

REAL TIME ACTIVITY MONITORING



by
Gamze Uslu

Submitted to the Institute of Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science
in
Computer Engineering

Yeditepe University
2013

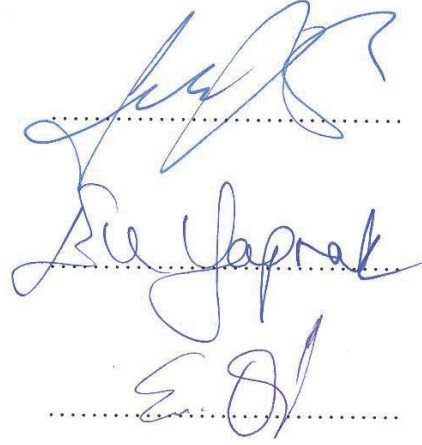
REAL TIME ACTIVITY MONITORING

APPROVED BY:

Prof. Dr. Şebnem Baydere
(Thesis Supervisor)

Prof. Dr. Ece Yaprak

Assist. Prof. Dr.Esin Onbaşıođlu



The image shows three handwritten signatures in blue ink, each written over a horizontal dotted line. The top signature is for Prof. Dr. Şebnem Baydere, the middle one is for Prof. Dr. Ece Yaprak, and the bottom one is for Assist. Prof. Dr. Esin Onbaşıođlu.

DATE OF APPROVAL:/..../2013

ACKNOWLEDGEMENTS

It is with immense gratitude that I acknowledge the support and help of my Professor Şebnem Baydere. Pursuing my thesis under her supervision has been an experience which broadens the mind and presents an unlimited source of learning.

I thank Bora Baydere, Research Assistants Özgür Altun, İsmail Uğur Bayındır and Erdiñç Bakır, Wireless Network Lab's student members Erman Gönül, Orhan Can, İlbey Kurt and Baturay Özcan, also Merve Bıçakçı for simplifying the process of sample collection, though it was a very time consuming process. In addition, I would like to express my appreciation to Research Assistant Erdem Çakır for helping me organize the testing environment.

Finally, I would like to thank my family for their endless love and support, which makes everything more beautiful.

ABSTRACT

REAL TIME ACTIVITY MONITORING

Activity monitoring systems (AMS) are responsible for detecting actions performed by humans. For AMS to be effectively deployed in daily life, they should be operating in real-time and partition the continuously streaming activity data to determine what activity corresponds to each partition. In this thesis, a Support Vector Machine based real time continuous activity monitoring system, named RT-CAM, is proposed. We approach continuous activity detection problem by modelling the activities as simple and composite actions. Simple actions being the smallest meaningful actions which can not be further divided into smaller logical actions whereas composite actions are combinations of simple actions. The proposed model detects simple and composite activities in real time, collecting the data with a single 3D accelerometer to produce a non-invasive solution. We verified our model on hand oriented set of simple actions *eat*, *pour*, *drink*, *toothBrush* and *turnKey*, with real data acquired from human subjects instead of computer generated synthetic data. We showed that the selected activities can be distinguished in real time though they generate quite similar patterns to each other. The strength and novelty of the proposed model lies in the fact that the system does not necessitate being trained with patterns of transitions and does not run a dedicated algorithm for transition detection. We carried out experiments on 4 different subjects and present our best achieved results. Intra-person test results are the following: *ToothBrush*, *drink*, *drink_toothBrush*, *toothBrush_drink* and *toothBrush_pour* are recognized with 100% accuracy. *Drink_toothBrush_pour* and *toothBrush_drink_pour_turnKey* are detected with 80% and 70% accuracy respectively. Inter-person test results are the following: *ToothBrush*, *drink* and *pour* are detected with 100% accuracy. *Drink_toothBrush*, *drink_toothBrush_pour* and *drink_toothBrush_turnKey* are recognized with 80% accuracy. Real time overhead introduced by RT-CAM is 0.055 seconds, which is better than best achieved result in the literature. Considering all these features, RT-CAM is an applicable solution in real time continuous activity monitoring.

ÖZET

GERÇEK ZAMANLI AKTİVİTE TAKİBİ

Aktivite takip sistemleri (ATS), kişilerin aktivitelerinin tespitinden sorumludur. ATS'nin etkinliği, gerçek zamanlı çalışabilmesine ve kesintisiz şekilde akan aktivite verisini bölümlendirerek her bölümün hangi aktiviteye karşılık geldiğini belirleyebilmesine bağlıdır. Bu tez kapsamında, RT-CAM olarak adlandırılan, Destek Vektör Makinaları tabanlı bir gerçek zamanlı kesintisiz aktivite takibi sistemi önerilmiştir. Aktiviteler, basit ve bileşik hareketler olarak modellenmekte; basit hareketler, anlamlı daha küçük hareketlere bölünemeyen aktiviteler olmak üzere, bileşik hareketler basit hareketlerin kombinasyonlarından oluşmaktadır. Önerdiğimiz model, basit ve bileşik hareketleri gerçek zamanlı olarak tespit etmekte, günlük hayata müdahale etmeyen bir çözüm sunmak için verileri tek bir 3B ivmeölçer ile toplamaktadır. Yöntemimizi, elin baskın olduğu basit hareketler olan *ye*, *dök*, *iç*, *dişFırçala* ve *anahtarÇevir* aktivitelerinden oluşan veri kümesi üzerinde, bilgisayar tarafından üretilmiş yapay verilerle değil, insan deneklerle yapılmış gerçek testlerle doğruladık. Seçilmiş aktivitelerin, birbirine oldukça benzer modeller ürettiği halde, gerçek zamanlı olarak ayırt edilebildiğini gösterdik. Önerdiğimiz yöntemin gücü ve yeniliği, sistemimizin, hareketler arası geçişlerin modellerini tutmayı gerektirmemesinden ve bu geçişlerin tespitine tahsis edilmiş bir algoritma çalıştırmamasından kaynaklanmaktadır. 4 farklı denek üzerinde yaptığımız deneylerde elde ettiğimiz en başarılı sonuçlar aşağıda belirtilmektedir: İnter-kişisel testlerde, *dişFırçala*, *iç*, *iç-dişFırçala*, *dişFırçala-iç* ve *dişFırçala-dök* 100% başarıyla; *iç-dişFırçala-dök* ve *dişFırçala-iç-dök_anahtarÇevir* ise sırasıyla 80% ve 70% başarıyla yakalanmıştır. İnter-kişisel testlerde, *dişFırçala*, *iç* ve *dök* 100% başarıyla, *iç-dişFırçala*, *iç-dişFırçala-dök* ve *iç-dişFırçala_anahtarÇevir* ise 80% başarıyla tespit edilmiştir. RT-CAM'in gerçek zamanlı işleme getirdiği ek yük 0.055 saniyedir ve bu sonuç literatürde ulaşılmış en iyi sonuçtan daha başarılıdır. Tüm bu özellikleri göz önünde bulundurulduğunda, RT-CAM, gerçek zamanlı kesintisiz aktivite takibinde uygulanabilecek bir çözümdür.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	III
ABSTRACT.....	IV
ÖZET	V
LIST OF FIGURES	IX
LIST OF TABLES.....	XII
LIST OF SYMBOLS/ABBREVIATIONS.....	XVI
1. INTRODUCTION.....	1
1.1. Why Design AMS.....	1
1.2. Phases of Activity Monitoring Systems.....	2
1.3. Parameters Affecting System Design.....	2
1.4. Problem Definition.....	4
1.5. Motivation & Aim.....	4
1.6. Overview of the Proposed Model	5
1.6.1. Contribution.....	7
1.7. Organization of the Thesis	8
2. RELATED WORK.....	9
2.1. Data Acquisition.....	9
2.1.1. Wearable Sensors.....	9
2.1.2. Environmental Sensors	16
2.2. Knowledge Discovery	16
2.2.1. Feature Extraction.....	17
2.2.2. Classification	20
2.2.3. SVM.....	21
2.3. Continuous Real Time Monitoring	23
2.4. Transition Detection.....	25
2.5. Segmentation.....	26
2.6. Approaches Applying A Dedicated Algorithm For Detecting Transitions.....	26
2.7. Summary	27
3. SUPPORT VECTOR MACHINES.....	28
3.1. Summary	35

4.	REAL TIME CONTINUOUS MONITORING	36
4.1.	Knowledge Discovery	36
4.1.1.	Segmentation	36
4.1.2.	Adaptation of SVM.....	36
4.2.	Summary	41
5.	EXPERIMENTAL SETUP	42
5.1.	Ambient Assisted Living Scenario.....	42
5.2.	Sensor Data Analysis	43
5.2.1.	Data Collection and Processing Tools	43
5.2.2.	Data Set Generation	43
5.3.	Real Testbed Environment.....	46
5.3.1.	System Model	47
5.3.2.	Methodology	47
5.3.3.	Testbed Setup.....	48
5.3.4.	Examination.....	51
5.4.	Discussion	51
5.5.	Summary	53
6.	PERFORMANCE ANALYSIS & EVALUATION.....	55
6.1.	T_iP_i Detection Accuracy	55
6.1.1.	Simple Action Accuracy (SA)	56
6.1.2.	Composite Action Accuracy (CA).....	56
6.2.	T_iP_j Detection Accuracy	58
6.2.1.	Simple Action Accuracy (SA)	58
6.2.1.1.	Discussion.....	58
6.2.2.	Composite Action Accuracy (CA).....	60
6.2.2.1.	Discussion.....	61
6.3.	Real Time Overhead	61
6.4.	Comparison of RT-CAM	62
6.5.	Summary	63
7.	CONCLUSION AND FUTURE WORK.....	64
	REFERENCES	66
	APPENDIX A: NUMERICAL EXAMPLE FOR RT-CAM _{KD}	77
	APPENDIX B: DETECTED ACTIVITIES IN T_iP_i TESTS	79

