficiency. Higher is better; close to top-right is best; therefore, our method is best for four camera use.

2.7 FUTURE WORK

We plan to develop a video data set that includes post-fall confounding events, such as moving on the floor and getting back up. Additionally, we plan to include events that do not include falls that appear to be falls, such as checking under the couch.

The MBR tracker we used was designed to track only one moving object and gets confused when multiple moving objects are on screen, such as a person and a chair. We plan to find or develop an MBR tracker that can discriminate between people and inanimate objects. Even better would be to find or develop an MBR tracker that only tracks an intended person.

Each suspected fall event SM's fall likelihood score could be based on the severity of the fall in terms of length of fall and distance fallen. More severe falls would generate

larger fall likelihood scores, reducing the duration to wait between suspected fall and declaring a fall had occurred.

The threshold values for the suspected fall event and orientation SMs could be learned based on the camera angle and position, perhaps with a person initially walking onto screen to give an example of sit, stand, and lay. Additionally, different body types may require different threshold values, so a method for modifying the thresholds to personalize should be investigated.

One such method for automatically training thresholds is to convert the orientation and suspected fall event SMs into HMMs. The advantage would be a statistical model that can be trained with ground truth data for each HMM.

The Ok-to-lay region list could be improved by automatically recognizing safe to lay regions using object recognition software. Furthermore, the list could be dynamically updated to account for room re-arranging.

Cameras with programmable processors are commercially available [6]. The non-MBR tracking state machines could be executed on a single, low-power microcontroller. The MBR tracking algorithm contains many parallelizable calculations, particularly in loops scanning over each pixel in an image, for example, Auvinet [4] accelerated his MBR tracking algorithm with a GPU, achieving an average speed-up of 29.3%.

2.8 CONCLUSION ON ACCURACY AND EFFICIENCY OF MOVING-REGION-BASED FALL DETECTION

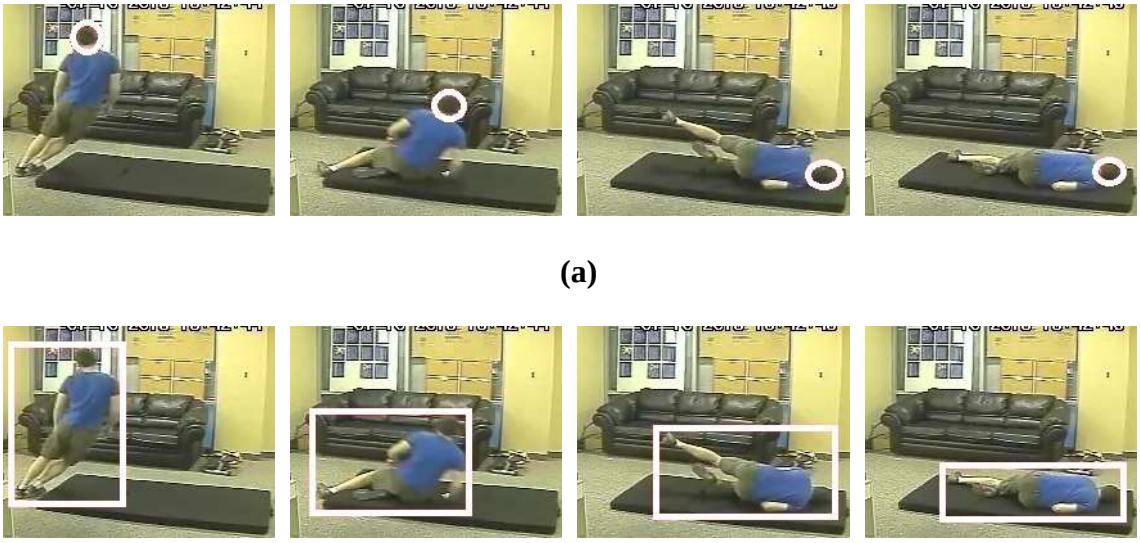
Our fall detection algorithm was more accurate than other MBR-based algorithms, while being equally efficient. Also, our algorithm was an order of magnitude more efficient than projection-based algorithms, while being more accurate with 3 cameras and equally accurate with 4+ cameras. We evaluated our algorithm using the commonly used video data set from Univ. of Montreal. A single-camera achieved 0.960 sensitivity and 0.995 specificity, while 2 or more cameras achieved 0.990 or better sensitivity and specificity. Our algorithm ran at 32.08 FPS for one camera while single-threaded on a 3.30 GHz Xeon processor.

Chapter 3. VIDEO-BASED FALL DETECTION EFFICACY: 3D HEAD TRACKING, 2D HEAD TRACKING, AND MOVING-REGION TRACKING

3.1 INTRODUCTION TO HEAD TRACKING AND MOVING-REGION FALL DETECTION

Video-based fall detection algorithms typically use the MBR around the moving-region, shown in Figure 1(b), to predict falls [1][17][31][39][52][57][58]. Recently, Rougier has shown that 3D head tracking, shown in Figure 1(a), can be used to predict falls [49][51], but at the higher frames per second (FPS) of 7.7 [49], as opposed to 32.1 FPS with MBR (see Section 3.5.4).

This chapter compares the fall detection efficacy of MBR, 2D head, and 3D head tracking. Figure 3-1 shows the same picture sequence of head tracking and MBR tracking. This chapter's experiments include using manual 2D and 3D head-tracking on 87 recordings, including confounding and non-confounding, fall and non-fall scenarios to determine the improved fall detection accuracy that head tracking's extra computational cost might provide.



(a)

(b)

Figure 3-1: Picture sequence showing fall with (a) head tracking and the same fall with (b) moving-region tracking.

3.2 RECORDINGS

Our recordings [25] took place at a UC Riverside laboratory in-home environment, and include video, 2D head tracking, 3D head tracking, and moving-region tracking data. A camera recorded at 15 frames per second at a resolution of 352x240 pixels. The cameras are shown in Figure 2-3. The cameras came from a popular in-home 8-camera monitoring set made by Q-See and sold by Costco for around $400. The cameras were positioned and had a viewing direction as shown in Figure 3-1. The cameras were 182cm above the floor.

Each recording had a length of one minute, starting the first 5 seconds without the actor so that the MBR tracking algorithm could learn the background. The recordings

included 1 male actor, age 27. The falls occurred between 7 and 37 seconds into the recording.

We collected 87 recordings. 18 recordings contained confounding scenarios (5 fall, 13 non-fall), while 69 contained non-confounding scenarios (35 fall, 34 non-fall). A confounding scenario was typically harder for the fall detectors to correctly classify the recording, such as looking under the couch for a dropped item or falling next to an upright vacuum that was just used. Non-confounding scenarios include standing falls, tripping falls, and slipping falls described in Rantz's paper [46], as well as walking, watching TV, and cleaning.

Fourteen non-confounding recordings (9 falls and 5 non-falls) were used for training the state machines and thresholds described in Section 3.4. The remaining non-confounding and all confounding recordings were used for testing in this chapter.

3.3 HEAD AND MOVING-REGION TRACKING

We developed manual 2D head tracking software in Python. The software would display a frame, wait for the user to click on the frame, then display the next frame. The location of the click (as a pixel value) on the frame was stored, along with the frame's time position in the recording. The software displayed 15 frames per 1 second of recording. The manual head tracking was performed by a research assistant who was instructed to continuously click on the center of the actor's head.

Similarly, we developed manual 3D head tracking software in Python. The software would display a frame from the video along with 10 vertical head height options and a

no-actor present option. Each vertical head height option was a picture of the actor, ranging from the actor's head being 6 inches (actor laying on the ground) to 80.25 inches (actor jumping upward) from the ground. When a vertical head height or no-actor present option was selected, the software would show the next frame. The manual vertical head height tracking was performed by a research assistant who was instructed to select the option that best matched the current frame.

The MBR tracker is discussed in Section 2.3.

3.4 HEAD-BASED FALL DETECTOR

The ***2D head-based fall detector*** uses the pixel-value of the vertical position of the head to detect falls with 4 state machines (SMs): suspected fall event, orientation, OK-to-lay, and fall sense. The 2d head-based fall detector consists of the same SMs described in Section 2.4, except that the suspected fall event SM uses the vertical position of the head to calculate a suspected fall score that grows proportional to the amount and speed of the increase in the vertical position of the head. Head tracking commonly uses the moving-region as a feature [49][51], so we use the MBR as part of 2D head-based fall detection.

The state machines and thresholds were trained with the 14 non-confounding recordings described in Section 3.2. The fall recordings included falling from a standing position by tripping, slipping, and loosing consciousness, and backwards, forwards, and rightwards, falling while sitting in a chair, falling while reaching for a lamp from the couch, and falling while cleaning. The non-falls recordings included a couch nap, reading on the couch, walking around the room, cleaning the room, and watching TV.

The suspected fall event SM, similar to Figure 2-6, differs in that a fall is suspected when the newest element in the buffer is greater than or equal to 100 (indicating the head is near the floor). The value of 100 indicating that the head is near the floor was decided by choosing the vertical position in the frame that was approximately 1/3 of a meter above the floor where the floor meets the wall in the far side of the room. The "fill buffer" and "check for fall" states output a suspected fall score of 0. The "suspected fall event detected" state outputs a suspected fall score of 2.

3.4.1. 3D HEAD-BASED FALL DETECTOR

The ***3D-based fall detector*** takes input from the 3D vertical head height tracker, and contains the same 4 state machines as the 2D head-based fall detector, with modifications to the suspected fall event SM. The suspected fall event SM's buffer stores the vertical head height in inches. The newest vertical head height value in the buffer must be less than 15 inches, also the newest and oldest vertical head heights in the buffer must not have the no-actor present option (meaning, the actor must be present on the screen).

3.4.2. MOVING-REGION-BASED FALL DETECTORS

The top of the MBR can be considered an approximation of head tracking because the head of the person tends to be at the top of the MBR. The ***MBR-top-based fall detector*** takes input from the MBR tracker, instead of the head tracker, and contains the same 4 state machines as the 2D head-based fall detector, with modifications to the suspected fall event SM. The suspected fall event SM's buffer stores the top of the MBR

values. The newest top of MBR value in the buffer must exceed 1.3 times the value of the oldest.

The ***MBR-height-based fall detector*** takes input from the MBR tracker, instead of the head tracker, and contains the same 4 state machines as the 2D head-based fall detector, with modifications to the suspected fall event SM. The suspected fall event SM's buffer stores the height of the MBR values. The newest height of MBR value in the buffer must exceed 0.9 times the value of the oldest.

The ***MBR-width-based fall detector*** takes input from the MBR tracker, instead of the head tracker, and contains the same 4 state machines as the 2D head-based fall detector, with modifications to the suspected fall event SM. The suspected fall event SM's buffer stores the width of the MBR values. The newest width of MBR value in the buffer must exceed the value of the oldest.

3.5 EVALUATION OF HEAD AND MOVING-REGION FALL DETECTORS

We evaluated the accuracy of head and MBR-based fall detectors. We used the state machines and thresholds described in Section 5 that were trained with our training recordings.

If only a single fall detector was used, then we required that fall detector's fall likelihood score to meet or exceed 100 to be declared a fall. If one camera with multiple fall detectors were used, then we required that fall camera's single-camera fall score to

meet or exceed 100. If multiple cameras were used, then the multi-camera fall score was required to meet or exceed 100.

3.5.1. FALL DETECTION ACCURACY ON NON-CONFOUNDING SCENARIOS

We evaluated the accuracy of 2D head, 3D head, and MBR fall detectors with our recordings of non-confounding scenarios. The non-confounding scenarios included standing falls, tripping falls, and slipping falls described in Rantz's paper [16] (which included the suggestions from an experienced nurse in elder-person care), as well as walking, watching TV, and cleaning.

The 2D head, 3D head, and MBR-based fall detectors each achieved 1.00 sensitivity and 1.00 specificity. Thus indicating that the MBR-based fall detectors are sufficient for fall detection, as well as, 3D head and 2D head fall detectors.

3.5.2. FALL DETECTION ACCURACY ON CONFOUNDING SCENARIOS

The confounding scenarios are intended to find the pathological differences between the classification accuracy between the 3D head, 2D head, and MBR-based fall detectors. The confounding scenarios are deliberately atypical scenarios in the attempt to find pathological differences.

Table 3-1 shows the comparison between head-based and MBR-based fall detector accuracy with our confounding scenario recordings. 3D head, 2D head, and MBR-based

fall detectors achieved the same sensitivity of 0.80, whereas the 3D head-based fall detector had a 1.00 specificity, 2D head-based had a 0.85, and the MBR-based has a 0.54. The difference in specificity between 3D head, 2D head and MBR-based fall detectors was caused by the head-based fall detector knowing that the head was not near the ground.

Of the fall recordings, all fall detectors failed to correctly classify "fall w/ vacuum 2" as a fall because the orientation SM declared the person to be sitting when the person was actually laying. The MBR tracker contained the fallen person and upright vacuum, which gave an orientation that appears to be a person sitting. Although the respective suspected fall event SM for 2D and 3D head-based fall detectors suspected a fall, the respective fall sense SM decided a fall had not occurred because the orientation SM declared that the person was not laying. The same result was observed for top, height, and width of MBR.

The 3D head-based fall detector accurately classified all non-fall recordings. 2D head and the MBR-based fall detectors misclassified "crouch with box" and "kneel and move chair". The "crouch with box" recording is particularly difficult because the moving-region includes a large cardboard box and the actor is kneeling while looking in the box, hence the orientation SM outputs "laying". Also, the person's head is close to the ground. For similar reasons, "kneel and move chair" was misclassified.

The 2D head detector correctly classified the remaining non-fall recordings, whereas the MBR-based fall detectors also misclassified "sit quickly" and "sit then toss up item". In both recordings, the orientation SM declared the actor to be laying. The head-based fall detectors did not misclassify these recordings because the head position/height was

not near the ground. If the top-based fall detector knew the head was not on the ground, then the top and 2D head-based fall detectors would have achieved the same accuracy.

The top-based detector misclassified "hands up, down, then lay" because the actor's arms horizontally extended caused the orientation SM to declare the actor to be laying and the top of the MBR decreased by more than 1.3 times.

The height and width-based detectors misclassified "sit then hands to side" because the orientation SM declared the MBR to be laying when the actor was sitting with his arm's stretched horizontally. The top-based detector did not suspect a fall because the top of the MBR did not decrease by 1.3 times.