APPENDIX C: REAL TIME OVERHEAD IN T₁P₁ TESTS90



LIST OF FIGURES

Figure 1.1. System architecture of RT-CAM	7
Figure 2.1. Sensor located on the chest for evaluating VESPA [23].....	10
Figure 2.2. Motion analysis equipment used in the study of Leonard et. al. [26]	11
Figure 2.3. Accelerometry based proprietary activity monitors explained in the work by Godfrey et. al. [30].....	15
Figure 2.4. eWatch device used in the work by Maurer et. al. [33].....	18
Figure 2.5. The SIMS used in the work by Zhang et. al. [44]	21
Figure 2.6. Sensing module used in the work by Sazonov et. al. [5].....	23
Figure 3.1. SVM architecture	29
Figure 3.2. Elements in sets P (in blue) and N (in red).....	29
Figure 3.3. Three support vectors marked in yellow	30
Figure 3.4. The discriminating hyperplane	31
Figure 3.5. Set of data points P (in blue) and N (in red).....	32
Figure 3.6. Feature space representation of data points.....	32
Figure 3.7. Support vectors in feature space (in yellow).....	33
Figure 3.8. Discriminating hyperplane calculated for nonlinear example	34

Figure 5.1. Sensing device.....	43
Figure 5.2. Raw sensor data for toothBrushing and drinking.....	44
Figure 5.3. Raw sensor data for turningKey and pouring.....	45
Figure 5.4. Raw sensor data for eating	46
Figure 5.5. Histogram modelling the data set from which f is derived	47
Figure 5.6. Testbed overview	48
Figure 5.7. Auxiliary objects used in performing the activities.....	48
Figure 5.8. Drinking action as the combination of (a), (b), (c) and (d) with given order....	49
Figure 5.9. Pouring action.....	49
Figure 5.10. Eating action as the combination of (a), (b), (c), (d) and (e) with given order	50
Figure 5.11. TurningKey action.....	50
Figure 5.12. ToothBrushing action	51
Figure 5.13. Training output for toothBrushing and drinking	52
Figure 5.14. Training output for turningKey and pouring	53
Figure 5.15. Training output for eating.....	54
Figure 6.1. Real time overhead for simple actions	61

Figure 6.2. Real time overhead for composite actions62



LIST OF TABLES

Table 2.1. Results of VESPA [23] in the form of (mean \pm standard deviation).....	11
Table 2.2. Results related to the study of Karantonis et. al. [28].....	12
Table 2.3. Best achieved results in the work of Lyons et. al. [29].....	13
Table 2.4. Properties of commercial accelerometer based activity monitors listed in the work by Godfrey et. al. [30].....	14
Table 2.5. Sensors and their functionality in the work by Kasteren et. al. [10]	16
Table 2.6. SVM classification accuracy (%) with different kernels in the work by Sazonov et. al. [5].....	22
Table 2.7. Accuracy of transition detection for various transition periods.....	24
Table 2.8. Effect of shrinking and expansion on recognition accuracy in the work by Okeyo et. al. [63].....	25
Table 6.1. Real time classification accuracy of simple actions (T_iP_i)	56
Table 6.2. Recognition accuracy of composite actions (T_iP_i).....	57
Table 6.3. Recognition accuracy of simple actions (T_iP_j)	59
Table 6.4. Composite action accuracy results (T_iP_j).....	60
Table 6.5. Comparison of RT-CAM with the work by Aiello et. al. [62]	63
Table B.1. Drinking	79

Table B.2. Drinking_pouring.....	80
Table B.3. Drinking_toothBrushing	80
Table B.4. Drinking_toothBrushing_pouring.....	81
Table B.5. Drinking_toothBrushing_pouring_turningKey.....	81
Table B.6. Drinking_toothBrushing_turningKey	82
Table B.7. Drinking_toothBrushing_turningKey_pouring.....	82
Table B.8. Drinking_turningKey	83
Table B.9. Pouring	83
Table B.10. Pouring_turningKey.....	84
Table B.11. ToothBrushing_drinking.....	84
Table B.12. ToothBrushing_drinking_pouring	85
Table B.13. ToothBrushing_drinking_pouring_turningKey	85
Table B.14. ToothBrushing_drinking_turningKey.....	86
Table B.15. ToothBrushing_drinking_turningKey_pouring	86
Table B.16. ToothBrushing	87
Table B.17. ToothBrushing_pouring.....	87
Table B.18. ToothBrushing_turningKey	88

Table B.19. TurningKey	88
Table B.20. TurningKey_pouring.....	89
Table C.1. Drinking	90
Table C.2. Drinking_pouring.....	91
Table C.3. Drinking_toothBrushing	91
Table C.4. Drinking_toothBrushing_pouring.....	92
Table C.5. Drinking_toothBrushing_pouring_turningKey.....	92
Table C.6. Drinking_toothBrushing_turningKey	93
Table C.7. Drinking_toothBrushing_turningKey_pouring.....	93
Table C.8. Drinking_turningKey	94
Table C.9. Pouring.....	94
Table C.10. Pouring_turningKey.....	95
Table C.11. ToothBrushing_drinking.....	95
Table C.12. ToothBrushing_drinking_pouring	96
Table C.13. ToothBrushing_drinking_pouring_turningKey	96
Table C.14. ToothBrushing_drinking_turningKey.....	97
Table C.15. ToothBrushing_drinking_turningKey_pouring	97

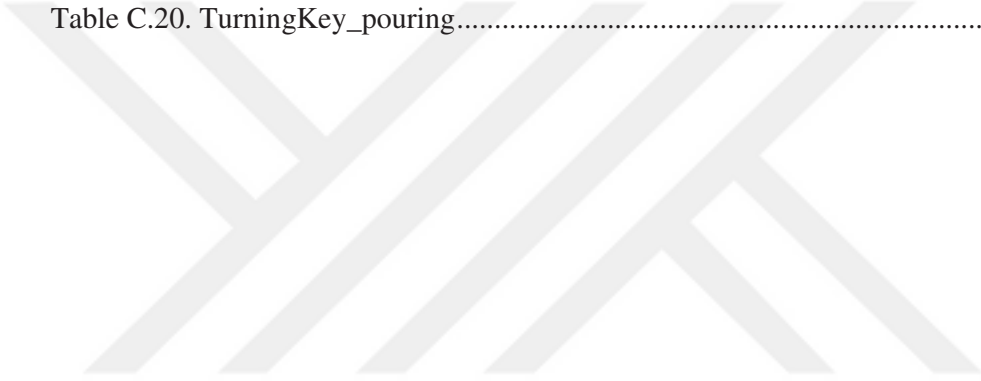
Table C.16. ToothBrushing 98

Table C.17. ToothBrushing_pouring 98

Table C.18. ToothBrushing_turningKey 99

Table C.19. TurningKey 99

Table C.20. TurningKey_pouring 100



LIST OF SYMBOLS/ABBREVIATIONS

AMS	Activity monitoring systems
ANN	Artificial neural networks
BT-SVM	Binary Tree Integrated SVM (BT-SVM)
RT-CAM	Real time continuous activity monitoring model
SVM	Support Vector Machines
WSN	Wireless sensor network

1. INTRODUCTION

Elderly and disabled people need assistance of other people in the course of their daily routines. Assistive technology emerges as a way of substituting the assisting people so that elderly and disabled people can live without relying on other people's assistance. Assistive technology aids people by means of ambient care systems. Featured by the ability of comprehending the people's needs and generating responses accordingly, ambient care systems have to detect what action is being performed by a person at first. Activity Monitoring Systems (AMS) are the components which are responsible for action detection in ambient care systems. Though health care is the predominant field where AMS is used, there are also other application areas such as security [1] and process control.

1.1. WHY DESIGN AMS

Activities performed by people contain a rich spectrum of information about how to improve or fix problematic situations. Utilizing activity information to come up with such improvements can be exemplified as follows: Fall detection [2] can be used in determining a person is *about to fall* and reacting in a fall preventing manner. Chronic disease management [3,4], rehabilitation systems [5], disease prevention [6,7], monitoring of health status [8] are some of the application areas of activity monitoring in the field of health care. Another aspect where activity monitoring is extensively useful is process control. Monitoring activities of people responsible for implementing procedures in production facilities, such as water purification plants, helps output quality increase since production steps are ensured to be fulfilled as planned. Also in bomb disposal missions [1], performance of cooling systems integrated to operators' suits is highly effected by the operators' posture which raises monitoring activity as one of the essentials in security applications. Increasing performance of sportsmen by means of injure prevention [9] is another domain showing the usefulness of activity monitoring for young and healthy part of the population.

1.2. PHASES OF ACTIVITY MONITORING SYSTEMS

Activity monitoring systems are composed of data collection and data classification techniques. Discriminating characteristics of the collected data are determined by feature extraction methods. To find out which features are more discriminating, feature selection methods are employed. Selected features are processed with classification techniques and type of activity is generated.

Sensors utilized in data collection phase can be grouped as environmental and wearable. Environmental sensors [10-15] are placed on objects and can detect complex activities thanks to interaction between these objects and people. Nevertheless, they are insufficient in detecting activities performed without interacting with the objects equipped with sensors. Cameras [16-19] are also environmental sensors but they are regarded as an element intervening with the daily life, thus people tend to reject being monitored by AMS with cameras. Wearable sensors are preferred because of reasons such as not requiring interaction with objects, not interfering with daily life and not limiting people's actions. Accelerometer and gyroscope [20-30] are commonly used wearable sensors. Though wearable smart systems are frequently utilized in activity monitoring, most of them can not go beyond prototyped structures and studies are needed to determine problems which will arise as a result of using these prototypes in daily basis. For this reason, more research should be done in the fields such as smart signal processing, interoperability of communication standards, efficiency of electronic components and energy [31]. Statistical features [32-34], wavelet coefficients [35-37] and custom coefficients [38,39] are examples of features extracted. Central and independent component analysis [40], forward-backward selection [41], correlation [33] are methods used in feature selection. Artificial Neural Networks [42-44], Support Vector Machines [5] [45,46], Bayesian classifiers [47] and Hidden Markov Models [6,48] can be mentioned as classification procedures.

1.3. PARAMETERS AFFECTING SYSTEM DESIGN

AMS are built considering a number of parameters which effect performance, interoperability and usability.