Although the confounding scenarios consisted of atypical situations in the daily lives of elderly persons, the scenarios helped identify pathological differences between the 3D head, 2D head, and MBR-based fall detectors. Specifically, the 3D head and 2D head-based fall detectors had the advantage of knowing how far the head was from the ground, which enabled a few non-fall scenarios to be correctly classified, whereas the MBR-based fall detectors misclassified.

Confounding scenarios - falls					
Confounding scenario	*3D Head*	*2D Head*	*MBR top*	*MBR height*	*MBR width*
Fall w/ vacuum 1	✓	✓	✓	✓	✓
Fall w/ vacuum 2					
Put book in shelf	✓	✓	✓	✓	✓
Look under couch	✓	✓	✓	✓	✓
Take picture off wall	✓	✓	✓	✓	✓
Confounding scenarios - non-falls					
Confounding scenario	*3D Head*	*2D Head*	*MBR top*	*MBR height*	*MBR width*
Crouch with box	✓				
Set cushion on couch	✓	✓	✓	✓	✓
Sit quickly	✓	✓			
Hands to side then sit	✓	✓	✓	✓	✓
Sit then hands to side	✓	✓	✓		
Hands up, down, then lay	✓	✓		✓	✓
Hands up, down, then sit 1	✓	✓	✓	✓	✓
Hands up, down, then sit 2	✓	✓	✓	✓	✓
Sit then hands up/down	✓	✓	✓	✓	✓
Lay then toss up item	✓	✓	✓	✓	✓
Sit then toss up item	✓	✓			
Stand then toss up item	✓	✓	✓	✓	✓
Kneel and move chair	✓				
Confounding scenarios - accuracy summary					
Sensitivity	**0.80**	0.80	0.80	0.80	0.80
Specificity	**1.00**	0.85	0.54	0.54	0.54

Table 3-1: Confounding scenarios comparison of 2D head, 3D head, top, height, and width-based fall detector accuracy. ✓ means correct.

3.5.3. ATTEMPTED IMPROVEMENTS OF MOVING-REGION-BASED FALL DETECTION

We evaluated whether the top of MBR could be used to determine that the person was not on the ground by modifying the top of MBR's suspected fall event SM to additional require that the newest element in the buffer equal or exceed 100 (indicating the top of MBR is on the floor), then re-evaluated with the non-confounding and confounding scenarios. The top of MBR's fall detector still achieved perfect sensitivity and specificity of 1.00 for the non-confounding scenarios. For the confounding scenarios, the specificity increased to a perfect 1.00, whereas the sensitivity decreased to 0.20. The

same modification yielded the same results for the height and width-based MBR fall detectors. The cause of the change with the confounding scenarios accuracy is that the MBR would encompass a non-person object, such as the top of a vacuum or chair, which was not near the floor. The trade-off between sensitivity and specificity means fewer false alarms at the expense of more missed falls, which may be beneficial for practical applications, in which very few false alarms are acceptable.

We evaluated whether including a supplemental camera positioned with an orthogonal view of the room improved the MBR-based fall detectors with the confounding scenarios. A subset of the recordings described in Section 3 also had supplemental camera recordings and those recordings were used for this evaluation, including 40 non-confounding scenario recordings (18 fall and 22 non-fall) and all 18 confounding scenario recordings (5 fall and 13 non-fall).

Of the non-confounding scenarios, the top-based fall detector achieved the same perfect sensitivity and specificity of 1.00. Whereas the width-based and height-based fall detectors achieved the same 1.00 sensitivity, but achieved the lower 0.91 specificity and 0.95 specificity, respectively.

Of the confounding scenarios, the top-based, height-based, and width-based fall detectors achieved the higher 1.00 sensitivity but the lower specificity of 0.23, 0.31, and 0.31, respectively. The sensitivity improved because the supplemental camera correctly classified "fall w/ vacuum 2" as a fall. The supplemental camera's orientation SM determine the actor to be laying in "fall w/ vacuum 2", whereas the main camera's orientation SM determined the actor to be sitting. The specificity decreased because the

supplemental camera's orientation SM declared the person to be laying in some of the recording in which the main camera's orientation SM declared the person not to be laying.

3.5.4. EVALUATION OF MBR TRACKER EFFICIENCY

We evaluated computational efficiency of our MBR tracker, which is discussed in detail in Section 2.6.2. The main point is that our MBR tracker's efficiency is an order-of-magnitude more efficient than 3D models.

3.5.5. FUTURE WORK

Much of the future work is discussed in Section 2.7. Additionally, future work will investigate computationally efficient methods of tracking the head or other computationally efficient mechanisms for detecting falls in confounding scenarios. The trade-off between precision of head location and lower computation will be investigated. Furthermore, future work needs to maintain a focus on classification accuracy as automated head tracking is likely to be less than perfect tracking.

3.6 CONCLUSION OF THE EFFICACY OF 3D HEAD TRACKING, 2D HEAD TRACKING, AND MOVING-REGION TRACKING

The key difference between the head and MBR-based fall detectors was that the head-based allowed a rule that the head had to be near the ground to be considered a

potential fall, which prevented some confounding non-fall recordings from being classified as falls. The accuracy of MBR-based fall detection matches the accuracy of head-based fall detection for non-confounding scenarios with perfect 1.00 sensitivity and specificity. Head-based and MBR-based also had equivalent sensitivities of 0.75 for confounding scenarios. However, the 3D head-based fall detector had a specificity of 1.00, 2D head-based had 0.85, and MBR-based had 0.54 for confounding scenarios. The difference in specificity was entirely caused by the head-based fall detectors knowing the head was not near the ground, whereas the MBR-based fall detector did not know. Our state machine-based algorithm with MBR data ran at 32.1 FPS for one camera while single-threaded on a 3.30 GHz Xeon processor, approximately 5-10x faster than 3D head tracking algorithms with comparable fall detection accuracy. MBR-based fall detection is suitable for a variety of scenarios; however, when higher accuracy is necessary and confounding situations are likely, the extra computation cost of head-tracking may be justified. We used manual head tracking in this work to determine an upper-bound. Automatic head trackers may have lower accuracy.

Chapter 4. ACCURATE AND EFFICIENT ALGORITHMS THAT ADAPT TO PRIVACY-ENHANCED VIDEO FOR IMPROVED ASSISTIVE MONITORING

4.1 INTRODUCTION TO VIDEO-BASED ASSISTIVE MONITORING AND PRIVACY-ENHANCED VIDEO

In-home assistive monitoring uses technology such as sensors and cameras to aid live-alone aging persons, typically involving automation. A common monitoring goal, for example, is to automatically detect that a person may have fallen and provide prompt notification to caregivers. A different goal is to automatically estimate a person's daily energy expenditure. Low expenditure is correlated with development of dementia, with falls, with depression, and other features, and thus detecting negative trends may help caregivers introduce appropriate interventions. Other goals include detecting other critical situations such as a person not arising by a certain morning time, staying in a room (such as a bathroom or laundry) much longer than normal, leaving the house at night an not returning within a time period, etc.

Body-worn sensors, household sensors, and cameras may be used in assistive monitoring, often in some combination. Video processing on camera video does well in achieving certain goals, with the benefit of not requiring a person to wear anything

special. However, video has the obvious drawback of decreased privacy. We previously examined several video privacy-enhancements, such as automatically covering a person by a box, that still enable some monitoring goal achievement. However, such privacy-enhancements may degrade quality of the automated monitoring. In this chapter, we examine adaptive algorithm's ability to compensate for privacy-enhanced video goal performance, and describe the idea of creating algorithms that automatically adapt themselves based on the video being received to improve quality.

4.2 BACKGROUND RELATED TO VIDEO-BASED ASSISTIVE MONITORING

4.2.1. PRIVACY-ENHANCED VIDEO IN ASSISTIVE MONITORING

Privacy-enhanced video increases the sense of privacy preservation over raw video [18][24]. 15 participants, who were over the age of 65, surveyed by Demiris [18] felt that the use of a camera for in-home assistive monitoring was obtrusive, while "many participants felt that [silhouetting] was more appropriate," (numerical data was not provided). In our earlier work [24], we surveyed 328 participants (average age of 20 years) with raw video, and privacy-enhanced video including person covered by blur, person replaced by silhouette, person covered by filled oval, person covered by filled rectangle, and person replaced by blue-outlined rectangle with trailing arrows videos. The effectiveness of each privacy enhancement was rated on two metrics: privacy score and

sufficiency of privacy. The privacy score was based on a 6-choice Likert scale for the statement, "This [privacy enhancement] protects Grandpa's privacy". Three videos were shown for each privacy enhancement. The privacy scores across the three videos were summed, giving a privacy score range from 0 (strongly disagree) to 18 (strongly agree). Raw's privacy score was 2.4, whereas other scores were: blur 9.5, silhouette 11.6, bounding-oval 14.0, bounding-box 15.5, and trailing-arrows 16.0. The sufficiency of privacy was the percentage of participants who responded that a privacy enhancement provided a sufficient amount of privacy vs. a insufficient amount of privacy. Only 2% said that raw video provided sufficient privacy, whereas other scores were: blur 23%, silhouette 59%, bounding-oval 88%, bounding-box 96%, and trailing-arrows 98%.

4.2.2. ASSISTIVE MONITORING

Commonly, only traditional sensors like motion sensors and door sensors are used in commercial assistive monitoring systems, to detect anomalies and provide configurable event-of-interest detection. BeClose [9] monitors traditional sensors for anomalies in daily activities and notifies a caregiver in the event of an anomaly. Motorola's Homesight [41] is configured by the caregiver or a technician to detect events of interest, such as a person leaving home at night. A camera can be configured to take a picture or record video when an event of interest is detected but is not used as a sensor to detect events. QuietCare [45] performs anomaly detection after learning typical patterns with motion sensors that are placed throughout the home and notifies the caregiver when an anomaly is detected. SmartHome [54] and X10 [60] offer a variety of monitoring kits and

products, such as an eight camera system with basestation security and a programmable thermostat.

Similar to commercial systems, academic assistive monitoring systems commonly use traditional sensors to detect anomalies, while some also integrate camera monitoring and provide programmable platforms. CASAS [47] performs anomaly detection on sensors, such as motion and light sensors, by learning normal behavior then detecting salient events. The Gator Tech Smart House [30] is a sensor platform that is programmable for detecting events of interest or anomalies. The types of sensors are vast, including floor sensors, plug sensors, and smart microwaves; however, cameras are not used to monitor the interior of the home due to privacy concerns [28]. We previously developed the Monitoring and Notification Flow Language (MNFL) [23] that integrates traditional sensors with cameras for assistive monitoring. MNFL also provided users with an easy to use and fast way to learn graphical programming for describing assistive monitoring systems. Users connect graphical blocks to describe the intended monitoring system. A graphical block could be a sensor (including video), actuator, or single function, such as OR-logic or a threshold.

Automated fall detection has been performed with on-body accelerometers [11], and off-body sonar [38] and cameras [3][20][50]. The accelerometers and sonars provide high-levels of privacy in that the identity and specific activity is not easily discernible. Some camera-based solutions are privacy-enhanced, such as Anderson [3] that uses extracted person-silhouettes to detect falls. We previously developed a camera-based solution [20] that allowed the user to choose among raw video or 4 privacy

enhancements: person covered by blur, person replaced by silhouette, person covered by filled oval, and person covered by filled rectangle.

Commercial automated energy expenditure is typically estimated by a device worn around the upper arm, such as the BodyBugg [13] or Body Media's armband [12]. These devices cost about $100-$150 with $5-7 monthly subscriptions and estimate energy expenditure using algorithms operating on data from the device's multiple sensors, which include a tri-axial accelerometer, a heat flux sensor, a galvanic skin response sensor, and a skin temperature sensor. The first version of the BodyBugg was compared to doubly labeled water, which is the most tested and reliable indirect measurement of energy expenditure method [53], for energy expenditure estimation and had on average 90% accuracy [34]. In this chapter, we used the second (assumedly more accurate) version of the BodyBugg, to compare video-based energy expenditure estimations with "actual" numbers. Another common approach to energy expenditure estimation is to wear a pedometer, which counts the number of steps taken. Although costing as little as a few dollars, pedometers are far less accurate for energy expenditure estimation [16].

Researchers have estimated energy expenditure with video-based approaches. Yao [61] used a body-worn camera attached to the user's shirt on the chest just below the neck, to distinguish between walking and jogging by measuring frame displacements, i.e., shifts in the video content. Yao's method was not evaluated against other energy estimation devices. In previous work, we [22] used a stationary camera to determine energy expenditure by analyzing the moving region in raw video.

One of our previous works [21] demonstrated that privacy enhanced video can be as accurate as raw video across many monitoring goals, but some privacy enhancements experienced monitoring goal accuracy degradation compared to raw video. This chapter extends that work by introducing and demonstrating that adaptive algorithms can compensate for the degradation. Additionally, this chapter demonstrates that the adaptive algorithms are computationally more efficient than the non-adaptive algorithm.

4.2.3. ADAPTIVE ALGORITHMS

An adaptive algorithm changes a programs behavior depending on the sensor data [10][55][59] and/or available resources [44][29]. An example of an algorithm that adapts to sensor data was implemented by Wood [59], in which an assisted-living sensor network adjusted the power usage for individual sensors based on the monitored person's behavior. In particular, a sensor's sampling rate was proportionally adjusted to the rate of sensor data change. Another example is fault monitoring in asynchronous distributed systems by Sotoma [55], in which time-out lengths are adjusted based on the recent history of elapsed times for pings. An example of an algorithm that adapts to available resources is adaptive sorting by Petersson [44], in which the available memory dictated the sorting algorithm chosen. Another example was developed by Guo [29] that adjusted sensor boundaries based on sensor placement. This adaptation allows for readjustments when a sensor moves or is no longer functional.

This chapter expands on our previous work by introducing algorithms that adapt to privacy-enhanced video to regain some of the monitoring goal quality loss due to the privacy enhancements. The video is analyzed for characteristics of privacy

enhancements, such as a solid-colored rectangle covering the person, which triggers an adaptation, such as hunt the video for that solid-color. Section 6 goes into further detail.

4.3 RECORDINGS

The recordings took place at a UC Riverside laboratory in-home environment. Three separate groups of recordings were taken: energy estimation, MNFL monitoring goals [23], and fall detection.