- *Invasiveness/obtrusiveness*: Some data collection tools incorporated in AMS tend to limit the actions of people. Such a restriction influences the decision of a person on using the design. Consequently, sensors are categorized in terms of invasiveness and obtrusiveness. Cameras are regarded as invasive data collection devices since they limit the person's actions by interfering with the privacy. Wearable sensors such as accelerometers emerge as non-invasive alternatives to cameras. A data collecting tool which is invisible to user is characterized as unobtrusive such as sensors embedded within environment. In terms of being comfortably used by subjects, unobtrusive sensors may seem to be superior to wearable sensors but they have to be placed anywhere the subject may be which is another infeasibility justifying use of wearable sensors.
- *Heterogeneity of sensor types*: Data collecting tools can also be a WSN (wireless sensor network) of heterogeneous sensors types. This leads to deciding on whether data classification had better be carried out in distributed (in-network) or centralized (on gateway) fashion. Data collecting tool should also satisfy interoperability since integrating AMS to existing WSNs is preferable to establishing a new WSN specific to AMS. Incorporating AMS to available WSNs raises other issues like service differentiation [49].
- *Number of sensors and sensor location*: Number of sensors and sensor location can impact classification accuracy however, sensors placed in excessive amounts or uncomfortable locations may challenge people's movements.
- *Activity set*: Activity set to recognize are affected by the subject profile/context and application scenarios.
- *Spatiality*: Spatiality of AMS deployment should be taken into consideration as well. Indoor and outdoor monitoring can differ in terms of technologies to use since some technologies are not operable in both indoors and outdoors such as GPS. Also, layout of objects can vary in different environmental settings which results in deviations between the way same activity is performed.
- *Variance in activity practice*: Another concern making AMS design intricate is that same person can perform the same activity differently even under same deployment conditions in terms of spatiality.
- *Uncertainty of sensor data*: Uncertainty of sensor data is another source of unstable data, which enforces loss resilient feature extraction and classification techniques.

- *Complexity of the activities:* Complexity of activities may require employing multiple types of sensors, making uncertain sensor data a more challenging problem.
- *Availability of data set:* Having to acquire excessive amount of training and test data is another point which makes AMS system design challenging.

1.4. PROBLEM DEFINITION

Activity monitoring is mapping the data representing the activity signal such as acceleration, pressure, etc. to classes of activity such as walking, sitting, lying and so on. Activity data are acquired using sensors and sensor data are exposed to knowledge discovery methods, which are feature extraction and data classification, in order to obtain activity classes. Real time continuous activity monitoring is a subset of the activity monitoring problem. Real time monitoring introduces more complexity than off-line analysis since sensor data acquisition and data classification should be concurrent, which results in data loss and real time delay. Similarly, continuous monitoring is harder to handle compared to monitoring individual actions. Continuous monitoring is the recognition of the sequence of individual actions between which activity transitions exist. Because continuous data contains both individual actions and transitions, the start and end point of the actions have to be found within the continuous data, which introduces segmentation problem.

Mapping the sensor data to activity classes accurately is not enough when real time continuous monitoring is considered to be deployed in real life conditions. The system should respond in a reasonable amount of time, therefore a perfectly accurate classification technique which deteriorates responsiveness is not preferred. Also, the system should be acceptable by users, hence should not incorporate sensors enforcing inconveniences for the user.

1.5. MOTIVATION & AIM

Activity monitoring systems are widely studied but in terms of real time continuous monitoring, still several problems exist in the literature. These problems can be summarized as the following:

- *Segmenting the continuous data*: Activity data contain sequences of actions which makes determining start and end of actions controversial.
- *Training the system with patterns of transitions*: Methods following this approach are not suitable for extending the set of actions, damaging scalability and restricting the system for being used in specific cases.
- *Necessitating large amount of training data*: Acquiring training data is a very challenging part of AMS, therefore it is necessary to be able to generate patterns with minimum amount of training data.
- *Necessitating person-specific training data*: Monitoring activities of different people upon training the system with only one subject greatly simplifies deployment of AMS.
- *Real time prediction delay*: Since ultimate goal of AMS is operating in real-time, they should respond as quickly as the application domain necessitates.

Considering the criteria mentioned above, this dissertation aims to produce an applicable model for real time continuous activity monitoring based on 3D acceleration data from one sensor on the body.

1.6. OVERVIEW OF THE PROPOSED MODEL

Since activity data are acquired from sensors, activities are represented as row sensor data in the beginning of activity monitoring procedures. To extract the action sequences and map raw sensor data to activity labels, actions should be modelled. In this study, activities are modelled as simple and composite actions. Activities which can not be divided into meaningful smaller actions are called simple actions whereas combinations of simple actions are evaluated as composite actions.

The ultimate goal of the proposed AMS is marking simple actions incorporated in a given composite action in real time since all other AMS requirements are based on continuously monitoring people. For this reason, a composite action stores two types of knowledge to be extracted: Constituting simple actions and transitions between those simple actions, splitting the activity monitoring problem to two subproblems.

Our previous works [50] and [51] present simple and composite activity detection schemes we proposed prior to this thesis. In the work published in [50], the simple and composite action model is developed and experimented, classifying *walk*, *sit*, *lie* and *stand* activities with 92%, 100%, 88%, 96% accuracy respectively, data collection tool being a single 3D acceleration sensor. In our other work [51], our model is enhanced incorporating two sensors, each of which contains a 3D accelerometer, and classifying with a combination of classifiers to evaluate real time continuous activity monitoring success. Naïve Bayes, Susan Corner Detector (SCD) and Hidden Markov Model (HMM) are combined to form our hybrid classification module. Our method containing multiple sensors and multiple classifiers identify simple actions *walk*, *walk while hands in pocket*, *sit*, *stand* and *wheelchair driving* with 94%, 96%, 94%, 94%, 98% accuracy respectively. Though this model identifies transitions yielding 100% accuracy, due to real time processing delay, detection success rate of individual actions deteriorate, hence in this thesis we develop a real time continuous activity monitoring scheme, which we call RT-CAM. RT-CAM, which stands for real time continuous activity monitoring system, is a solution to identify the composite actions in real-time. Figure 1.1 illustrates the system architecture of RT-CAM. Operation of RT-CAM is initiated by sensor transmitting continuous activity signal, which is composed of cs segments, to the DA (Data acquisition) unit. Each segment, lasting for the period t , is forwarded from DA unit to SAD (Simple action detection) module. Then, the segment is processed by SAD for being labelled as one of the simple actions (SA) in the action set after knowledge extraction process is applied. We intend to address the following features with RT-CAM:

- Transition detection is achieved without training the system with patterns of transitions. Transition detection model does not necessitate combining features either.
- Presented results are achieved with real data acquired from human subjects instead of computer generated synthetic data, resulting in a more realistic proof of model.
- RT-CAM utilizes a single 3D accelerometer which makes it a non-invasive solution.

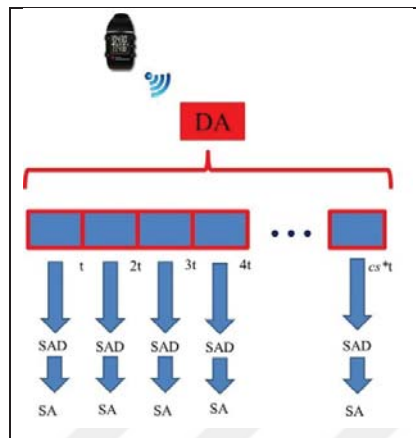


Figure 1.1. System architecture of RT-CAM

1.6.1. Contribution

Considering the problems highlighted in Chapter 1.5, the following contributions are made: We develop a non-invasive solution which we call RT-CAM, pursuing real time continuous monitoring of hand oriented activities. A preliminary study of this work is accepted as a publication [52]. RT-CAM does not require generating combinations of features and carries out transition detection without being trained with patterns of transitions. RT-CAM is evaluated regarding the following criteria:

- Intra-person and inter-person classification accuracy
- Successful recognition rate of simple actions
- Successful recognition rate of composite actions as a whole
- Real time delay

According to our findings, our model RT-CAM is promising for real time continuous monitoring of hand oriented activities. Also, it does not necessitate large number of subjects who should provide training data. With regard to the features of RT-CAM mentioned, the contribution of the thesis can be summarized as follows:

- Designing a tool which can detect transitions
- Recognizing hand based activities with high detection rates

- Distinguishing the selected activities in real time though they are similar to each other

1.7. ORGANIZATION OF THE THESIS

The rest of the thesis is organized as follows: Chapter 2 presents the related work, categorizing the activity monitoring studies in terms of data acquisition, knowledge discovery, continuous real-time monitoring and transition detection aspects. Chapter 3 elaborates on the Support Vector Machines method which we adapt for real time continuous monitoring. Chapter 4 explains RT-CAM in terms of knowledge discovery. Chapter 5 defines the experimental setup and Chapter 6 contains the performance analysis and evaluation of RT-CAM. Finally, Chapter 7 presents the conclusions and future work regarding RT-CAM.

2. RELATED WORK

In this chapter, activity monitoring studies are investigated considering how they approach the activity monitoring problem. These studies are grouped by the following criteria:

- *Data acquisition modules/sensors they employ:* Since one of the predominant sensor categorization issues is wearability, we group the data acquisition modules as **wearable sensors** and **environmental sensors**.
- *Knowledge discovery methods involved:* Studies are categorized by **feature extraction** and **classification** techniques. Also, a section is dedicated to **SVM** since our proposed method is an SVM based solution.
- *Continuous real time monitoring:* Some studies target only activity detection and does not perform continuous real time monitoring, hence we separately investigate studies on **continuous real time monitoring**.
- *Transition detection:* Which segmentation policy is used and whether a dedicated algorithm is needed for transition detection are important in terms of transition detection, hence studies are reviewed in terms of **segmentation** and the necessity of a **dedicated transition detection algorithm**.

2.1. DATA ACQUISITION

2.1.1. Wearable Sensors

Accelerometer has been used in a wide variety of activity monitoring studies. Clarke-Moloney et. al. [20] used accelerometer (ActivPALTM) to measure effect of mobility, age and ulcer size on ulcer healing. Considering a set of actions including walking, standing, sitting and lying, they concluded that mobility patterns do not vary remarkably by age between control group and the patients with leg ulcers.

Bourke et. al. [21] located tri-axial accelerometer sensors on trunk and thigh to differentiate between falls and activities of daily living (ADL). The tri-axial accelerometer is formed with two bi-axial Analog Devices ADXL210. They conclude that the most

suitable location for a fall sensor seems to be the trunk. In another research distinguishing between falls and ADL by thresholding the vertical velocity of the trunk by means of a wearable inertial sensor, Bourke et. al. [22] profile the vertical velocity of elderly people in a home environment. Godfrey et. al. [23] developed a system to evaluate the ability of detecting postural activity and postural transition using a single triaxial accelerometer located on the trunk, as shown in Figure 2.1, and an algorithm with lower complexity compared to the one executed in [53]. They compared their method Velocity Estimate and Scalar Product Activity (VESPA) to Discrete Wavelet Transform (DWT). Velocity estimation for VESPA, compares the absolute values of negative and positive peaks around the time of postural transitions to determine whether the postural transition is stand to sit (StSi) or sit to stand (SiSt) and detect walking. Results regarding VESPA are shown in Table 2.1.