4.3.1. ENERGY ESTIMATION RECORDINGS

We recorded 4 actors for 9 activities each in a mock in-home environment that included a living room and dining room, shown in Figure 2-3(a). The energy expenditure estimation video dataset is available online, and the weblink is available in the references [26]. The environment was located in a research laboratory at the University of California at Riverside. Each recording contained a single actor performing a single activity, such as reading while seated, wiping down surfaces, or using a stair stepper exercise machine. Each recording lasted 30 minutes. Video was recorded at 30 frames per second, and energy estimates were recorded with a BodyBugg. A requirement for each recording was that the actor never left the camera's view.

We grouped the 9 activities into 3 categories based on the number of Calories expended per minute according to the BodyBugg: low activity level (less than 3 Cal./min.), medium activity level (between 3 and 6 Cal./min.), and high activity level (more than 6 Cal./min.). The grouped activities are described in previous work [22] but shown in Table 4-1 for the reader's convenience.

The camera came from a popular in-home 8-camera monitoring set made by Q-See and sold by Costco for around $400. The camera was positioned and had a viewing direction as shown in Figure 2-3(a). The camera was 182cm above the floor.

4 male actors performed the various activities. Actor 1 was 18 years old, actor 2 was 21, actor 3 was 20, and actor 4 was 21.

Activity description	Activity level
Sit on couch while reading a book	Low
Sit on couch while using a laptop	Low
Sit at dinner table while eating meal	Low
Slow-speed pace	Medium
Wipe the surfaces	Medium
Sweep the floors	High
Moderate-speed pace	High
Use stair stepper	High
Quick-speed pace	High

Table 4-1: Nine activities used for energy expenditure estimation that were recorded and categorized as either low activity level, medium activity level, or high activity level.

4.3.2. MNFL MONITORING GOALS RECORDINGS

The same recording environment and camera placement were used for MNFL monitoring goals as for energy estimation recordings, described in Section 4.3.1. Each recording contained a single actor. In some recordings, the actor performed one or more general events, such as entering or exiting the apartment. In some recordings, the actor performed a specific task, such as read on couch or sweep the floors. The recordings were grouped by monitoring goal, and each group had a set video length, such as 30 minute recordings for in room too long detection.

The actors were a 27-year old UCR graduate student researcher on this project and two UCR undergraduate student volunteers, ages 22 and 21, all males.

The MNFL video dataset is available online, and the web link is available in the references [40].

4.3.3. FALL DETECTION RECORDINGS

The same recording environment and camera placement were used for fall detection as for energy estimation recordings, described in Section 4.3.1. For this chapter, we recorded 24 raw videos (12 with and 11 without falls) of a 27-year old UCR graduate student researcher on this project. Each video is one minute long. The fall videos included stumbling and slipping on the floor, and loosing balance from the couch while reaching for a lamp. The non-fall videos included sweeping the floor, napping and watching television on the couch, and searching for a lost item involving stooping and bending.

The fall video dataset is available online, and the web link is available in the references [27].

4.4 FOREGROUNDING VIA FOREGROUND-BACKGROUND SEGMENTATION

Foregrounding, also referred to as MBR tracking, is described in detail in Section 2.3.

4.5 PRIVACY ENHANCEMENTS CONSIDERED

Cameras typically output raw video, which we define as follows:

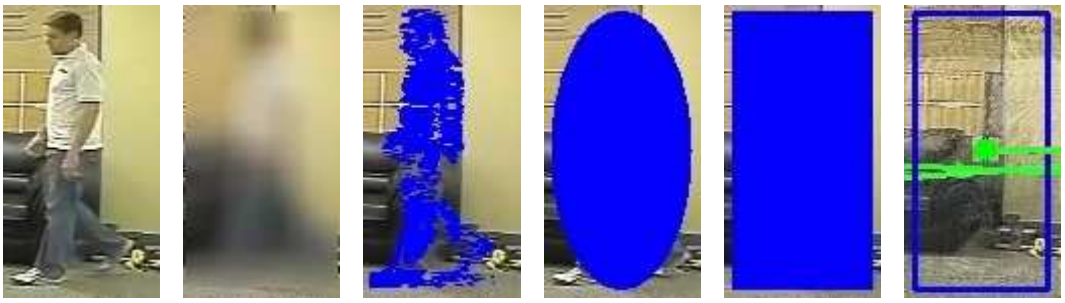
- Raw video, shown in Figure 4-1(a), is normal video that shows the camera's scene as clearly as possible.

A privacy-enhanced video intentionally obscures the appearance of a person in the video to protect that person's privacy. Raw video is perceived to have less privacy than privacy-enhanced video [18][24]. We consider five privacy enhancements:

- Blur video (Figure 4-1(b)) smears the video, typically restricting the smearing to the region with movement.
- Silhouette video (Figure 4-1(c)) covers the movement with an outline of the person filled with a solid color.
- Bounding-oval video (Figure 4-1(d)) covers the movement with a bounding oval around each person.
- Bounding-box video (Figure 4-1(e)) covers the movement with a bounding box around each person.
- Trailing-arrows video (Figure 4-1(f)) replaces the movement with a hollow box having a dot in the center, with trailing colored arrows indicating prior activity.

We built a tool to convert raw video to privacy-enhanced video. The raw video was processed with our foreground-background segmentation algorithm to extract a foreground and the MBR around the foreground. The blur video blurred the region of the raw video in which the MBR resides. The silhouette video changed a pixel to blue if that pixel was part of the foreground and within the region of the MBR. The bounding-oval video covered the region of the MBR with a solid blue oval that has the same height and

width of the MBR. The bounding-box video covered the region of the MBR with a solid blue rectangle with the same height and width of the MBR. The trailing-arrows video covered the region of the MBR with a hollow blue rectangle with the same height and width of the MBR. A green dot was displayed at the center of the MBR for trailing-arrows, and colored arrows trailed from prior activity.



(a) **(b)** **(c)** **(d)** **(e)** **(f)**

Figure 4-1: Pictures from the same moment of the same recording of (a) raw, (b) blur, (c) silhouette, (d) bounding-oval, (e) bounding-box, and (f) trailing-arrows.

4.6 ADAPTIVE ALGORITHMS FOR PRIVACY-ENHANCED VIDEO PERSON EXTRACTION

Foregrounding, as described in Section 4, may be used to extract the person from privacy-enhanced video since the privacy enhancement tends to move with the person. However, privacy-enhanced-video has degraded monitoring goal accuracy compared to raw video with foregrounding [21], such as blur due to the blending of foreground and background to cause the blur effect. An adaptive algorithm may compensate for the degraded accuracy by using characteristics of the privacy-enhanced video, such as the moving region is mostly one color, to improve person extraction by triggering

adaptations, such as search the frame for that one color. As Figure 4-2 shows, the default algorithm for person extraction is foregrounding and that an adaptation may be used if a particular characteristic is detected.

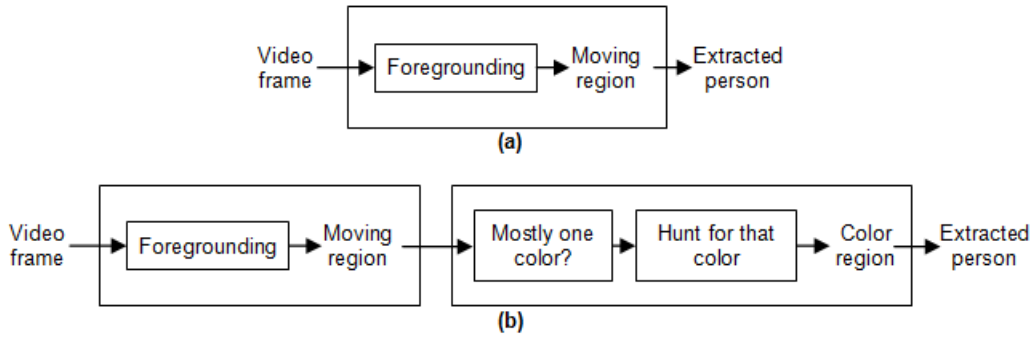
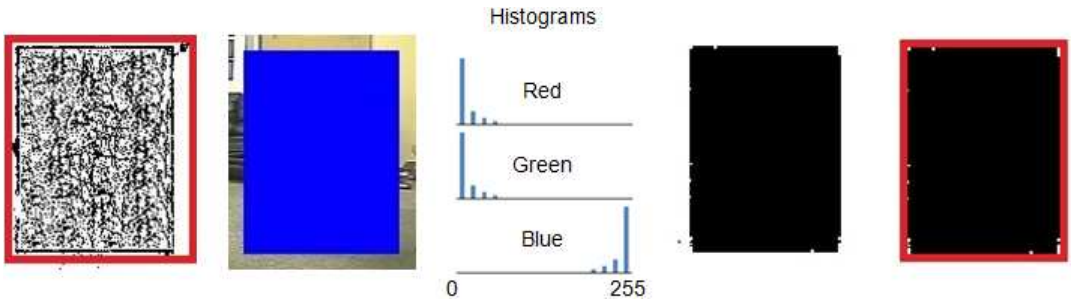


Figure 4-2: (a) Default person extraction is foregrounding. (b) If a characteristic in moving region was detected (e.g., mostly one color), then an adaptation (e.g., hunt for that color) was used to refine person extraction.

4.6.1. ADAPTATION: SPECIFIC-COLOR HUNTER

If the moving region was mostly one color (such as with bounding-oval and bounding-box) for a sufficiently long time, then we adapted by specifically hunting for that color. The MBR result of foregrounding a video frame, shown in Figure 4-3(a), was used to define the size of the moving region, shown in Figure 4-3(b). We determined whether the moving region of the video frame was one color by creating a histogram for red, green, and blue, shown in Figure 4-3(c) and described in Figure 4-4(b). The histograms used bin sizes of 4, e.g. red pixel colors between 0 and 3 were counted for one bin and red pixel colors between 4 and 7 were count for another. If the largest bin of each histogram contained more than double that of the second largest bin of the respective histogram, then the moving region was considered to be mostly one color. The largest bin

of each histogram formed the color range, e.g. the largest bin of the red histogram formed the red range.



(a) **(b)** **(c)** **(d)** **(e)**

Figure 4-3: (a) Default person extraction is foregrounding. (b) If a characteristic in moving region was detected (e.g., mostly one color), then an adaptation (e.g., hunt for that color) was used to refine person extraction. Specific-color hunter used (a) the MBR result of foregrounding to create (b) the moving region. (c) The histograms for red, green, and blue determined whether the moving region was mostly one color. If so, then (d) an image of only that color was used to build (e) an MBR.

The specific-color hunter traversed all pixels of the video frame, as described in Figure 4-4(c). If a pixel was within the color ranges determined by the histograms give or take 8, then the pixel was colored white. Otherwise, the pixel was colored black. The resulting image, shown in Figure 4-3(d), was passed through the MBR builder, described in Section 4, to generate an MBR representing the extracted person, shown in Figure 4-3(e) .

The specific-color hunter was controlled by an automated sliding scale, shown in Figure 4-4(a), to determine whether the moving region had contained mostly one color for a sufficient amount of time. The scale slider incremented each time the moving region contained mostly one color and decremented otherwise. The scale ranged between 0 and 40. If the scale slider was 15 or greater, then the specific color hunter was turned on. If

the scale slider reached 0, then the specific color hunter was turned off. The scale slider was initialized to 0.

```
1  Given a recording: Video recording
2
3  Slider sldr = new Slider()
4
5  while ( recording.hasMoreFrames() )
6      Image currFrame = recording.getNextFrame()
7      Image foreground = foregrounding(currFrame)
8      Rectangle currMBR = mbrBuilder(foreground)
9
10    Color c = isRegionOneColor(currFrame, currMBR, c)
11    if ( c == null )
12        sldr.sliderUpdate( -1 )
13    else
14        sldr.sliderUpdate( +1 )
15
16    if ( sldr.colorHunterOn() )
17        foreground = specificColorHunter(currFrame, c)
18        currMBR = mbrBuilder(foreground)
19
20    outputToFile(currMBR)
```

(a)

```
1  Color isRegionOneColor(Image frame, Rectangle mbr, Color c)
3      Image movingRegion = frame.getSubImage(mbr)
4      HistogramArray histograms = buildHistograms(movingRegion)
6      for each (Histogram h in histograms) // (r, g, b)
7          if (h.getLargestBin() > 2*h.getSecondLargestBin())
6              c <- h.largest_bin
7      else
8          return null
9
10     return c
```

(b)

```
1  Image specificColorHunter(Image frame, Color c)
2      if ( c == null )
3          c = previousColor
4      Image foreground = emptyImage()
5      for each (Pixel p in frame)
6          if p.similarTo(color)
7              foreground[p] = white
8          else
9              foreground[p] = black
10
11     return foreground
```

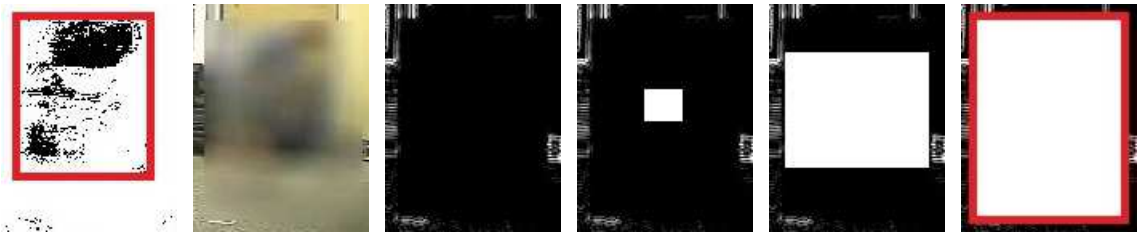
(c)

Figure 4-4: Pseudocode for (a) specific-color hunter adaptation, (b) check whether the moving region contains mostly one color, and (c) scan a frame for a specific-color.

The parameters of histogram analysis, color comparison, and sliding scale were trained manually with training video with a male graduate student on this project taken in the same environment described in Section 4.3.

4.6.2. ADAPTATION: EDGE-VOID FILLER

Video in which the moving region contains few edges suggests a blur privacy enhancement was used because blurring blends pixel colors together thus causing less edges to exist. If the moving region contained few edges, then we assumed a privacy enhancement was used and adapted by filling the void of edges in each direction starting from the center of motion to create an MBR. The MBR result of foregrounding a video frame, shown in Figure 4-5(a), was used to define the size of the moving region, shown in Figure 4-5(b). We determined whether the moving region, shown in Figure 4-5(b), contains few edges by performing edge detection, shown in Figure 4-5(c), then calculating the ratio of edges to non-edges in the moving region. Edge detection was performed by applying a Gaussian blur to the moving region of a video frame to reduce noise, then converting the image to grey scale. The default Laplace operator [33] in OpenCV 2.4.0 [43] was applied to the grey scale image resulting in an image with the edges detected, in which areas of larger pixel color change were colored closer to white than black. We calculated the ratio of edges to non-edges by summing the number of pixels in the edge detected image that had color values less than 25, then divided the sum by the area of the moving region. The moving region was said to contain few edges if the ratio of edges to non-edges was less than 0.30.



(a) **(b)** **(c)** **(d)** **(e)** **(f)**

Figure 4-5: Edge-void filler used (a) the MBR result of foregrounding a frame with a person sweeping to create (b) the moving region. (c) Edge detection was performed on the moving region. If the ratio of black-to-non-black pixels in the edge detection exceeded a threshold, then (d)-(f) a rectangle filled the black pixels, resulting in (f) the person extraction.

As Figure 4-5(c) shows, the location that the blur resided in the moving region was void of edges after edge detection. However, the boundaries of the blurred region are not clear. The observation is that an MBR can be created by filling the edge-void from the center of the moving region until each side of the rectangle hits a detected edge. Edge-void filler is shown in Figure 4-5(d)-(f) and described in Figure 4-6(c). Since foregrounding the privacy enhancements, such as blur, often resulted in a smaller than expected MBRs as seen in Figure 4-5(a), we increased the size of the moving region by 10% in both directions on the x and y-axis. We then performed the same edge detection described above to the size-increased moving region. On the resulting edge detected image, a 2x2-pixel rectangle was drawn at the center of the foregrounding's MBR. The rectangle grew one pixel left, right, up, then down until that direction touched a pixel with a color greater than or equal to 50. When the rectangle could not grow in any direction, that rectangle was said to be the MBR of the person.

```
1  Given a recording: Video recording
2                                                              (a)
3  Slider sldr = new Slider()
4
5  while( recording.hasMoreFrames() )
6      Image currFrame = recording.getNextFrame()
7      Image foreground = foregrounding(currFrame)
8      Rectangle currMBR = mbrBuilder(foreground)
9
10     if ( containsFewEdges(currFrame, currMBR) )
11         sldr.sliderUpdate( +1 )
12     else
13         sldr.sliderUpdate( -1 )
14
15     if ( sldr.voidFillerOn() )
16         foreground = edgeVoidFiller(currFrame
currMBR)
17         currMBR = mbrBuilder(foreground)
18
19     outputToFile(currMBR)
```

```
1  Boolean containsFewEdges(Image frame, Rectangle mbr)
2      Image movingRegion = frame.getSubImage(mbr)                (b)
3      Image blurredImage = applyGaussBlur(movingRegion)
4      Image greyImage = convertToGreyScale(blurredImage)
5      Image edgesDetected = applyLaplaceOp (greyImage)
6
7      Number edgePixels = countEdgePixels(edgesDetected)
8      Number nonEdgePixels = pixelCount(edgesDetected) - edgePixels
9
10     if ( ( edgePixels / nonEdgePixels ) < 0.30 )
11         return true
12     else
13         return false
```

```
1  Image edgeVoidFiller(Image frame, Rectangle mbr)
2      Rectangle biggerMBR = increaseRectSize (mbr)          (c)
3      Image bgrMovingReg = frame.getSubImg(biggerMBR)
4      Image bgrBlurImage = applyGaussBlur(bgrMovingReg)
5      Image bgrGreyImg = convertToGrey (bgrBlurImage)
6      Image bgrEdges = applyLaplaceOp(bgrGreyImg)
7
8      Rectangle newMBR = centerOfRectangle(biggerMBR)
9      Boolean canGrow = true
10     while ( canGrow )
11         canGrow,newMBR = growMBR(newMBR, bgrEdges)
12
13     return newMBR
```

Figure 4-6: Pseudocode for (a) specific-color hunter adaptation, (b) check whether the moving region contains mostly one color, and (c) scan a frame for a specific-color.

Edge-void filler was controlled by an automated sliding scale, described in Figure 4-6(a), to determine whether the moving region had contained few edges for a sufficient amount of time. The scale was between 0 and 15. The scale slider incremented when the moving region contained few edges and decremented otherwise. The adaptation turned on when the scale slider reached 15 and turned off when the scale slider reached 0. The scale slider was initialized to 0.

The parameters of edge-to-non-edge threshold, growth algorithm, and sliding scale were trained manually with the same training video as Section 4.6.1

4.7 CAMERA-BASED ASSISTIVE MONITORING GOALS

Live-alone persons, particularly elderly live-alone persons, may wish to be monitored for situations of interest that indicate a problem. The assistive monitoring goals would be determined by the monitored person or that person's caregiver, which could be an adult child or nursing staff. Some goals can be solved with a camera-based approach, such as the following 8 goals that this chapter considers: energy expenditure estimation, in room too long, leave but not return at night, arisen in morning, not arisen in morning, in region too long, abnormally inactive during day, and fall detection.

The camera could be installed by the monitored person, the caregiver, or a trained technician. Similarly, the assistive monitoring goals using the installed camera could be configured by the monitored person, caregiver, or a trained technician using an assistive

monitoring language such as MNFL [23]. Our previous work [21] describes in detail the importance of and particular method for solving the 8 monitoring goals used in this work.

4.8 EXPERIMENTS OF NON-ADAPTIVE (FOREGROUNDING) CAMERA-BASED MONITORING GOALS

Our previous work [21] describes in detail our experimental method and results using non-adaptive algorithms. The combined fidelity was produced by applying Fisher's transformation to each correlation between an actor's video-based and BodyBugg energy estimation, averaging the transformations, then applying the inverse of Fisher's transformation on the average to create a combined correlation. Table 2 contains the aggregated results of the experiments. Energy estimation and fall detection yielded the widest variation of results of the monitoring goals. For energy estimation, raw video had a significantly higher accuracy ($p < 0.001$) than the other privacy enhancements. The combined fidelities for each privacy-enhancement was the same as raw with $p < 0.001$. For fall detection, raw video had the highest accuracy with 1.0 for average sensitivity and specificity, while silhouette had the second highest with 0.92 average sensitivity and 0.83 average specificity. Of the remaining monitoring goals, the sensitivity and specificity were perfect 1.0, except blur video's sensitivity of 0.5 for in region too long and trailing-arrows room-too-long sensitivity of 0.0 and abnormally inactive specificity of 0.67.

Privacy enhancement	Energy estimation average accuracy	Energy estimation combined fidelity	Room too long average sensitivity/ specificity	Arisen in morning average sensitivity/ specificity	Region too long average sensitivity/ specificity	Abnormally inactive average sensitivity/ specificity	Fall detection average sensitivity/ specificity
Raw	*90.9%*	*0.997*	*1.0 / 1.0*	*1.0 / 1.0*	*1.0 / 1.0*	*1.0 / 1.0*	*1.0 / 1.0*
Blur	80.5% †	0.994 ‡	1.0 / 1.0	1.0 / 1.0	0.5 / 1.0	1.0 / 1.0	0.75 / 0.75
Silhouette	85.0% †	0.998 ‡	1.0 / 1.0	1.0 / 1.0	1.0 / 1.0	1.0 / 1.0	0.92 / 0.83
Bounding-oval	85.6% †	0.997 ‡	1.0 / 1.0	1.0 / 1.0	1.0 / 1.0	1.0 / 1.0	0.83 / 0.83
Bounding-box	84.3% †	1.000 ‡	1.0 / 1.0	1.0 / 1.0	1.0 / 1.0	1.0 / 1.0	0.92 / 0.75
Trailing-arrows	79.8% †	0.997 ‡	0.0 / 1.0	1.0 / 1.0	0.0 / 1.0	1.0 / 0.67	0.5 / 0.67

Table 4-2: Foregrounding experiments for each assistive monitoring goal by privacy enhancement. Higher metrics are better. The privacy enhancements of silhouette, bounding-oval, and bounding-box cause some degradation in quality compared to raw video.

† indicates significantly lower than raw ($p < 0.001$)

‡ indicates significantly the same as raw ($p < 0.001$)

4.9 EXPERIMENTS OF ADAPTIVE CAMERA-BASED ASSISTIVE MONITORING GOALS

The previous section shows that privacy-enhanced video causes some degradation in monitoring quality. This section examines whether our two proposed adaptations can decrease that degradation. We applied the same experimental methods used in Section 4.8 with the adaptive algorithms described in Section 4.6.

4.9.1. IMPACT OF SPECIFIC-COLOR HUNTER AND EDGE-VOID FILLER ADAPTATIONS ON ALL VIDEOS

We ran the two adaptations on all privacy enhancements to see the adaptations' impacts. Table 4-3 summarizes the changes from Table 4-2 due to applying the specific-color hunter adaptation to each video. Table 4-4 does so for the edge-void filler adaptation. The data shows that the adaptations are helpful for particular privacy enhancements but hurtful for others, as is to be expected.

Privacy enhancement	Energy estimation average accuracy	Energy estimation combined fidelity	Room too long average sensitivity/ specificity	Arisen in morning average sensitivity/ specificity	Region too long average sensitivity/ specificity	Abnormally inactive average sensitivity/ specificity	Fall detection average sensitivity/ specificity
Raw	-1.90% †	= ‡	= / =	= / =	= / =	= / =	= / =
Blur	**+0.20%** †	**=** ‡	= / =	= / =	= / =	= / =	**+0.08** / =
Silhouette	**+0.10%**	**+0.013** ‡	= / =	= / =	= / =	= / =	= / **+0.09**
Bounding-oval	**+2.50%** †	**+0.003** ‡	= / =	= / =	= / =	= / =	**+0.17 / +0.17**
Bounding-box	-1.70% †	**=** ‡	= / =	= / =	= / =	= / =	= / **+0.17**
Trailing-arrows	-0.30% †	= ‡	= / =	= / =	= / =	= / =	= / =

Table 4-3: Specific-color hunter experiments compared to the foregrounding experiments. Specific-color hunter had many improvements in fall detection, and with bounding-oval in energy estimation accuracy.

*** Positive values are bold and mean improved performance by an absolute amount, e.g. +2.50% means specific-color hunter had 2.5 more percentage points, e.g. 85.6% to 88.1% is +2.50%.**

*** Negative values mean degraded performance by an absolute amount.**

*** = means no change in performance.**

† indicates significantly different than foregrounding ($p < 0.05$)

‡ indicates significantly the same as foregrounding ($p < 0.001$)

Privacy enhancement	Energy estimation average accuracy	Energy estimation combined fidelity	Room too long average sensitivity/ specificity	Arisen in morning average sensitivity/ specificity	Region too long average sensitivity/ specificity	Abnormally inactive average sensitivity/ specificity	Fall detection average sensitivity/ specificity
Raw	-1.80% †	= ‡	= / =	= / =	= / =	= / =	-0.17 / =
Blur	**+6.30%** †	**+0.036** ‡	-0.5 / =	= / =	-0.5 / =	-0.5 / -0.17	**+0.25** / -0.07
Silhouette	-3.80% †	= ‡	= / =	= / =	= / =	= / =	= / =
Bounding-oval	**+1.90%**	**+0.006** ‡	= / =	= / =	= / =	= / =	-0.41 / -0.25
Bounding-box	**+4.10%** †	**+0.005** ‡	= / =	= / =	= / =	= / =	= / -0.42
Trailing-arrows	=	= ‡	= / =	= / =	= / =	= / =	= / =

Table 4-4: Edge-void filler experiments compared to the foregrounding experiments. Edge-void filler had big improvements in energy estimation accuracy with blur, bounding-oval, and bounding-box.

*** Notes and notation same as Table 4-3.**

4.9.2. PERFORMING ADAPTATIONS SPECIFICALLY

The device that privacy-enhances the assistive monitoring video could also send metadata indicating which privacy enhancement was applied. Knowing the privacy enhancement enables a privacy-enhancement-aware (PE-aware) algorithm to select the best non-adaptive or adaptive algorithm for each particular monitoring goal. By default, PE-aware will select the non-adaptive algorithm, foregrounding, to minimize computation. For example, Table 5 shows the energy expenditure estimation accuracy comparison of privacy enhancement by non-adaptive and adaptive algorithms. Specific-color hunter had the highest accuracy for bounding-oval compared to foregrounding and edge-void filler; therefore, PE-aware selects specific-color hunter to run on the bounding-oval video for energy expenditure estimation accuracy. The adaptive algorithms compensated for performance degradation of the non-adaptive algorithm with privacy-

enhanced video compared to raw video. The average privacy enhancement degradation of the non-adaptive algorithm compared to raw was from 90.9% accuracy to 83.9%. The adaptive algorithms significantly improved accuracy of the average privacy enhancement up to 87.1%. Trailing-arrows is exempt from this analysis because trailing-arrows performance does not change across non-adaptive and adaptive algorithms.

Table 4-6 shows the fall detection sensitivity and specificity for non-adaptive and adaptive algorithms. The adaptive algorithms compensated for performance degradation with foregrounding of privacy-enhanced video compared to raw video. The privacy-enhanced video with foregrounding degradation was from 1.0 to 0.86 sensitivity and 1.0 to 0.79 specificity. Adaptive algorithms compensated privacy-enhanced video up to 0.92 sensitivity and 0.90 specificity.

	Non-adaptive	Adaptive		
Privacy enhancement	Foregrounding	Specific-color hunter	Edge-void filler	PE-aware
Blur	80.5%	<	86.8% †	86.8% †
Silhouette	85.0%	85.1% †	<	85.1% †
Bounding-oval	85.6%	88.1% †	<	88.1% †
Bounding-box	84.3%	<	88.4% †	88.4% †
Average	**83.9%**			**87.1% †**
Raw	**90.9%**	<	<	90.9%

Table 4-5: Energy expenditure estimation accuracy for non-adaptive and adaptive algorithms.

Foregrounding on raw video achieved the higher accuracy of 90.9% over the average privacy enhancement with 83.9%. PE-aware algorithm improved the accuracy of the average privacy enhancement to 87.1%.