Moore et. al. [24] monitored the gait in Parkinson's disease using combined accelerometer and gyroscope sensor array located on ankle. They calculated stride evaluating vertical linear acceleration and pitch angular velocity of the leg with an accuracy of 5 cm. Culhane et. al. [25] evaluated mobility of older adults using an accelerometer based system consisting two Analog Devices ADXL202 accelerometers. Their monitoring system detected static activities sitting, standing and lying along with dynamic activities with 92% accuracy and higher using a threshold setting approach. Leonard et. al. [26] researched on people with hyperactive, hypoactive and combinations of delirium monitoring them with



Figure 2.1. Sensor located on the chest for evaluating VESPA [23]

Table 2.1. Results of VESPA [23] in the form of (mean \pm standard deviation)

Action	Sensitivity (%)	Specificity (%)	Total postural transitions	Type of subject
lying	100 \pm 0	-	-	Young and healthy (10 subjects)
StSi	92 \pm 9	85 \pm 11	42 \pm 0	
SiSt	85 \pm 11	92 \pm 9		
walking	100 \pm 0	-	-	
lying	100 \pm 0	-	-	Elderly and healthy (10 subjects)
StSi	89 \pm 8	83 \pm 11	42 \pm 0	
SiSt	83 \pm 11	89 \pm 8		
walking	98 \pm 1	-	-	

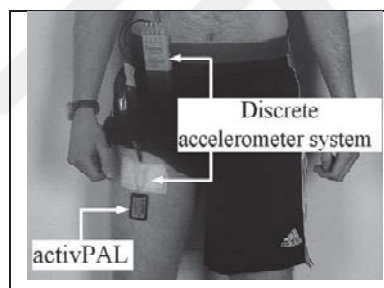


Figure 2.2. Motion analysis equipment used in the study of Leonard et. al. [26]

an accelerometer based system during 24 hours. They distinguished between static and dynamic activities and posture changing frequency. Their motion analysis equipment, shown in Figure 2.2, incorporates a discrete accelerometer located on the waist and, for comparison purpose, activPALTM mobility monitor which can detect sitting/lying, standing along with stepping [54] and could be found in the market. Kangas et. al. [27] compare low-complexity fall detection algorithms in a triaxial accelerometer setting where sensors are positioned in waist, wrist and head. Three algorithms with different complexity are studied on beginning of the fall, falling velocity, fall impact and posture after fall. Simple threshold-based algorithms are found to sufficiently detect falls with a sensitivity of 97-98% and specificity of 100% employing a triaxial accelerometer on the waist or head. Each accelerometer is formed with three uniaxial capacitive accelerometers (VTI Hamlin SCA

CDCV1G, amplitude range $\pm 12g$) [55] and are connected to a distinct data logger, with the sampling frequency being 400 Hz.

Karantonis et. al. [28] studied on real time classification of movements using a single triaxial accelerometer located on the waist with embedded intelligence fulfilling most of the signal processing operations on wearable unit. They tested their system in a laboratory environment with six subjects and obtained 90.8% overall accuracy. Table 2.2 displays the results of their study in more detail. Triaxial accelerometer utilized in measurement is a wireless module with 22x50x50 mm dimensions, except for the antenna, and 51 g weight.

Table 2.2. Results related to the study of Karantonis et. al. [28]

Discrimination	Accuracy (%)
Activity - rest	100
Postural orientation	94.1
Walking	83.3
Falls	95.6

Triaxial accelerometer unit is built using two orthogonally united dual axis accelerometers (MXR7210GL, MEMSIC, Inc., North Andover, MA) with the range $\pm 10g$, noise level 5.06 mg rms and bandwidth roughly 100 Hz. Lyons et. al. [29] formed a system monitoring older adults using an accelerometer based threshold method. The system they defined can differentiate between static and dynamic activities, also it is capable of detecting sitting, standing and lying postures. They monitored one older adult subject during four days for an average seven hours per day with two accelerometers, one of which is located on the trunk whereas the other is positioned at the thigh. Their testing unit employed two dual axis accelerometers (Analog Devices, ADXL202) and sampled the accelerometer signals at 50 Hz. They compared two thresholding techniques and their best achieved results on a daily basis are shown in Table 2.3.

Godfrey et. al. [30] reviews the studies which measure human movement by accelerometry. Properties of commercial accelerometer based activity monitors listed in their research are summarized in Table 2.4 and the products are shown in Figure 2.3.

Table 2.3. Best achieved results in the work of Lyons et. al. [29]

Activity	Accuracy (%) $\bar{x}(\sigma)$
sitting	93 (7.0)
standing	95 (4.1)
lying	84 (2.3)
moving	97 (1.8)

Table 2.4. Properties of commercial accelerometer based activity monitors listed in the work by Godfrey et. al. [30]

Proprietary activity monitors by accelerometry	Features
RT3 tri-axial research tracker kit [56]	Waist mounted
activPAL™ Professional	Uni-axial piezoresistive accelerometer, also cadence, number of steps and energy expenditure
ActiGraph GT1M	Single axis piezoelectric accelerometer, auto-turn on feature upon programming [56]
Cyma Step Watch3	Programmed through a PC, microprocessor controlled step counter (can record steps for up to two months)
Dynastream AMP331	Ankle mounted, two accelerometers (one uni-axial and one bi-axial), data analysis with SpeedMax technology (Dynastreams), data downloaded to a PC using an RF protocol (established commercially by Dynastream) and available for further processing in Excel.
Ossur PAM: Prosthetic Activity Monitor™	Lower leg mounted, one bi-axial and one uni-axial accelerometer
IDEEA: intelligent device for energy expenditure and physical activity®	Identify and differentiate more than 40 types of activities, including 15 different parameters of gait. Provides information on the onset, duration and frequency of each activity and computes the amount and intensity of these activities. Multiple sensors on a wide variety of locations on the upper and lower leg, wrist, sternum and foot via cables.



Figure 2.3. Accelerometry based proprietary activity monitors explained in the work by Godfrey et. al. [30]

2.1.2. Environmental Sensors

Kasteren et. al. [10] monitor the activities with a wireless sensor network as well as generative hidden Markov model and discriminative conditional random field as the monitoring model. Their proposed system employ reed switches, pressure mats, mercury contacts, passive infrared (PIR), float sensors and temperature sensors. Table 2.5 shows how they utilize these sensors for activity monitoring.

Yang et. al. [12] propose an activity recognition method using simple object information related to activities. Their method involves a penalized naive Bayes classifier. They compare their method with hidden Markov models and conditional random fields and conclude that their method achieves reduction in computation up to an order of magnitude in both learning and inference, without damaging accuracy.

Table 2.5. Sensors and their functionality in the work by Kasteren et. al. [10]

Sensor type	Sensor role in the activity monitoring setting
Reed switches	Measuring open-close states of doors and cupboards
Pressure mats	Measuring sitting on a couch or lying in bed
Mercury contacts	Movement of objects (e.g. drawers)
Passive infrared (PIR)	To detect motion in a specific area
Float sensors	Measuring the toilet being flushed
Temperature sensors	Measuring the use of the stove or shower

2.2. KNOWLEDGE DISCOVERY

Sensor data are mapped to activity labels upon knowledge discovery which incorporates *feature extraction* and *classification* operations. Feature extraction process generates a set of representative data, namely features, from the sensor data. Feature extraction can be as simple as calculating the mean value of the sensor data or other more complex methods such as signal processing algorithms. Another part of knowledge discovery is data classification. Features are input to classification schemes to obtain the activity mappings.

A data classification method processes the features to find which activity class they belong to.

2.2.1. Feature Extraction

Feature extraction methods utilized in the following studies reviewed can be briefly explained in the list below, before going on with presenting the studies in more detail:

- *Wavelet analysis* can provide information on both time and frequency domain characteristics of a signal unlike Fourier analysis [37].
- *Frequency-domain* features require transforming a window of sensor data to feature domain, generally using a fast Fourier transform [37].
- *Time-domain* features are generally statistical features obtained from a window of sensor data [37].
- *Heuristic* features are generated from a fundamental understanding which can be intuitive [37].
- *Median filter* affects the energy contained in the signal. A greater length of the median filter yields a smoother signal, resulting in greater amount of lost energy [39].
- *Independent component analysis* tries to solve the mixing matrix knowing the observed signals, that is a sample of the random variable x .
 - i. n being the number of observed signals (These signals are different linear mixtures of n statistically independent, non-gaussian source signals),
 - ii. s being an n -dimensional random vector, whose elements are the sources,
 - iii. The elements of the observed random vector x are different mixtures of the sources $x=As$, where A is an $n \times n$ mixing matrix which is to be solved [40].
- *Principal components* can be calculated using the eigendecomposition of the sample covariance matrix [40].

Maurer et. al. [33] designed an AMS with multiple sensors on different body positions identifying activities in real-time. They tested on *sitting*, *standing*, *walking*, *ascending stairs*, *descending stairs* and *running* with a separate sensing device (eWatch) worn on left wrist, belt, necklace, in the right trouser pocket, shirt pocket and bag. eWatch, which is

shown in Figure 2.4, incorporates a bi-axial accelerometer, light and temperature sensors along with microphone. In their experiments, they used both axes of the accelerometer and the light sensor, values being recorded at 50 Hz and accelerometer's both axes functioning at $\pm 2g$ range. Averaging the percentage of correctly classified feature vectors for all activities to express the recognition accuracy, they reported the most successful results to be around 88% with sensing device placed in the bag. This result is achieved in two different experimental configurations one of which is set up with features from Y axis of the accelerometer whereas the other one includes features from the x^2+y^2 value of the accelerometer. Decision tree classifier (C4.5 algorithm) with a 5-fold cross validation is used. They trained the classifier with the data acquired from six subjects to obtain a general classifier. The wrist is conveyed to be the best performing sensor location.



Figure 2.4. eWatch device used in the work by Maurer et. al. [33]

Preece et. al. [36] reviews feature extraction techniques targeting acceleration data to classify dynamic activities. Methods extracting time and frequency domain signal properties are compared considering two data sets acquired from 20 subjects. They conclude that although non-stationary activities could be classified using wavelet transform, dynamic activities of healthy subjects can better be classified with frequency based features. Success of their study is higher than 95% accuracy in intersubject classification with the best feature set they experimented.

Preece et. al. [37] investigates methods applied on data acquired from body-mounted sensors in order to classify normal activities and falls. They split features into several groups including *heuristic*, *time-domain*, *frequency-domain* or *time-frequency (wavelet)*. They state that distinguishing between postures and dynamic activities and identifying falls accurately is possible with these features and simple classification techniques based on thresholding, adding that activity sets involving larger number of activities require more advanced classification methods, sometimes combining different features.

Mathie et. al. [39] differentiate between activity and rest, with sensitivities larger than 0.98 and specificities ranging between 0.88 and 0.94. They applied their method on data collected from 26 subjects using a single tri-axial accelerometer located on the waist. They studied three characteristics of the signal: length of a smoothing median filter, width of the averaging window and the acceleration magnitude threshold.

Mantjarvi et. al. [40] achieved 83-90% accuracy in recognizing motions using independent component analysis and principal component analysis, observing ignorable differences between these two methods. One tri-axial accelerometer is placed on each of left and right side of the hip, classifying the collected data with multilayer perceptron classifier. Activities recognized are *start-stop points*, *level walk*, *down stairs walking* and *up stairs walking*.

Some studies pursuing activity monitoring with SVM necessitate trying different features and using combinations of features to increase classification success.

Foubert et. al. [57] detected *sitting* and *lying* postures, collecting the data with optical sensor arrays deployed on bed. In their work, eight types of features extracted from pressure signal and three classification techniques are experimented. *Sitting* and *lying* postures are detected with 100% accuracy using SVM, but combining different types of features. Most successful result they achieved is 94% when a single type of feature is used.

Qian et. al. [58] monitored activities using video samples, classifying the actions with Binary Tree Integrated SVM (BT-SVM). However, the algorithm presented necessitates determining which features are the best for BT-SVM. With this algorithm regarding best

classifying features as the best features; static actions such as *standing*, *sitting* and *squatting* along with *walking*, *jogging*, *sitting while standing*, *squatting while standing* and *falling* are classified with 95% accuracy on the average. The same algorithm classified *walking*, *jogging*, *running*, *boxing*, *clapping*, *waving hands* actions in Schüldt's database with 88.69% accuracy.

Kim et. al. [59] detect *running*, *walking*, *walking with a stick*, *crawling*, *forward boxing*, *boxing in standing position* and *sitting* related activity data collected from Doppler radar, using SVM. In the classification pursued by training SVM with six different features, success of the features ranges between 30.3% and 70.1%. Most successful result (92.8%) is achieved combining features. Decision tree structure is integrated to carry out multi-class categorization with SVM.

2.2.2. Classification

Studies reviewed in terms of classification methods incorporate ANN, Decision Tree and SVM. ANN employs a mathematical function representing the relationships between features to classify and classes to which these features belong. An optimization process is followed to map features to classes. Decision trees introduce a hierarchy of rules to group features into different hierarchies of classes based on the level of discriminativeness [43]. SVM finds a separating hyperplane between two classes and tries to map test data to one of these classes considering which region of the hyperplane they are located. The following are studies employing these methods for classification.

Engin et. al. [42] classified human tremor signals collecting acceleration data from Parkinsonian, essential and healthy subjects. Linear prediction coefficients, wavelet transform detail coefficients, wavelet transform based entropy and variance, power ratio and higher order cumulants are features extracted and then input to artificial neural network (ANN) classifier. They used scaled-conjugate (SCG) and Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithms and concluded that BFGS (91.02%) outperforms SCG (88.48%) in terms of accuracy.

Parkka et. al. [43] describes classification methods used in daily activities. They experimented on 16 people, logging roughly 31 h data in total inside an everyday environment. Experiments recorded in 2 h sessions with a set of wearable sensors placed on subjects. They tried three classification algorithms, which are custom decision tree (82%), automatically generated decision tree (86%) and artificial neural network (82%), total accuracy being designated in parenthesis. Experimented activities are *lying*, *rowing (with a rowing machine)*, *cycling with an exercise bike*, *sitting/standing*, *running*, *Nordic walking* and *walking*.

Zhang et. al. [44] presented an ANN to identify activities *level walking*, *running*, *ascending* and *descending stairs*, speeds of activities being determined by subjects as *slow*, *normal* and *fast*. Accurate identification success is 98.78%, 98.33%, 97.33%, and 97.29% for *walking*, *running*, *ascending* and *descending stairs* respectively. Also, speed of walking and running is estimated with average error 0.050 ± 0.747 km/h (mean \pm standard deviation). Their data collection device, which is in the form of insole, is the multiple pressure sensors containing SIMS illustrated in Figure 2.5.



Figure 2.5. The SIMS used in work by Zhang et. al. [44]

2.2.3. SVM

Begg et. al. [45] developed a method to recognize aging based changes in gait using SVM. In the experiments applied on 12 young and 12 elderly subjects, 91.7% overall accuracy is achieved in differentiating between two gait patterns. Selecting features from more than one gait type increases gate recognition success. Differentiating between three age groups with 100% accuracy is achieved thanks to using features from three data types, selecting one feature set form each type. Foot-ground reaction forces are logged using AMTI, USA. For logging lower limb movement, a 3D PEAK (Peak Performance Inc., USA) Motion

analysis component is used with reflective markers worn on hip, knee, ankle, heel and toe. Extracted features are basic, kinetic and kinematic gait attributes, totalling 24.

Sazonov et. al. [5] classified postures and activities of subjects suffering from stroke using a multi-class SVM [60] based method on data acquired through a shoe-based sensing tool. They achieved the classification accuracy results shown in Table 2.6 for *sit*, *stand*, *walk*, *ascend* and *descend* activities using SVM with linear and Gaussian kernels.

Table 2.6. SVM classification accuracy (%) with different kernels in the work by Sazonov et. al. [5]

Action name	Linear kernel (%)	Gaussian kernel (%)
sit	100	100
stand	100	100
walk	99	100
ascend	81	96
descend	97	99

Figure 2.6 illustrates a detailed view of the sensing tool used in the work by Sazonov et. al. [5]. Five force-sensitive sensors are placed within the insole per shoe such that they are in contact with heel, metatarsal bones and the toe, which significantly effect the contact of the foot with ground. Sensor data are sampled at 25 Hz and sent to the computer wirelessly. Their wireless data acquisition tool is on top of Wireless Intelligent Sensor and Actuator Network (WISAN) [61]. Collected data are composed of 116 segments lasting 15 to 25 seconds, totaling roughly half an hour.

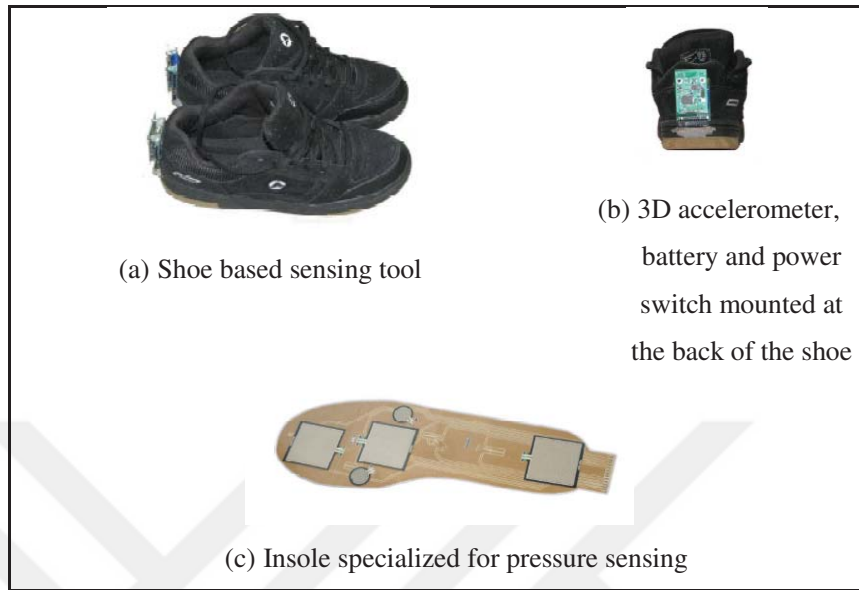


Figure 2.6. Sensing module used in the work by Sazonov et. al. [5]

2.3. CONTINUOUS REAL TIME MONITORING

Aiello et. al. [62] develops a human activity monitoring system, which is a MAPS (Mobile Agent Platform for Sun SPOTs) based application. Their system is capable of recognizing *lyingDown*, *sitting*, *standingStill* and *walking* in real-time with in-node signal processing. Their system is built on top of a wireless body sensor network with a pair of 3D accelerometers, one of them positioned on the waist and the other one located on the thigh. The waist sensor computes mean of X, Y and Z axis measurements as well as minimum and maximum of values along X axis while the thigh sensor computes minimum along X axis. Practising transitions *standingStill_walking*, *walking_standingStill*, *standingStill_sitting*, *sitting_lyingDown*, *lyingDown_sitting* and *sitting_standingStill*, they identified all transitions with 100% accuracy when transition period lasts five seconds. Among transitions completed in less than five seconds, only transition *standingStill_walking* can be recognized with 100% accuracy, which puts a restriction on transition duration as a prerequisite of correct identification. Their study incorporates a K-Nearest neighbour based classifier where $K=1$ and Manhattan distance is used. Table 2.7 specifies transition detection accuracy values with corresponding transition periods in the study of Aiello et. al. [62]. Comparison of our work RT-CAM with Aiello et. al. [62] is presented in Section 6.4.

Table 2.7. Accuracy of transition detection for various transition periods

Transition type	Matching percentage	Duration of transition period (s)
walking_standingStill	0	<1
standingStill_sitting	0	<1
sitting_lyingDown	0	<1
lyingDown_sitting	~67	<1
sitting_standingStill	90	<4
standingStill_walking	100	Any duration between 0 to 9 seconds

Okeyo et. al. [63] presents a dynamic segmentation model for real time continuous activity recognition. They utilize varying time windows, benefitting from temporal properties of the acquired data from sensors. They develop an ontology based prototype and achieve 84.4%, 91.9% and 88.7% accuracy, applying three different methods.. Table 2.8 shows the effect of shrinking and expanding the window on recognition accuracy. They calculate the accuracy according to Equation (2.1) where tp , fp , tn , fn indicate *true positive*, *false positive*, *true negative* and *false negative* respectively. They feed activity segments to ontology module.