*** < means the adaptive algorithm performed worse than foregrounding,**

*** = means the adaptive algorithm performed the same as foregrounding,**

† indicates significantly greater than foregrounding ($p < 0.05$)

Privacy enhancement	Non-adaptive: Foregrounding (sensitivity / specificity)	Adaptive: Specific-color hunter (sensitivity / specificity)	Adaptive: Edge-void filler (sensitivity / specificity)	Adaptive: PE-aware (sensitivity / specificity)
Blur	0.75 / 0.75	0.83 / 0.75	<	0.83 / 0.75
Silhouette	0.92 / 0.83	0.92 / 0.92	=	0.92 / 0.92
Bounding-oval	0.83 / 0.83	1.0 / 1.0	<	1.0 / 1.0
Bounding-box	0.92 / 0.75	0.92 / 0.92	<	0.92 / 0.92
Average	**0.86 / 0.79**			**0.92 / 0.90**
Raw	**1.0 / 1.0**	=	<	1.0 / 1.0

Table 4-6: Fall detection sensitivity and specificity of non-adaptive and adaptive algorithms.

Foregrounding privacy-enhanced video degraded accuracy from foregrounding raw video, from 1.0 sensitivity to 0.86 and 1.0 specificity to 0.79. Adaptive algorithms compensated accuracy back up to 0.92 sensitivity and 0.90 specificity.

*** < means the adaptive algorithm performed worse than foregrounding,**

*** = means the adaptive algorithm performed the same as foregrounding**

PE-aware selects foregrounding for energy expenditure estimation fidelity for all privacy enhancements, except bounding-box, which selects specific-color hunter. For the MNFL goals, PE-aware always selects foregrounding. Specific-color hunter ties foregrounding for each MNFL goal and edge-void filler performs worse with blur in room too long, region too long, and abnormally inactive.

4.9.3. ADAPTIVE ALGORITHM EFFICIENCY

We evaluated the computation efficiency of the two proposed adaptive algorithms and the non-adaptive (foregrounding only) algorithm by determining the processing rate of each algorithm in frames per second (FPS) on each privacy enhancement. The recordings used during the evaluation were the 36 energy estimation recordings, which

were each at least 30 minutes in length. The first 5 seconds of each recording were used to train the background. The computer used during evaluation had a 3.30 GHz Intel Xeon processor with 8GB of memory and ran 64-bit Windows 7. The computer was rebooted after each algorithm's evaluation. The algorithms were run single-threaded.

Table 4-7 shows the FPS of foregrounding by privacy enhancement, and the percent difference of FPS for each adaptive algorithm and foregrounding. Color hunter has a significantly ($p < 0.01$) higher FPS than foregrounding for blur by 4.9% and fading lines by 5.0%, but a significantly lower ($p < 0.01$) FPS for raw by 0.9%. Similarly, void filler has a significantly higher ($p < 0.01$) FPS than foregrounding for blur by 3.2%, silhouette by 4.1%, bounding-box by 1.7%, and fading lines by 2.6%, but a significantly lower FPS for raw by 1.1%. Across privacy enhancements, the adaptive algorithms had significantly higher FPS ($p < 0.01$) than foregrounding, color hunter by 1.5% on average and void filler by 1.9%.

Privacy enhancement	Foregrounding (FPS)	Color hunter % difference from foregrounding	Void filler % difference from foregrounding
Raw	128	-0.9%‡	-1.1%‡
Blur	142	+4.9%‡	+3.2%‡
Silhouette	110	+0.7%	+4.1%‡
Bounding-oval	120	-1.1%	+1.1%
Bounding-box	124	-0.3%	+1.7%
Fading lines	116	+5.0%‡	+2.6%‡
Average	123	**+1.5%‡**	**+1.9%‡**

Table 4-7: Foregrounding FPS compared to adaptive algorithm FPS across privacy enhancements. The adaptive algorithms tend to have higher FPS than foregrounding. Higher is better.

‡ indicates significantly different from foregrounding ($p < 0.01$)

4.10 CONCLUSION ON ADAPTIVE ALGORITHMS FOR VIDEO-BASED IN-HOME MONITORING

Privacy-enhanced video degrades the accuracy of monitoring goals compared to raw video when using non-adaptive algorithms. We showed that two adaptive algorithms, specific-color hunter and edge-void filler, can help compensate for that degradation without loosing computational efficiency. Energy estimation accuracy from foregrounding raw to privacy-enhanced degraded from 90.9% to 83.9%, but the adaptive algorithms significantly compensated by bring the accuracy back up to 87.1%. Similarly, fall detection accuracy degraded from 1.0 sensitivity to 0.86 and 1.0 specificity to 0.79, but the adaptive algorithms compensated accuracy back up to 0.92 sensitivity and 0.90 specificity. Additionally, the adaptive algorithms were computationally more efficient than the non-adaptive algorithm, with color hunter processing 1.5% more frames per second than foregrounding and void-filler 1.9% more.

Future work includes further compensations to get closer to the results from raw video, determining which privacy enhancement is seen based on video rather than relying on metadata, increasing the number of evaluated monitoring goals, increasing the number of privacy enhancements considered, and using data from other types of sensors too like motion or sound sensors.

Chapter 5. PRIVACY PERCEPTION AND FALL DETECTION ACCURACY FOR IN-HOME VIDEO ASSISTIVE MONITORING WITH PRIVACY ENHANCEMENTS

5.1 INTRODUCTION TO PRIVACY PERCEPTION

Privacy is critical for adoption of video-based monitoring technology [7][8][18], but the ability to detect critical events decreases as methods to protect privacy are added. We explore tradeoffs among privacy perception and fall detection accuracy via an experiment involving several hundred participants. We considered the video privacy settings shown in , including raw video and five privacy-enhanced videos: blur, silhouette, oval, box, and trailing-arrows. We assume a system may include several video privacy setting options because the ability to customize has been shown to be important to the usage of assistive technology [8][18].

This chapter explores privacy and fall detection tradeoffs among those privacy settings, and examines whether a privacy enhancement technique exists that provides sufficient perceived privacy protection while enabling accurate fall detection by humans.

Blurring video is a common approach to privacy protection but may not provide sufficient privacy [14]. Lee [37] surveyed 47 participants with video privacy settings of

raw video, slightly-blurred video, heavily-blurred video, and no video (audio only). 44 participants preferred to "disable their camera when privacy was desired" and used raw video when privacy was not desired, thus indicating that blur may not be a practical video privacy setting. Boyle [14] surveyed 20 participants with two video privacy settings: blur video and pixelized video. Pixelized video segmented a raw video into a 2-dimensional grid with equal-sized squares. The color of each pixel in each square was averaged and that average color was applied to the entire square. Each video privacy setting could be adjusted from level 1 to 10, in which level 1 had extremely high levels of blur (or pixelization) and level 10 had extremely low levels of blur (or pixelization). Participants gave level 5 blur video and level 6 pixelized video a privacy score of about 3.5 out of 5.0, in which a score of 5.0 is highly protected privacy, and participants were able to identify activities at level 5 blur video and level 7 pixelized video. Neustaedter [42] surveyed 20 participants with 10 levels of blur to determine the balance between privacy and awareness, including high risk activities, such as changing clothes. Blur levels 1 and 2 were adequate for privacy, and blur levels 3 to 5 were adequate for awareness. Since more privacy is required than a viewer is able to determine what is happening, blur is not a good privacy setting.

Silhouetted video has been examined for in-home privacy. Demiris [18] surveyed 15 participants over the age of 65, all of whom felt that the use of a camera for in-home monitoring was obtrusive, while "many participants felt that [silhouetting] was more appropriate," (numerical data was not provided).

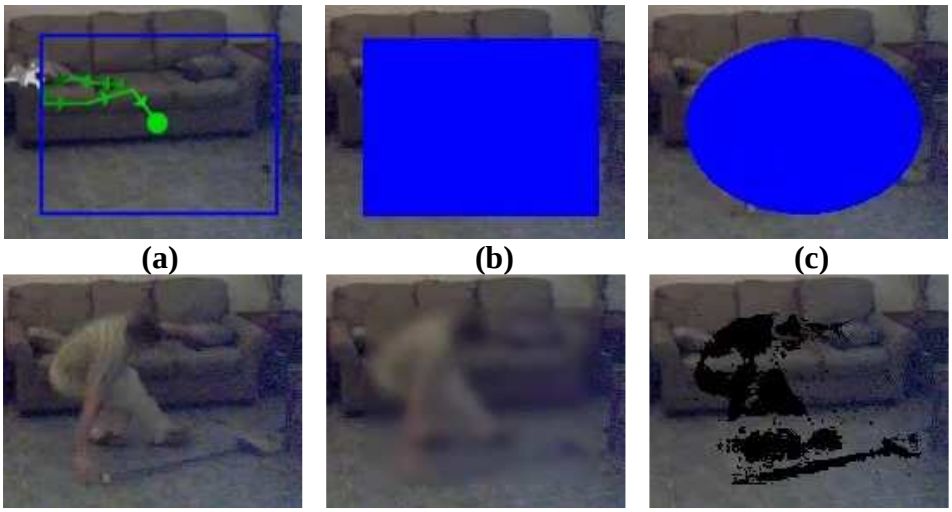


Figure 5-1: The same frame of an original video showing (a) raw, (b) blur, (c) silhouette, (d) oval, (e) box, (f) trailing-arrow video.

Exotic video privacy settings are often preferred for privacy protection over raw video but have comparatively worse detection accuracy [15][62]. Caine [15] surveyed 25 participants over the age of 65, and compared raw video to point-light video, which is a black background with white dots representing the silhouette of the person, and a blob-tracker video, which is a black background with a colored blob that provides information on the location and activity level of the person. Point-light video and blob-tracker video were both perceived to provide significantly more privacy ($p < 0.05$) than raw video. The ability to detect problem situations such as falls was not evaluated. Zhao [62] surveyed 20 participants with the following video privacy settings: raw video, pixelized video, live-shadow video, shadow video and edge-detected video. Live-shadow video pixelated only the foreground of the raw video. Shadow video placed transparent, pixelated foreground video onto a background image. Edge-detected video was a black and white video in which high contrast edges were white and the rest of the video was black.

Participants identified both the actor and the activity of the actor with raw video in 85% of the videos, 73% for pixelized videos, 66% for edge-detected videos, 55% of live-shadow videos and 26% of shadow videos. The sufficiency of privacy provided was not evaluated.

5.2 VIDEO PRIVACY SETTINGS

A video privacy setting is a video attribute that indicates the type of privacy enhancement applied to the video. We describe six such privacy settings below.

Cameras typically output raw video:

- Raw video (Figure 5-1(a)) is unaltered video that shows the camera's scene as clearly as possible.

A privacy-enhanced video obscures a person's appearance in the video. We consider five privacy enhancements:

- Blurred video (Figure 5-1(b)) smears the video, typically restricting the smearing to the region with the person.
- Silhouetted video (Figure 5-1(c)) replaces the person with a solid shape of the person.
- Bounding-oval video (Figure 5-1(d)) covers the person with a solid oval.
- Bounding-box video (Figure 5-1(e)) covers the person with a solid box.
- Trailing-arrows video, a method we developed for this experiment (Figure 5-1(f)), replaces the person with a hollow box having a dot in the center, with trailing colored arrows indicating prior activity.

We built two tools to convert raw video to privacy-enhanced video. The first tool takes as input raw video and a background image, and outputs the MBR data and a blurred, silhouetted, bounding-oval, and bounding-box video. The first tool is described in Section 2.3.

The second tool converted the MBR data of the raw video and a background image to a trailing-arrow video using ActionScript [1]. The tool drew the MBR as a hollow blue box, as in Figure 5-2. A bright green dot was positioned in the center of the hollow box. The dot was solid when there was activity and hollow when there had been no activity for four or more seconds. A solid line connected a dot created from activity to the previous dot. Midway through the line was an arrow indicating the direction of the activity. The color of a line started as bright green, the same color as the dot. The color of a line faded to black in 10 seconds (Figure 5-2(a)), then to white in another 10 seconds (Figure 5-2(b)), thus showing a trace of the person's activity. Additionally, if there was no activity for at least four seconds, then text would appear near the last activity indicating how long since the last activity. For example, if there had not been activity for seven seconds, then the text would be, "7 sec since last activity," as in Figure 5-2(b).

In an initial trial, which did not include any participants from this chapter's experiment, participants indicated two difficulties interpreting the trailing-arrow video. The first was that participants had a hard time determining where the person had exited a scene. In response, we adjusted the fading of the last five lines, such that the last line did not fade at all and the previous four lines would consecutively fade more to dark green. The second difficulty was that the participants had a hard time interpreting the motion

history when there was a lot of activity. We reduced the number of lines with two pruning techniques: do not place a line more than every 333ms, and do not place a new line if the MBR has not moved at least 25 pixels in any direction. These modifications were included in the trailing-arrow videos used in this chapter's experiment.

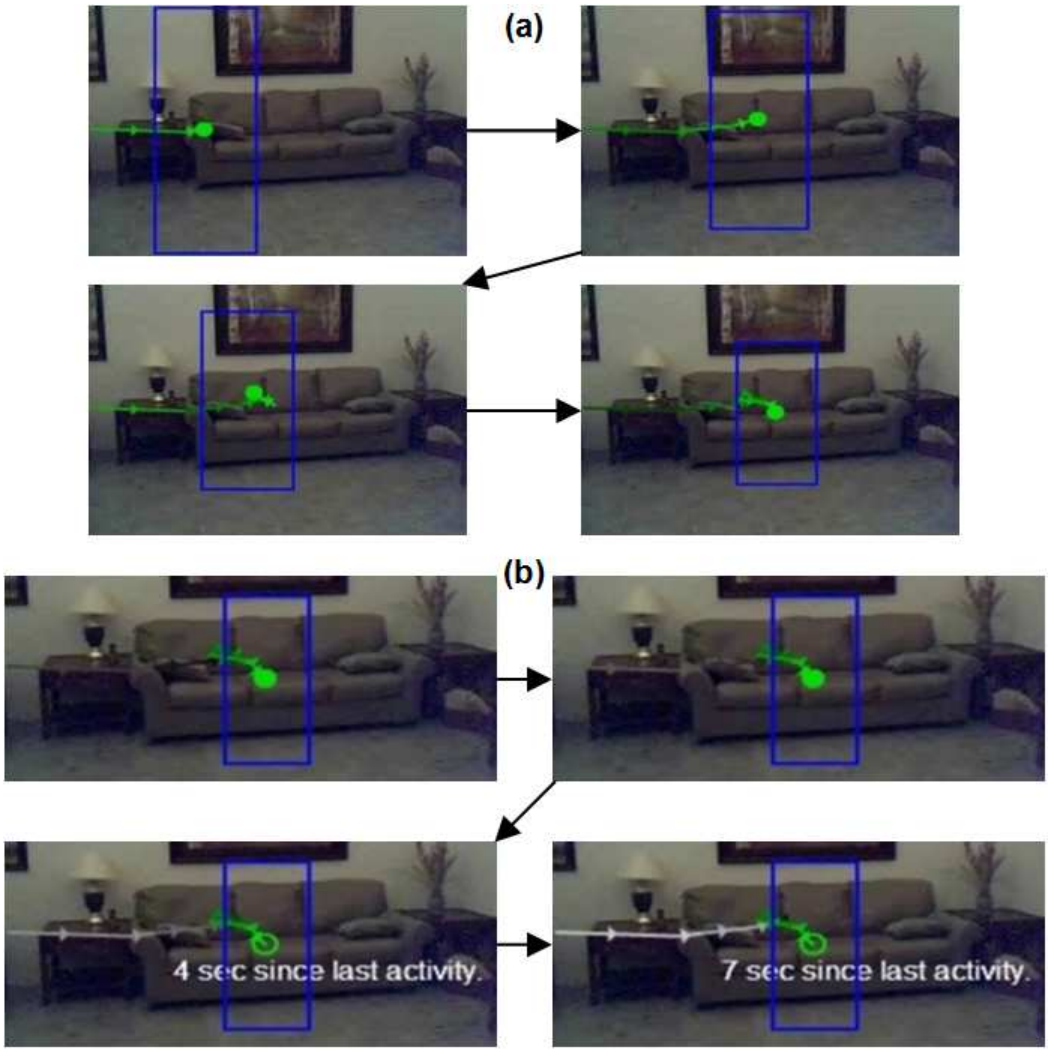


Figure 5-2: Two trailing-arrows picture sequences: (a) person walks into scene toward couch as the green trail turns to black, and (b) person sitting on couch as black trail turns to white.

5.3 METHODS

We conducted an experiment to explore tradeoffs between privacy perception and fall detection on raw video and privacy-enhanced video, and to determine whether a privacy setting exists providing sufficient perceived privacy and accurate fall detection.

5.3.1. PARTICIPANTS

The participants were undergraduate students at the University of California, Riverside (UCR) with an average age of 20 years, ranging from 17 to 42 years. The average participant had some experience providing in-home care and almost no experience receiving in-home care. The average participant uses a social networking website several times a day and has installed several software programs and hardware components, such as a printer or webcam. The average participant has little to no programming experience. The total number of participants was 376. We excluded 48 participant's responses because of duplicate unique identifiers caused by participants incorrectly inputting his or her computer number, and participant's arriving late in lab then using another participant's computer who already finished. Therefore, the results consider 328 participants.

The participants were enrolled in CS 8, a course that introduces basic computing applications such as Microsoft Office and web applications, intended for non-computing and non-engineering majors and having no prerequisites. 98% percent of the participants were non-computing and non-engineering majors. Most UCR degree programs outside science and engineering schools require undergraduates to take any course in computing

or math as an elective; CS 8 attracts approximately 2,000 students per year (the university enrolls about 3,500 new students per year). CS 8 was chosen for the experiment due to our access to the course labs and the large number of students who take the course. The experiment was the first lab assignment for CS 8, held in the first week of the 2012 winter quarter, conducted during a scheduled three hour lab section. The experiment was conducted in twelve 3-hour lab sections, each section with 30 or less students.

The participants were blind to the conditions and purpose of the experiment. The participants were participating as part of their computing applications coursework, thus reducing self-selection bias. The experiment was approved by UCR's IRB (Institutional Review Board).

The participants completed a background information survey with questions related to technology comfort and caregiving experience. No significant correlation was found between any privacy or detection measurements and any of the background information survey. The background information results are detailed in our previous work [24].

5.3.2. DESIGN

The video privacy settings were evaluated for perceived privacy and for fall detection accuracy. A first measure was 18 videos randomly selected from 23 videos (18 rather than the full 23 due to our desire to have students spend about 30-45 minutes total on the experiment, to avoid fatigue), randomly ordered and randomly assigned video privacy setting constrained such that each participant saw all six video privacy settings three times. For each video, the participant was asked to detect what was happening and rate the level of privacy. A second measure was a ranking of each of the six video privacy

settings with respect to privacy and detection ability. A third measure was a dichotomy between sufficient and insufficient privacy for each of the six video privacy settings. The data was analyzed by comparing privacy perception and detection accuracy across video privacy settings.

5.3.3. MATERIALS

The lab sections included identically configured Windows computers and web browsers. The experiment consisted of four web pages: introduction, unique participant number, background information, and detection and privacy videos.

The introduction web page stated the purpose of the experiment to be to evaluate different ways of using video to monitor people, then listed the titles of the next three web pages. The web page instructed the participant to work alone. A hyperlink loaded a web page prompting the participant to enter a provided unique identifier, after which the background information web page loaded.

The background information web page is detailed in our previous work [24]. At the bottom of the web page was a "Submit form" button. If the participant had not selected an option for each question, then the web page would have a pop-up informing the participant that a question was missed and that the missing question's prompt would be highlighted in red. Otherwise, the detection and privacy videos web page loaded.

The detection and privacy videos web page had three sections: introduction, videos with questions, and final questions. The introduction started with an orientation to the participant's task: "Imagine your Grandpa has a camera in his home. He is concerned about falls and has asked you to occasionally monitor the camera video. The camera's

view is of Grandpa's living room." Next, the videos with questions section were outlined as follows: "Watch the following 18 videos (and introduction video). Each video lasts about 1 minute and is silent. The order of the videos is random. The videos use various styles, some with alterations intended to protect privacy." The introduction video showed each of the video privacy settings for the same ten seconds of the same original video. The introduction had a list of instructions pertaining to the answering of the 18 video's questions:

- "Watch from start to finish, seeking to detect if Grandpa has fallen. After finishing, you may rewind and rewatch all or part as many times as you'd like."

- "Answer the two questions that follow the video. Indicate that a fall has occurred if it is more likely than not to be a fall."

- "The goal is for you to detect falls without excessive awareness of non-fall activity. In judging privacy, avoid biasing your response based on your general opinion of monitoring. Instead, note that Grandpa has asked you to monitor him for falls."

- "Note: No persons were harmed in the making of these videos."

We recorded 23 videos (8 with a fall during the video, 8 with a fall prior to the video, and 7 with normal activity) without sound. The videos featured a sole, male actor, who was twenty-six years old. Screen shots of the videos are shown in our previous work [24]. We recorded the videos using a webcam that was capturing raw video at a rate of 15 frames per second. The videos were recorded in a living room. Each video was approximately one minute long. There were three categories of videos recorded: "fall during," "fall prior" and "normal activity." The "fall during" videos included stumbling

and slipping on the floor, and losing balance from the couch while reaching for a lamp. The "fall prior" videos included the actor laying on the floor in a few different angles and occasionally moving limbs. The "normal activity" videos included sweeping the floor, napping and watching television on the couch, and searching for a lost item. We converted each video into each privacy-enhanced video using our tools.

The videos with questions section of the detection and privacy videos web page randomly selected 6 "fall during", 6 "fall prior" and 6 "normal activity" videos. Each of the 18 videos had two questions. An example video with the two questions is shown in Figure 5-3. The video was embedded in the web page and had to be clicked to play. The video was above the two questions. The first question was, "This video shows that Grandpa:" and had four radio options to select in the following order:

- "falls starting at about ___ second(s) into the video and remains fallen for the rest of the video." (the ___ was a drop down menu with the options 1 through 60)

- "fell before the video started and remains fallen throughout the video."

- "is participating in normal activity (no fall occurred)."

- "I cannot determine whether a fall has occurred."

The second question was, "This style protects Grandpa's privacy" with responses in the following order: strongly disagree, disagree, slightly disagree, slightly agree, agree, and strongly agree.

Figure 5-3: Participants were instructed to watch the entire embedded video at least once then answer the two questions.

The final questions section of the privacy and detection videos web page included two ranking criteria for the ordering of the video privacy settings. A picture of each video privacy setting showed the same frame of the same original video as in Figure 5-4. The pictures could be drag-and-dropped to a new rank (left was higher) with a mouse, with

other pictures automatically shifting left or right accordingly. The starting location of each picture was randomized for each participant for each rank criterion. The first criterion was "how well each style allows you to detect a fall." Under the far left picture read, "1 (best for detecting)." Under the far right picture read, "6 (worst for detecting)." The second criterion was "how well each style protects the person's privacy." Under the far left picture read: "1 (most privacy)." Under the far right picture read: "6 (least privacy)."

The final questions section had a question for each video privacy setting: "Does this style provide sufficient privacy for Grandpa?" For each video privacy setting, there was a picture and the option to select Yes or No. The same picture was used as the ranking questions discussed previously. The order of the video privacy setting was randomized. Finally, there were three free response prompts:

- "Why do some styles provide sufficient privacy and not others?"
- "Please enter comments that you have regarding the privacy-protecting alterations."
- "Please enter any other comments you have regarding your experience."

At the bottom of the web page was a "Submit form" button. If the participant had not selected an option for each question, then the web page would have a pop-up informing the participant that a question was missed and that the missing question's prompt would be highlighted in red. Otherwise, a completion web page loaded instructing the participant to notify the teaching assistant of completion.

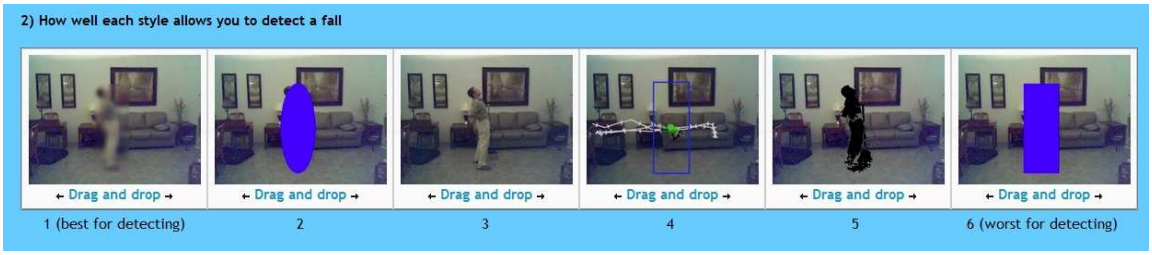


Figure 5-4: Rankings allowed participants to re-rank the video privacy setting by dragging and dropping the picture representing the video privacy setting. The initial ranking was randomized for each participant and for each ranking criteria.

5.3.4. PROCEDURE

Prior to the experiment, the 13 teaching assistants were instructed not to answer questions regarding the experiment, but to answer questions related to using the computers to answer the questions from the experiment. The teaching assistants were not participants. The teaching assistants were blind to the conditions and purpose of the experiment.

The experiment took place during lab sections. Each participant had a desktop computer assigned to him or her. Participants were given the full three hour lab section to work on the experiment. Most participants finished within the first hour, as we desired.

The lab section's teaching assistant gave participants a hyperlink to the experiment's introduction web page. The order of the experiment's web pages were static and linear in the following order: introduction, unique participant number, background information, and detection and privacy videos. Participants worked at their own pace. Participants were required to leave the lab section room once completed.

5.4 PRIVACY PERCEPTION RESULTS

We measured privacy perception in three ways: rank video privacy settings, privacy score for each video privacy setting, and asking if each video privacy setting provided sufficient privacy.

5.4.1. PRIVACY RANKING

The participants were asked to rank how well each video privacy setting protects the person's privacy. The rank from most to least privacy was: trailing-arrows, bounding-box, bounding-oval, silhouette, blur, and finally raw video. There was a statistically significant difference in ranking between all video privacy settings. The ranking is useful for establishing a relative order between the video privacy settings but does not give insight into the magnitude of the relative differences.

5.4.2. PRIVACY SCORE PER VIDEO

Each video shown (except the introduction video) had an associated prompt, "This style of video protects Grandpa's privacy" with six possible responses. We associated a number with each response: strongly disagree (-3), disagree (-2), slightly disagree (-1), slightly agree (+1), agree (+2), and strongly agree (+3). We summed the number associated with the response between each video privacy setting. There were three videos for each video privacy setting, thus the range was -9 to +9. We added +9 to each video privacy setting summation resulting in a range of 0 to +18, as shown in Figure 5-5. A score above 9 means that the video privacy setting on average was perceived by the

participants to protect privacy to some degree, whereas a score below 9 means that the video privacy setting did not protect privacy. There was a significant difference ($p < 0.005$) in the privacy scores between all video privacy settings.

The privacy score establishes the relative difference between the video privacy settings. The privacy score questions were asked immediately after watching each video and thus participants likely responded based on concrete examples rather than more abstract beliefs.

The particular combination of videos, video order, and privacy setting per video was the same across only one participant on average, which is too few to determine if a statistical confound occurred from the simultaneous manipulations of the randomly selected videos, randomly ordered videos and randomly assigned video privacy settings.

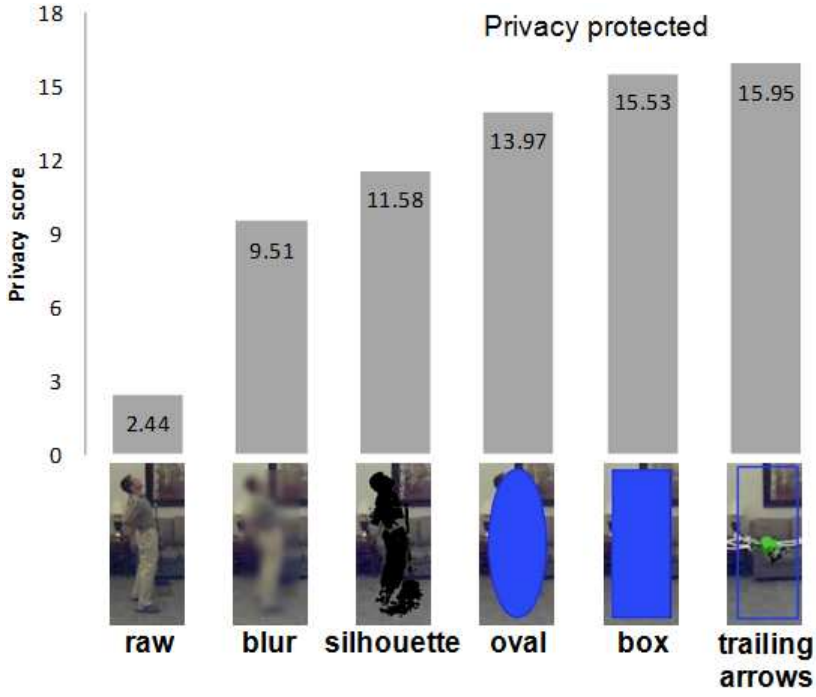


Figure 5-5: Privacy score per video: Summed privacy scores between each video privacy setting. The higher the score, the more the privacy was perceived to be protected.