Table 2.8. Effect of shrinking and expansion on recognition accuracy in the work by Okeyo et. al. [63]

Action name	Accuracy (%)		
	Without shrinking or expansion	With shrinking	With shrinking and expansion
MakeTea	95.1	95.1	95.1
MakeCoffee	100	100	100
MakeChocolate	100	100	100
MakePasta	96.6	89.7	72.4
BrushTeeth	73.7	89.5	89.5
HaveBath	67.9	85.7	85.7
WashHands	100	100	100
WatchTelevision	41.7	75	66.7
Average	84.4	91.9	88.7

They experimented on data synthetically generated by a computer application. Activity set they study includes the actions *makeTea*, *makeCoffee*, *makeChocolate*, *makePasta*, *brushTeeth*, *haveBath*, *washHands*, and *watchTelevision*.

$$\text{Accuracy} = (tp+tn)/(tp+fp+tn+fn) \quad (2.1)$$

2.4. TRANSITION DETECTION

Muscillo et. al. [64] detect individual actions *walking*, *stair climbing*, *stair descending* and the transition *stair descending to walking* with 96%, 98%, 78% and 40% accuracy respectively. They collect the activity data using a single 2 axis accelerometer and carry out classification with Markov Chain Model integrated to naïve Bayes classifier. They classify the activities in an off-line manner.

Ganea et. al. [65] detect *sit-to-stand* and *stand-to-sit* transitions with accuracy higher than 95% on healthy subjects and 89% on chronic pain subjects.

2.5. SEGMENTATION

Monitoring activity sequences necessitates determining the start and end of simple actions, in other words *segmentation*, in addition to identifying the constituent simple actions. There exist three commonly preferred methods [66] in order to segment continuous data: explicit segmentation, time based windowing and sensor event based windowing. In explicit segmentation [67], sensor data are partitioned such that each segment corresponds to an action. It necessitates finding the optimal segment size in the training phase where activity model is extracted. Segments used in prediction phase can be different from segments generating models in the training phase which can lead to deterioration in classification performance. In time based windowing [10] [68, 69], each segment lasts for equal amount of time. Time based windowing makes learning activity models easier in training phase compared to explicit segmentation. However, it necessitates selecting optimal time interval. Finally, with sensor event based windowing, each segment contains equal number of sensor events. Disadvantage of this method is the possibility of placing events, which are not really related to each other (for instance two sensor events between which a significant amount of time passed), into the same segment. Despite exhibiting this problematic behaviour, sensor event based windowing could be a beneficial solution if it is applied considering the relation between sensor events.

2.6. APPROACHES APPLYING A DEDICATED ALGORITHM FOR DETECTING TRANSITIONS

Boyd et. al. [70] detect individual actions *sitting*, *standing*, *walking*, *reaching for an object on a table* and *eating* along with transitions *sitting-standing-walking-sitting*, *sitting-reaching-walking*, *sitting-eating-walking* and *sitting-walking-standing* with 94% accuracy. They classify the data acquired through a single tri-axis accelerometer using log-likelihood ratio test and Hidden Markov Model. They first detect transitions and then actions between them which necessitates running a dedicated algorithm for transition detection. On the contrary, our approach first detects individual actions and infers the existence and identification of a transition based on the type of detected simple actions, eliminating the necessity of running a dedicated transition detection algorithm. They catch transitions within a one second delay.

Jarchi et. al. [71] recognize *walking vs reading* (90.5%), *walking vs lying down and getting up* (95.75%), *walking vs lying* (82.084%, average over 5 subjects), *walking vs falling* (90%, average over 4 subjects), values in paranthesis designating accuracy results. Transition accuracy they achieve is 90% which is the average over five subjects. They collect the data using a single ear-worn sensor and apply Singular Spectrum Analysis for classification. They generate similarity patterns for each transition.

Foubert et. al. [57] recognize *lie-to-sit* transition on young healthy and older healthy (98 %) and on hip fracture group and stroke group (90 %), with accuracy results designated in paranthesis. They classify the data using Support Vector Machines with the help of a combination of features. Their data collection tool is an array of optical pressure sensors deployed on bed. They run an algorithm dedicated for transition detection.

2.7. SUMMARY

We reviewed activity monitoring systems in terms of *type of sensors* (wearable or environmental sensors) and *knowledge discovery modules* (feature extraction and classification) they utilize. Also, we included an investigation of studies performing *continuous real time monitoring of activities*, *segmentation* policies and approaches applying *a dedicated algorithm for detecting transitions*. Considering the studies reviewed, our proposed approach RT-CAM can be placed in the categories in the literature, being described as follows: Our proposed real time continuous activity monitoring method is an adaptation of SVM where feature extraction phase is also responsible for most of the classification phase. To come up with a wearable solution which is non-invasive, we employ a single accelerometer. We do not run a dedicated algorithm to detect transitions, instead we infer the existence and type of transition considering the detected types of actions right before and after the transition. For segmenting the real time continuous data stream, we pick explicit segmentation as the segmentation policy.

3. SUPPORT VECTOR MACHINES

In this chapter, Support Vector Machines (SVM) method is explained since RT-CAM is an SVM based solution. We utilize SVM considering its advantages over the alternatives in application areas such as image based digit recognition [72,73], text classification [74] along with person detection and recognition [75].

General SVM architecture comprises generation of a separating hyperplane comparing features of two classes for training phase, followed by the prediction stage where test data features are checked to see on which side of the hyperplane they lie. This procedure is summarized in Figure 3.1. The concept of *support vector* indicates the features which determine the position of the separating hyperplane. Parameters of separating hyperplane are found calculating the solution of systems of equations in the form shown as Equation (3.1). Aim of SVM is to find X , which is the unknown of this equation. A is the coefficient matrix and B is the vector containing class labels, which are 1 or -1, since SVM is a binary classifier, meaning that SVM differentiates between only two classes. For multi-class problems, *one-against-the-rest* approach can be used. If M classes exist, M SVM's are formed, determining a hyperplane between class k and other classes. In prediction for a new instance, the decision generated by the SVM which puts the prediction furthest into the positive region of the instance space is picked [76]. Though solving multi-class problems with SVM is generally carried out using some voting schemes involving a set of binary classification decision functions [77,78], methods for multi-class classification which do not use the combination of binary rules also exist [79].

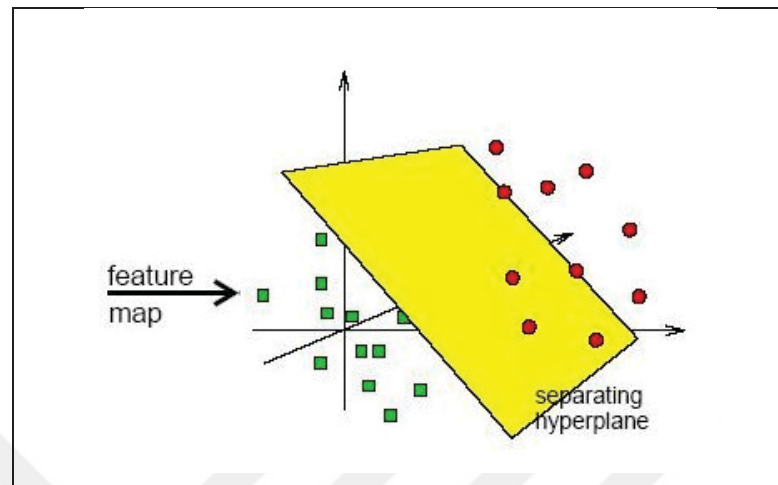
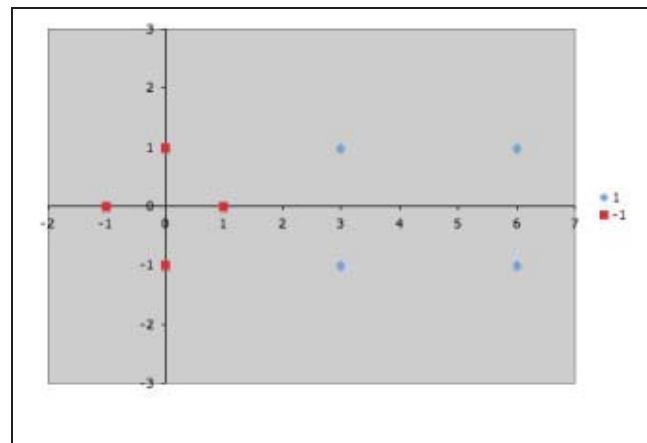


Figure 3.1. SVM architecture

$$AX=B \quad (3.1)$$

For a more detailed explanation, a numerical example [80] is presented as follows: Given the positively labelled data set P and negatively labelled data set N in \mathcal{R}^2 ; $P = \left\{ \begin{pmatrix} 3 \\ 1 \end{pmatrix}, \begin{pmatrix} 3 \\ -1 \end{pmatrix}, \begin{pmatrix} 6 \\ 1 \end{pmatrix}, \begin{pmatrix} 6 \\ -1 \end{pmatrix} \right\}$ and $N = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix} \right\}$, which are illustrated in Figure 3.2. Because data points are linearly separable as can be seen in Figure 3.2, a linear SVM, meaning that kernel function $\Phi()$ is the identity function, could be used.