5.4.3. PERCEIVED SUFFICIENCY OF PRIVACY

Ultimately, we wish to know if participants felt each privacy enhancement was sufficient in protecting privacy or not. Near the end of the experiment, participants were asked to declare each video privacy setting as providing sufficient privacy or as not providing sufficient privacy. The percentage of participants that reported each video privacy setting as sufficient is shown in Figure 5-6. There was a statistically significant difference ($p < 0.001$) in sufficiency of privacy between all video privacy settings, except the trailing-arrows video and the bounding-box video difference was not significant ($p = 0.249$).

This result suggests that only bounding-oval, bounding-box, and trailing-arrows provide sufficient privacy, and even more importantly, that the common privacy enhancements of silhouetting and blurring do not appear to provide sufficient accuracy. Additionally, since this question was administered after each video privacy setting had been seen, the participants had a chance to compare the video privacy settings. The comparison between settings is consistent with providing customizable assistant technology, which has been shown to be important to the usage of assistant technology.

The reason participants tended to report insufficient privacy for blur and silhouette (and of course raw) may have been that those video privacy settings still showed what the monitored person was doing, such as drinking coffee, scratching his or her face, etc. For example, one participant wrote in the free response at the end of the survey that, "as it [silhouette] is, you can make out what the person is doing." Another participant wrote,

"Box, oval and trailing arrows showed the action grandpa is taking but covered the appearance and shape of grandpa."

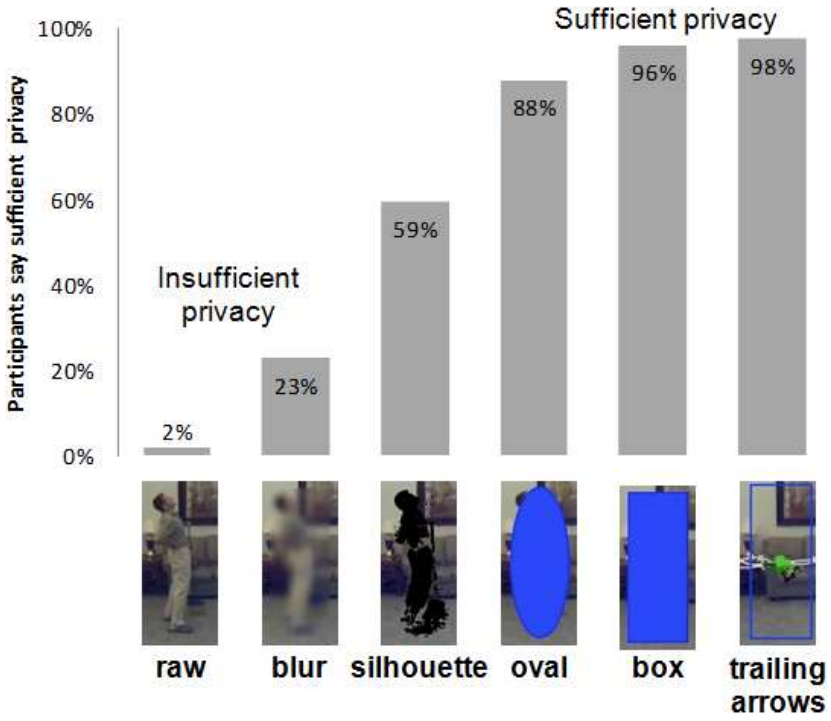


Figure 5-6: Sufficiency of privacy: The percentage of participants who reported that there was sufficient privacy provided for each video privacy setting.

5.5 FALL DETECTION RESULTS

We measured fall detection in two ways: ranking of video privacy settings based on perceived detection accuracy, and actual detection accuracy.

5.5.1. PERCEIVED FALL DETECTION ACCURACY RANKING

Participants ranked how well each video privacy setting was perceived to have allowed for him or her to detect a fall. The ranking from best to worst for detecting a fall was: raw, blur, silhouette, bounding-oval, bounding-box, and finally trailing-arrows

video. There was a significant difference ($p < 0.001$) in ranking between all video privacy settings. The perceived detection ranking is useful for establishing a relative order between video privacy settings but does not address actual detection ability nor the relative difference between video privacy settings.

5.5.2. ACTUAL FALL DETECTION ACCURACY

Each video shown (except the introduction video) had an associated prompt asking the participant to determine if a fall had occurred during the video, sometime prior to the video, or not at all. A correct answer had the correct option selected and if the correct option is a fall during the video, then the estimated time of the fall during the video is within 10 seconds of the actual fall. The detection accuracy is the number of correct answers divided by the number of questions averaged across the participants. The actual detection accuracy establishes a relative difference between video privacy settings and is a practical measure of fall detection ability.

Figure 5-7 shows the detection accuracy for each video privacy setting. There was not a significant difference ($p = 0.274$) in detection accuracy between raw and blur video, nor a significant difference ($p = 0.623$) in detection accuracy between blur and silhouette video. There was a significant difference ($p = 0.072$) in detection accuracy for raw and silhouette video. There was a significant difference ($p < 0.003$) in detection accuracy for the remaining pairs of video privacy settings.

A typical use case would be that the caregiver glances at the video and notices the care receiver's position (likely on floor) with unusually light activity, rewinds video to a period of interest (activity, then unusually light activity), then watches the interesting

point to determine whether a fall occurred. To examine the participant's ability to detect falls during the video, we categorize videos into fall-during (which would appear during a period of interest) and no-fall-during (a no-fall or pre-fall). We report the average sensitivity and specificity in Table 5-1. Sensitivity is the ratio of correctly detected fall-during videos over actual fall-during videos, e.g., if 5 fall-during videos were correctly detected but there were 6 total fall-during videos, then the sensitivity is 5/6 = 0.83. Specificity is the ratio of correctly detected no-fall-during videos over actual no-fall-during videos, e.g., if 11 no-fall-during videos were correctly detected but there were 12 total no-fall-during videos, then the specificity is 11/12 = 0.92.

The higher fall detection accuracy of bounding-oval over bounding-box may be due to the brief occasional exposures of the outer parts of a person’s head, arms, and legs with bounding-oval, which admittedly was unintentional on our part but turned out to be a useful feature. A participant explained during a post-experiment interview that such exposure helped the viewer know the monitored person’s orientation, such as which direction the person was facing. Also, one participant wrote, "I feel the oval was good because you get privacy, but you can still see the person's arms and head which can help in determining whether a person is hurt or has fallen. The bigger rectangles covering the entire body and movement make it difficult to determine this."

The particular combination of videos, video order, and privacy setting per video was the same across one participant on average, which is too few to determine if a statistical confound occurred from the simultaneous manipulations of the randomly selected videos, randomly ordered videos and randomly assigned video privacy settings.

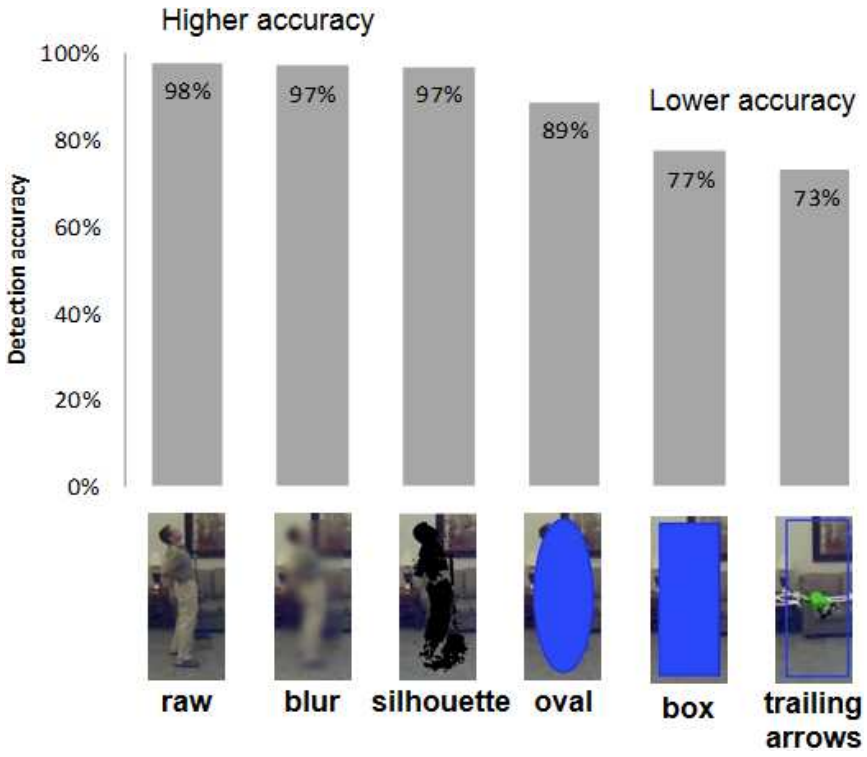


Figure 5-7: Actual fall detection accuracy: Correct answers over total answers for each video privacy setting averaged across all participants.

Video privacy setting	Sensitivity	Specificity
Raw	0.98	0.98
Blur	0.98	0.96
Silhouette	0.97	0.97
Bounding-oval	0.95	0.86
Bounding-box	0.64	0.84
Trailing-arrows	0.74	0.72

Table 5-1: Sensitivity is the correctly detected fall-during videos over total fall-during videos, and specificity is the correctly detected no-fall-during videos over total no-fall-during videos.

5.6 PRIVACY PERCEPTION AND FALL DETECTION

One purpose of our experiment was to explore the tradeoff space between fall detection and privacy perception. The tradeoff space between actual fall detection

accuracy and perceived sufficiency of privacy is shown in Figure 5-8. The ideal video privacy setting would have a value of 100% in both categories, thus being located in the top-right corner of the plot. Bounding-oval video is closest to the top-right corner, which is best assuming actual fall detection accuracy and perceived sufficiency of privacy are of equal concern. The next closest was bounding-box video, then trailing-arrows video. The significant lack of sufficient privacy provided by silhouette, blur, and raw video caused these video privacy settings to be far from the top-right corner. The tradeoff space between actual fall detection accuracy and privacy score per video had a similar distribution of video privacy settings, with bounding-oval video closest to the top right followed by bounding-box video then trailing-arrows video.

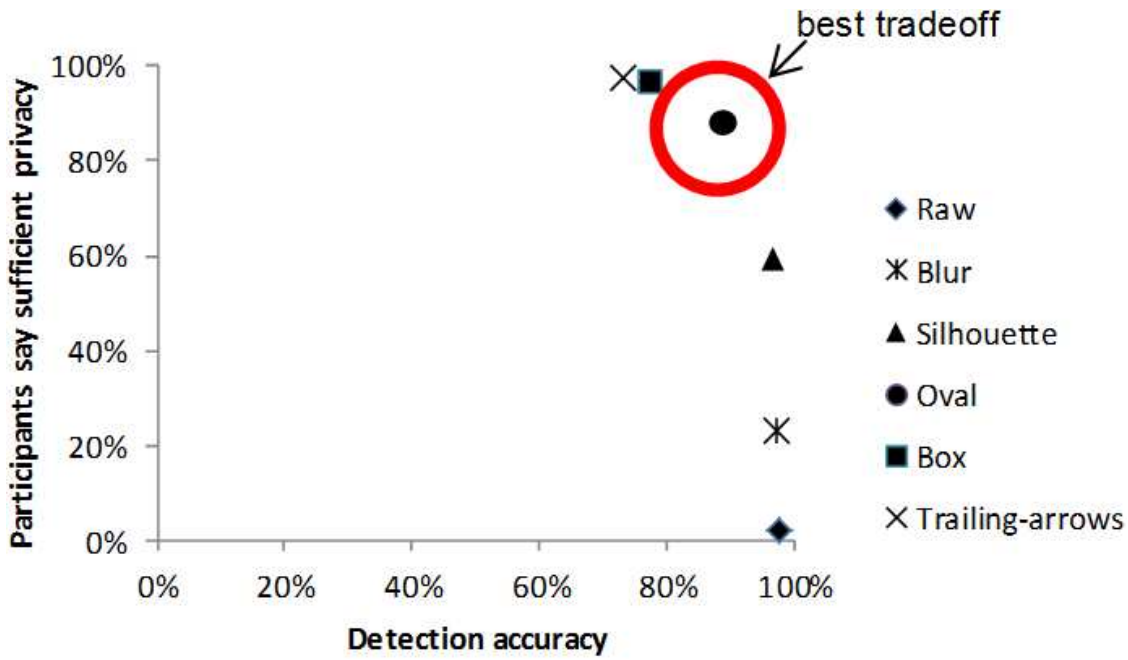


Figure 5-8: Tradeoff space for video privacy settings between actual fall detection accuracy (correct answers over total answers) and sufficiency of privacy (percentage of participants who reported privacy was sufficient).

5.7 CONCLUSION ON PRIVACY PERCEPTIONS OF PRIVACY-ENHANCED VIDEO AND FALL DETECTION ACCURACY

Our experiments demonstrated tradeoffs among privacy enhancements and fall detection accuracy for in-home assistive monitoring. In particular, placing an oval or box over a person was viewed as sufficiently protecting privacy (by 88% and 96% of participants, respectively) while still enabling observers to detect falls with good accuracy for bounding-oval (89%) and but lower accuracy for bounding-box (77%). The bounding-oval's improved accuracy (and slightly lower privacy) seemed to come from the oval occasionally and briefly not covering a person's extremities. Another key finding was that the common privacy enhancement methods of silhouetting and blurring, although commonly suggested for privacy purposes, were generally perceived as providing insufficient privacy. We also found that the trailing arrows enhancement, which we introduced to improve privacy and fall detection over a bounding box, failed to achieve improvement in fall detection. Furthermore, some users stated they found the method unpleasant or confusing to view, indicating a bias on our part towards a display comfortable perhaps for engineers.