Figure 3.2. Elements in sets P (in blue) and N (in red)

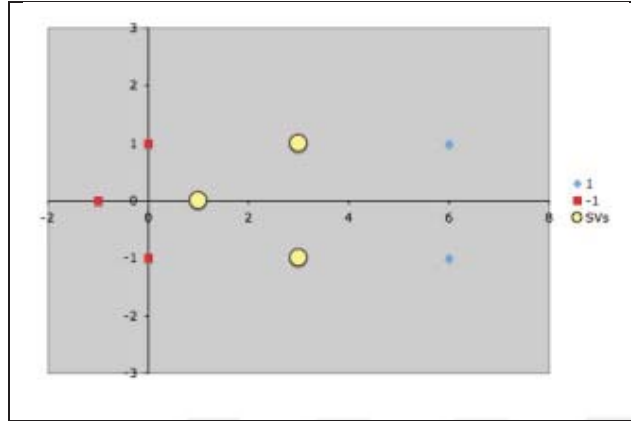


Figure 3.3. Three support vectors marked in yellow

It could be visually determined that the following vectors are support vectors, considering Figure 3.3: $S = \{s_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, s_2 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}, s_3 = \begin{pmatrix} 3 \\ -1 \end{pmatrix}\}$. Vectors will be used with a bias input 1 and they will be denoted by a superscripted d, which means if $s_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, then $s_1^d = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$. The goal is finding values for the α_i corresponding to Equation (3.2), (3.3) and (3.4).

$$\alpha_1 \Phi(s_1) \cdot \Phi(s_1) + \alpha_2 \Phi(s_2) \cdot \Phi(s_1) + \alpha_3 \Phi(s_3) \cdot \Phi(s_1) = -1 \quad (3.2)$$

$$\alpha_1 \Phi(s_1) \cdot \Phi(s_2) + \alpha_2 \Phi(s_2) \cdot \Phi(s_2) + \alpha_3 \Phi(s_3) \cdot \Phi(s_2) = +1 \quad (3.3)$$

$$\alpha_1 \Phi(s_1) \cdot \Phi(s_3) + \alpha_2 \Phi(s_2) \cdot \Phi(s_3) + \alpha_3 \Phi(s_3) \cdot \Phi(s_3) = +1 \quad (3.4)$$

Since $\Phi()=I$, meaning that kernel function is the identity function, Equation (3.2), (3.3) and (3.4) reduce to Equation (3.5), (3.6) and (3.7).

$$\alpha_1 s_1^d \cdot s_1^d + \alpha_2 s_2^d \cdot s_1^d + \alpha_3 s_3^d \cdot s_1^d = -1 \quad (3.5)$$

$$\alpha_1 s_1^d \cdot s_2^d + \alpha_2 s_2^d \cdot s_2^d + \alpha_3 s_3^d \cdot s_2^d = +1 \quad (3.6)$$

$$\alpha_1 s_1^d \cdot s_3^d + \alpha_2 s_2^d \cdot s_3^d + \alpha_3 s_3^d \cdot s_3^d = +1 \quad (3.7)$$

After dot products are calculated, Equation (3.8), (3.9) and (3.10) are obtained. Carrying out algebraic operations, the solution to the system identified by Equation (3.8), (3.9) and (3.10) is found to be $\alpha_1 = -3.5$; $\alpha_2 = 0.75$ and $\alpha_3 = 0.75$.

$$2\alpha_1 + 4\alpha_2 + 4\alpha_3 = -1 \quad (3.8)$$

$$4\alpha_1 + 11\alpha_2 + 9\alpha_3 = +1 \quad (3.9)$$

$$4\alpha_1 + 9\alpha_2 + 11\alpha_3 = +1 \quad (3.10)$$

In order to find the discriminating hyperplane using the α values, Equation (3.11) is used, yielding the result shown with Equation (3.12).

$$w^d = \sum_i \alpha_i s_i^d \quad (3.11)$$

$$w^d = \sum_i \alpha_i s_i^d = -3.5 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + 0.75 \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix} + 0.75 \begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix} \quad (3.12)$$

Last entry in w^d corresponds to hyperplane offset b since vectors are augmented with a bias previously, resulting in separating hyperplane equation $y=wx+b$ using $w=\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $b=-2$. The discriminating hyperplane, which successfully classifies the data points, is plotted in Figure 3.4.

The following is another numerical example [80], which elaborates the nonlinear case: Given the positively labelled data set P and negatively labelled data set N in \mathcal{R}^2 : $P = \left\{ \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 2 \\ -2 \end{pmatrix}, \begin{pmatrix} -2 \\ -2 \end{pmatrix} \right\}$ and $N = \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix} \right\}$ which are plotted in Figure 3.5. As can be seen, a discriminating hyperplane to separate these data points does not exist in the original input space. Using the kernel function defined in Equation (3.13), data points in sets P and N are transformed to $P'=\Phi(P)$ and $N'=\Phi(N)$ respectively as follows:

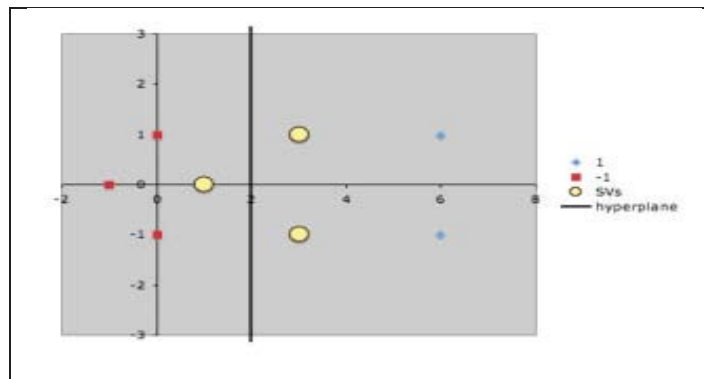


Figure 3.4. The discriminating hyperplane

$P' = \left\{ \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 6 \\ 2 \end{pmatrix}, \begin{pmatrix} 6 \\ 6 \end{pmatrix}, \begin{pmatrix} 2 \\ 6 \end{pmatrix} \right\}$ and $N' = \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix} \right\}$ which are illustrated in Figure 3.6.

$$\Phi_1 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{cases} \begin{pmatrix} 4-x_2+|x_1-x_2| \\ 4-x_1+|x_1-x_2| \end{pmatrix} & \text{if } \sqrt{x_1^2 + x_2^2} > 2 \\ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} & \text{otherwise} \end{cases} \quad (3.13)$$

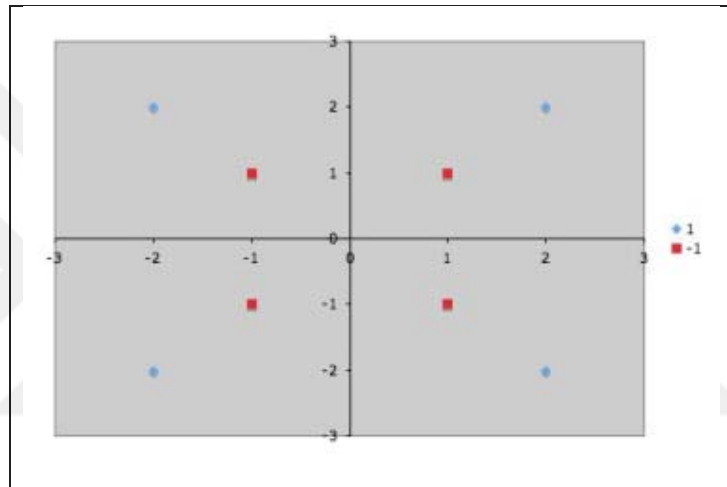


Figure 3.5. Set of data points P (in blue) and N (in red)

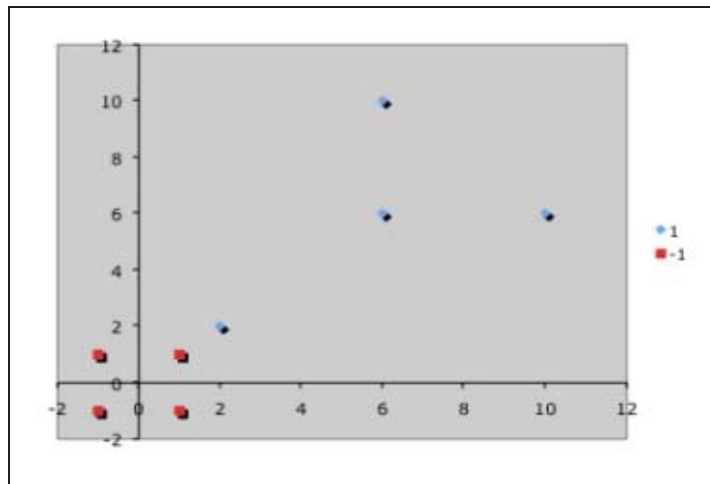


Figure 3.6. Feature space representation of data points

Two support vectors exist, which are $S = \{s_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, s_2 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}\}$ as illustrated in Figure 3.7. After augmenting vectors with 1 as the bias input, α_i values are calculated according to Equation (3.14) and (3.15).

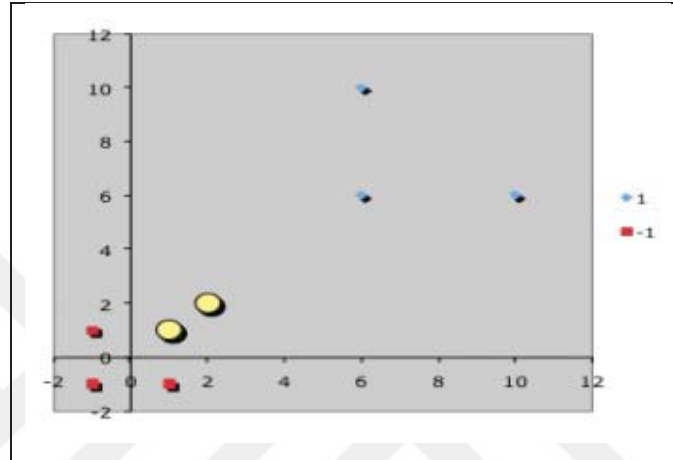


Figure 3.7. Support vectors in feature space (in yellow)

$$\alpha_1 \Phi_1(s_1) \cdot \Phi_1(s_1) + \alpha_2 \Phi_1(s_2) \cdot \Phi_1(s_1) = -1 \quad (3.14)$$

$$\alpha_1 \Phi_1(s_1) \cdot \Phi_1(s_2) + \alpha_2 \Phi_1(s_2) \cdot \Phi_1(s_2) = +1 \quad (3.15)$$

Using the kernel function shown in Equation (3.13); the problem is reduced to Equation (3.16) and (3.17).

$$\alpha_1 s_1^d \cdot s_1^d + \alpha_2 s_2^d \cdot s_1^d = -1 \quad (3.16)$$

$$\alpha_1 s_1^d \cdot s_2^d + \alpha_2 s_2^d \cdot s_2^d = +1 \quad (3.17)$$

After dot products are calculated, the system of equations shown in Equation (3.18) and (3.19) are obtained and the solution is found to be $\alpha_1 = -7$ and $\alpha_2 = 4$.

$$3\alpha_1 + 5\alpha_2 = -1 \quad (3.18)$$

$$5\alpha_1 + 9\alpha_2 = +1 \quad (3.19)$$

Then, the discriminating hyperplane is evaluated using Equation (3.20), yielding the result designated in Equation (3.21). Hence, separating hyperplane equation $y=wx+b$ is formed using $w=\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $b=-3$, which is plotted in Figure 3.8.