Ultimately, an in-home assistive monitoring system may provide users with multiple privacy-enhancement options, as configurability to people's different privacy and fall detection needs is important.

Our future work includes striving to find a method similar to trailing arrows that provides strong privacy protection while also providing high fall detection accuracy and while being comfortable to view by non-engineers. We are also developing automated fall detection algorithms operating on each type of privacy enhanced video. We are investigating human and automated detection of activities beyond fall detection, including activities of daily living, such as cooking, eating, sitting, standing, nighttime rising, leave home at night, and in room too long, along with detection of disorientation, of seizures, of strangers in the home, and more.

Chapter 6. CONTRIBUTIONS

An MBR-based fall detector can provide highest accuracy while also requiring the least amount of computation. Our MBR-based algorithm used an order of magnitude less computation time than 3D-based algorithms, while being more accurate with 3 cameras and equally accurate with 4+ cameras. A single-camera achieved 0.960 sensitivity and 0.995 specificity, while 2 or more cameras achieved 0.990 or better sensitivity and specificity.

MBR-based and head-based fall detection accuracies are the same for non-confounding scenarios achieving perfect 1.00 sensitivity and specificity, and the same sensitivity of 0.75 for non-confounding scenarios. However, the 3D head-based fall detector had a specificity of 1.00, 2D head-based had 0.85, and MBR-based had 0.54 for confounding scenarios. But, head-based fall detection algorithms increased computation by five to ten times versus MBR-based fall detection, and thus are only practical when computing resources are plentiful or the highest possible accuracy is demanded.

Privacy-enhanced video degrades the accuracy of monitoring goals compared to raw video for non-adaptive algorithms. However, the adaptive algorithms, specific-color hunter and edge-void filler, can help compensate for that degradation without loosing computational efficiency. For example, fall detection accuracy degraded from 1.0 sensitivity to 0.86 and 1.0 specificity to 0.79, but the adaptive algorithms compensated accuracy back up to 0.92 sensitivity and 0.90 specificity.

Placing an oval or box over a person was viewed as sufficiently protecting privacy (by 88% and 96% of participants, respectively) while still enabling observers to detect falls with good accuracy for bounding-oval (89%) and but lower accuracy for bounding-box (77%). Another key finding was that the common privacy enhancement methods of silhouetting and blurring, although commonly suggested for privacy purposes, were generally perceived as providing insufficient privacy.

REFERENCES

[1] ActionScript. http://www.adobe.com/devnet/action script.html. June 2014.

[2] Anderson, D., J.M. Keller, M. Skubic, X. Chen, and Z. He. Recognizing Falls from Silhouettes. Proceedings of the 28th IEEE EMBS Annual International Conference. New York City, USA, 2006.

[3] Anderson, D., 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 113, no. 1 (2009): 80-89.

[4] Auvinet, E., 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 15, no. 2 (2011): 290-300.

[5] Auvinet, E., C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau. Multiple cameras fall data set. Technical Report 1350, DIRO University de Montreal, 2010.

[6] Axis Communications. http://www.axis.com/products/video/ camera/. June 2014.

[7] Beach, S., R. Schulz, K. Seelman, R. Cooper and E. Teodorski. Trade-Offs and Tipping Points in the Acceptance of Quality of Life Technologies: Results from a Survey of Manual and Power Wheelchair Users. International Symposium on Quality of Life Technology, 2011.

[8] Beach, S., R. Schulz, J. Downs, J. Mathews, B Barron and K. Seelman. Disability, Age, and Informational Privacy Attitudes in Quality of Life Technology Applications: Results from a National Web Survey. ACM Transactions on Accessible Computing. Volume 2 Issue 1, May 2009.

[9] BeClose. http://beclose.com/. June 2014.

[10] Bodson, M. and J.E. Groszkiewicz. Multivariable Adaptive Algorithms for Reconfigurable Flight Control. IEEE Transactions on Control Systems Technology, volume 5, issue 2, pgs. 217-229, 1997.

[11] Bourke, A.K., J.V. O'Brien, and G.M. Lyons. Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm. Elsevier: Gait and Posture. Volume 26, pgs. 194-199, 2007.

[12] Body Media Armbands. http://www.bodymedia.com/Shop/Armband-Packages. June 2014.

[13] Bodybugg. http://www.bodybugg.com/. June 2014.

[14] Boyle, M., C. Edwards and S. Greenberg. The Effects of Filtered Video no Awareness and Privacy. Proceedings of ACM conference on Computer Supported Cooperative Work, 2000.

[15] Caine, K.E., A.D. Fisk and W.A. Rogers. Benefits and privacy concerns of a home equipped with a visual sensing system: A perspective from older adults. Proceedings of the Human Factors and Ergonomics Society, 50th Annual Meeting (pp. 180–184), 2006.

[16] Crouter, S.E., P.L. Schneider, M. Karabulut, and D.R. Bassett JR. Validity of 10 Electronic Pedometers for Measuring Steps, Distance, and Energy Cost. Medicine and Science in Sports and Exercise. Volume 35, pgs. 1455-1460, 2003.

[17] Cucchiara, R., A. Prati and R. Vezzani. A Multi-Camera Vision System for Fall Detection and Alarm Generation. Expert Systems, Volume 24, Issue 5, pgs 334-345, November 2007.

[18] Demiris, G., M.J. Rantz, M.A. Aud, K. D. Marek, H.W. Tyrer, and M. Skubic, A.A. Hussam. Older adults' attitudes towards and perceptions of 'smart home' technologies: a pilot study. Medical Informatics and The Internet in Medicine, 2004.

[19] Dryden, I. and K. Mardia, Statistical Shape Analysis. Chichester, U.K.: Wiley, 1998.

[20] Edgcomb, A. and F. Vahid. Automated Fall Detection on Privacy-Enhanced Video. IEEE Engineering in Medicine and Biology Society, 2012.

[21] Edgcomb, A. and F. Vahid. Automated In-Home Assistive Monitoring with Privacy-Enhanced Video. IEEE International Conference on Healthcare Informatics (ICHI), 2013.

[22] Edgcomb, A. and F. Vahid. Estimating Daily Energy Expenditure from Video for Assistive Monitoring. http://static.cs.ucr.edu/store/techreports/UCR-CS-2012-09260.pdf. November 2012.

[23] Edgcomb, A and F. Vahid. MNFL: The Monitoring and Notification Flow Language for Assistive Monitoring. ACM SIGHIT International Health Informatics Symposium (IHI), 2012.

[24] Edgcomb, A. and F. Vahid. Privacy perception and fall detection accuracy for in-home video assistive monitoring with privacy enhancements. ACM SIGHIT (Special Interest Group on Health Informatics) Record, 2012.

[25] Edgcomb, A. and F. Vahid, Video-based fall detection dataset with 2D and 3D head tracking, and moving-region tracking. http://www.cs.ucr.edu/~aedgcomb/3D_2D_head_an_MBR_videos.html. June 2014.

[26] Energy Expenditure Estimation Video Dataset. http://www.cs.ucr.edu/~aedgcomb/videoBasedEnergy Estimate.html. June 2014.

[27] Fall Video Dataset. http://www.cs.ucr.edu/~aedgcomb/fallVideos.html. June 2014.

[28] Gator Tech Smart House - Smart Floor. http://www.icta.ufl.edu/pervasive-applications.htm. June 2014.

[29] Guo, Z., M. Zhuo, and G. Jiang. Adaptive sensor placement and boundary estimation for monitoring mass objects. IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 38, no. 1, pp. 222–232, Feb. 2008.

[30] Helal, S., W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen. The Gator Tech Smart House: A Programmable Pervasive Space. IEEE Computer. March 2005.

[31] Hung, D.H. and H. Saito. Fall Detection with Two Cameras based on Occupied Area. In Proc. of 18th Japan-Korea Joint Workshop on Frontier in Computer Vision, pp. 33-39. 2012.

[32] Hung, D.H. and H. Saito. The Estimation of Heights and Occupied Areas of Humans from Two Orthogonal Views for Fall Detection. IEEJ Trans. EIS 133, no. 1, 2013.

[33] Jain, R., R. Kasturi, and B.G. Schunck. Machine Vision. Published by McGraw-Hill, Inc., ISBN 0-07-032018-7, 1995.

[34] Jakicic, J.M, M. Marcus, K.I. Gallagher, C. Randall, E. Thomas, F.L. Goss, and R.J. Robertson. Evaluation of the SenseWear Pro Armband to Assess Energy Expenditure during Exercise. Medicine and Science in Sports and Medicine. Volume 36, pgs. 897-904, 2004.

[35] Kangas, M., A. Konttila, P. Lindgren, I. Winblad, and T. Jämsä. Comparison of Low-Complexity Fall Detection Algorithms for Body Attached Accelerometers. Gait and Posture 28, no. 2, 2008.

[36] Kim, K., T.H. Chalidabhongse, D. Harwood and L. Davis. Real-Time Foreground-Background Segmentation using Codebook Model. Real-Time Imaging, Volume 11, Issue 3, June 2005, pgs. 172-185.

[37] Lee, A., A. Girgensohn and K. Schlueter. NYNEX Portholes: Initial User Reactions and Redesign Implications. Proceedings of ACM SIGGROUP, 1997.

[38] Liu, L., M. Popescu, M. Skubic, T. Yardibi, AND P. Cuddihy. Automatic Fall Detection Based on Doppler Radar Motion Signature. 5th International Conference on Pervasive Computing Technologies for Healthcare and Workshops, 2011.

[39] Miaou, S.-G., P.-H. Sung and C.-Y. Huang. A Customized Human Fall Detection System Using Omni-Camera Images and Personal Information. Proceedings of the 1st Distributed Diagnosis and Home Healthcare Conference, 2006.

[40] MNFL Video Dataset. http://www.cs.ucr.edu/~aedgcomb/MNFLevents.html. June 2014.

[41] Motorola Homesight. http://www.smarthomeusa.com/Products/Homesight-USB-Kit/manuals/homesight-software-user-guide.pdf. June 2014.

[42] Neustaedter, C., S. Greenberg and M. Boyle. Blur filtration fails to preserve privacy for home-based video conferencing. ACM Transactions on Computer Human Interactions (TOCHI), vol. 13, no. 1, pp. 1–36, 2005.

[43] OpenCV. http://opencv.org/. June 2014.

[44] Petersson, O. and A. Moffat. A Framework for Adaptive Sorting. Lecture Notes in Computer Science, 3rd Scandinavian Workshop on Algorithm Theory, 1992.

[45] Quietcare. http://www.careinnovations.com/Products/QuietCare/. June 2014.

[46] Rantz, M.J., M.A. Aud, G. Alexander, B.J. Wakefield, M. Skubic, R.H. Luke, D. Anderson, and J.M. Keller. Falls, technology, and stunt actors: New approaches to fall detection and fall risk assessment. Journal of nursing care quality 23, no. 3, 2008.

[47] Rashidi, P. and D.J. Cook. Keeping the Resident in the Loop: Adapting the Smart Home to the User. IEEE Transactions on Systems, Man and cybernetics, Part A: Systems and Humans, 2009.

[48] Robinovitch, S.N., F. Feldman, Y. Yang, R. Schonnop, P.M. Lueng, T. Sarraf, J. Sims-Gould, and M. Loughin. Video capture of the circumstances of falls in elderly people residing in long-term care: an observational study. The Lancet, 2012.

[49] Rougier, C., J. Meunier, A. St-Arnaud, and J. Rousseau. 3D Head Tracking for Fall Detection using a Single Calibrated Camera. Image and Vision Computing, 2012.

[50] Rougier, C., J. Meunier, A. St-Arnaud, and J. Rousseau. Fall Detection from Human Shape and Motion History using Video Surveillance. 21st International Conference on Advanced Information networking and Applications Workshops, 2007.

[51] Rougier, C., J. Meunier, A. St-Arnaud, and J. Rousseau. Procrustes shape analysis for fall detection. In The Eighth International Workshop on Visual Surveillance-VS2008, 2008.

[52] Rougier, C., J. Meunier, A. St-Arnaud, and J. Rousseau. Robust video surveillance for fall detection based on human shape deformation. Circuits and Systems for Video Technology, IEEE Transactions on 21, no. 5 (2011): 611-622.

[53] Schoeller, D.A., E. Ravussin, Y. Schutz, K.J. Acheson, P. Baertschi, and E. Jequier. Energy Expenditure by Doubly Labeled Water: American Journal of Physiology: Regulatory, Integrative, and Comparative Physiology. Volume 240, pgs. 823-830, 1986.

[54] Smarthome. http://www.smarthome.com/. June 2014.

[55] Sotoma, I. and E.R.M. Madeira. Adaptation - Algorithms To Adapt Fault Monitoring And Their Implementation On CORBA. Proceedings. 3rd International Symposium on Distributed Objects and Applications, pgs 219-228, 2001.

[56] Stauffer, C. and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. IEEE International Conference on Computer Vision and Pattern Recognition 1999; 2 : pg. 246–52.

[57] Thome, N., S. Miguet and S. Ambellouis. A Real-Time, Multiview Fall Detection System: A LHMM-Based Approach. IEEE Transactions on Circuits and Systesm for Video Technology, Vol. 18, No. 11, November 2008.

[58] Williams, A., D. Ganesan and A. Hanson. Aging in Place: Fall Detection and Localization in a Distributed Smart Camera Network. Proceedings of the 15th International Conference on Multimedia, 2007.

[59] Wood, A., G. Virone, T. Doan, Q. Cao, L. Selavo, Y. Wu, L. Fang, Z. He, S. Lin, and J. Stankovic. ALARM-NET: Wireless Sensor Networks for Assistive-Living and Residential Monitoring. University of Virginia Computer Science Department Technical Report, 2006.

[60] X10. http://www.x10.com. June 2014.

[61] Yao, N., R.J. Sclabassi, Q. Liu, J. Yang, J.D. Fernstrom, M.H. Fernstrom, and M. Sum. A Video Processing Approach to the Study of Obesity. Multimedia and Expo. Pgs. 1727-1730, 2007.

[62] Zhao, Q. A. and J. T. Stasko. Evaluating Image Filtering Based Techniques in Media Space Applications. Proceedings of ACM conference on Computer Supported Cooperative Work, 1998.

[63] Zivkovic, Z. Improved adaptive Gaussian mixture model for background subtraction. Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on. Vol. 2. IEEE, 2004.