$$w^d = \sum_i \alpha_i s_i^d \quad (3.20)$$

$$w^d = \sum_i \alpha_i s_i^d = -7 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + 4 \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ -3 \end{pmatrix} \quad (3.21)$$

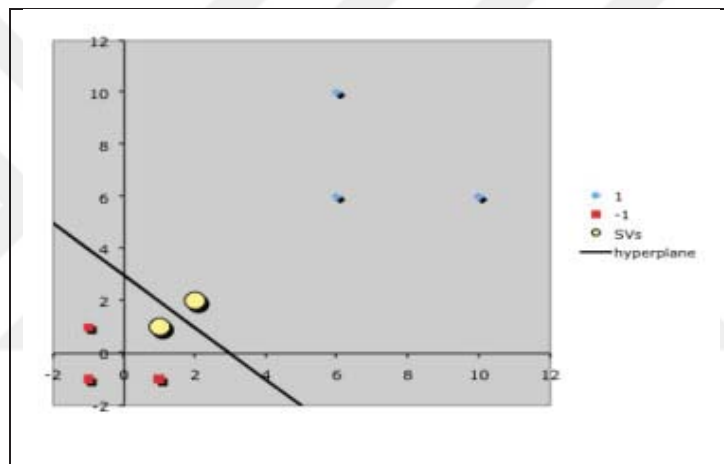


Figure 3.8. Discriminating hyperplane calculated for nonlinear example

So far, how the separating hyperplane generation, which is the training phase is explained. In order to predict the class of the test data, Equation (3.22) is used where $\sigma(z)$ yields the sign of z , identifying the test data to be in *positive* or *negative* class. For instance, the kernel function being the one shown in Equation (3.13), $x=(4,5)$ is labelled to be in negative class as illustrated in Equation (3.23).

$$f(x) = \sigma\left(\sum_i \alpha_i \Phi(S_i) \cdot \Phi(x)\right) \quad (3.22)$$

$$\begin{aligned} f\begin{pmatrix} 4 \\ 5 \end{pmatrix} &= \sigma\left(-7\Phi\begin{pmatrix} 1 \\ 1 \end{pmatrix} \cdot \Phi\begin{pmatrix} 4 \\ 5 \end{pmatrix} + 4\Phi\begin{pmatrix} 2 \\ 2 \end{pmatrix} \cdot \Phi\begin{pmatrix} 4 \\ 5 \end{pmatrix}\right) \\ &= \sigma\left(-7 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} + 4 \begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}\right) \\ &= \sigma(-2) \end{aligned} \quad (3.23)$$

What generalization accuracy will be achieved is not certain. If the generalization ability of the SVM model produced is not satisfactory, this is probably caused by the selection of kernel function.

3.1. SUMMARY

Being a binary classifier, SVM comprises generation of a separating hyperplane comparing features of two classes for training phase, followed by the prediction stage where test data features are checked to see on which side of the hyperplane they lie. For multi-class problems, several approaches exist which use and do not use combination of binary rules.

4. REAL TIME CONTINUOUS MONITORING

In this chapter, operations of RT-CAM components are explained. RT-CAM, employs an adaptation of SVM to recognize simple actions within the composite action they constitute and partitions the continuous activity data using explicit segmentation. For avoiding the use of a dedicated algorithm to detect transitions, RT-CAM relies on accurate classification of simple actions to infer the existence and identification of transitions based on the types of recognized simple activities.

4.1. KNOWLEDGE DISCOVERY

4.1.1. Segmentation

We follow explicit segmentation scheme explained in Section 2.5. Hence, each segment corresponds to a simple action. As elaborated in Section 2.5, optimal segment size should be determined in explicit segmentation since segment sizes differing in training and prediction phases can lead to deterioration in classification performance. We apply fixed sized partitioning on the continuous data where the subjects are made to perform activities such that each simple action and transition performed fit into the fixed segment size.

4.1.2. Adaptation of SVM

The proposed method, which is formed as an SVM adaptation, differs from the SVM in the following aspects:

- SVM is a binary classifier, which means only two classes can be differentiated using SVM. RT-CAM_{KD} approaches differently to the multi-class problem which is multi-activity differentiation in our context. RT-CAM_{KD} introduces the notion of *reference action* and assigns the reference action to be a pseudo class to form a basis of comparison between the actual classes. An error value, which is the measure of the difference between the training and test data, is calculated. Then, the action yielding the smallest error value is assigned to be the detected action.

- SVM tries to find a separating hyperplane between two classes to be differentiated and the type of the test data is determined considering which side of the hyperplane the test data lie whereas RT-CAM_{KD} avoids separating hyperplane calculation. By doing that, the operation of solving systems of equations carried out in SVM is replaced by the operation of multiplying a matrix by a vector in RT-CAM_{KD} where inverse matrix calculation, which is necessary for solving systems of equations, is eliminated.
- SVM maps the data to a new space using proper Kernel functions if the data are not linearly separable so that a separating hyperplane can be found in the new space. Though RT-CAM_{KD} eliminates the separating hyperplane calculation, a Kernel function is still used because we used the same Kernel function in our preliminary study of this work which is accepted as a publication [52] and we obtained successful results and we went on using the Kernel function.

Knowledge discovery module of RT-CAM, which we abbreviate as RT-CAM_{KD} is formed of the procedures explained in Equation (4.1)-(4.5) and Algorithm 4.1 to Algorithm 4.3.

Firstly, Z , av , *Kernel* and *error* functions are presented since they are used in the algorithms that follow. The legend shown below explains the notations used throughout the algorithms and equations.

- $C^{m \times n}$: a matrix with size $m \times n$
- C^{-1} : Inverse of $C^{m \times n}$
- $c(i,:)$: i th row of $C^{m \times n}$
- $c(:,i)$: i th column of $C^{m \times n}$
- $\mathbf{b}^n = (b_1^n, b_2^n, \dots, b_n^n)$ where \mathbf{b}^n represents a vector of size n
- $\mathbf{A} = \{a_1, a_2, \dots, a_s\}$ where \mathbf{A} denotes the set of simple actions
- $\mathbf{P}_a^{m \times n}$: pattern matrix for simple action a
- $\mathbf{R}^{m \times n}$: reference action where $r_{ij} = 1 \forall 1 \leq i \leq m \text{ and } \forall 1 \leq j \leq n$
- If $C^{m \times 3}$ is an acceleration matrix, then $c(:,1)$, $c(:,2)$ and $c(:,3)$ are x , y and z axis acceleration values respectively.

Padding function Z , explained in Equation (4.1), takes two vectors as the input and pads the vector, having smaller number of elements, with zero until number of elements in two vectors become equal. In Equation (4.1), V denotes an ordered pair such that $V=(\mathbf{a}^x, \mathbf{b}^y)$. When a vector operation necessitates the vectors having equal number of elements but it is possible that the vectors may not satisfy this requirement, Z function is used.

$$\mathbf{V}^y = Z(\mathbf{V}) = \begin{cases} \mathbf{V}^y = \{ (\mathbf{c}^y, \mathbf{b}^y) \mid c_i^y = a_i^x \ \forall 1 \leq i \leq x \text{ and } c_i^y = 0 \\ \quad \forall x + 1 \leq i \leq y, y > x \\ \\ \mathbf{V}^y = \{ (\mathbf{a}^x, \mathbf{c}^x) \mid c_i^x = b_i^y \ \forall 1 \leq i \leq y \text{ and } c_i^x = 0 \\ \quad \forall y + 1 \leq i \leq x, y < x \\ \\ \mathbf{V}^y = \{ (\mathbf{a}^x, \mathbf{b}^y) \}, \text{ otherwise} \end{cases} \quad (4.1)$$

Equation (4.2) shows av function which adds corresponding entries in two input vectors after padding them if they have different number of entries. The ordered pair $(\mathbf{x}^r, \mathbf{y}^r)$ is calculated using Z function.

$$\mathbf{c}^r = av(\mathbf{a}^m, \mathbf{b}^n) = \begin{cases} c_i^r = x_i^r + y_i^r, & \forall 1 \leq i \leq r \text{ and } m \neq n \\ c_i^r = a_i^m + b_i^n, & \forall 1 \leq i \leq m \text{ and } r = m = n \end{cases} \quad (4.2)$$

$$(\mathbf{x}^r, \mathbf{y}^r) = Z(\mathbf{a}^m, \mathbf{b}^n)$$

Equation (4.3) describes Kernel function K which appends a new entry to the input vector. The appended entry is calculated using the other entries in the input vector.

$$\mathbf{t}^{n+1} = K(\mathbf{v}^n) = \begin{cases} t_i^{n+1} = v_i^n, & \forall 1 \leq i \leq n \\ t_i^{n+1} = \sum_{i=1}^n \frac{1}{1 + e^{v_i^n}}, & i = n + 1 \end{cases} \quad (4.3)$$

error function, explained in Equation (4.4), finds an average error value to find a measure of difference between the two input vectors so that the difference between training and the test data can be determined.

$$\mathit{error}(\mathbf{t}^m, \mathbf{e}^n) = \begin{cases} \frac{1}{n} \sum_{i=1}^n \left| \frac{e_i^n - t_i^n}{t_i^n} \right|, & m = n \\ (\mathbf{t}^r, \mathbf{e}^r) = \mathbf{Z}(\mathbf{t}^m, \mathbf{e}^n) \\ \frac{1}{r} \sum_{i=1}^r \left| \frac{e_i^r - t_i^r}{t_i^r} \right|, & \text{otherwise} \end{cases} \quad (4.4)$$

Algorithm 4.1 explains *coreTraining* module. The input matrix $\mathbf{C}^{m \times n}$ represents the acceleration matrix processed as will be described in *featureExtraction* procedure. Row vectors $\mathbf{c}(\mathbf{i}, :)$ $\forall 1 \leq i \leq m/2$ belong to one of actual actions whereas $\mathbf{c}(\mathbf{i}, :)$ $\forall \frac{m}{2} \leq i \leq m$ belong to the pseudo class which is the reference action. Each row vector in the input $\mathbf{C}^{m \times n}$ is exposed to function \mathbf{K} . Vectors output by \mathbf{K} are compared in terms of number of elements they contain to pad the one having less elements with zeros. Dot product (\cdot) of the padded vectors are calculated and are stored in matrix $\mathbf{F}^{m \times m}$. Vector \mathbf{b}^m is populated with 1 and -1 values indicating the positive and negative class labels. As mentioned previously, RT-CAM_{KD} does not calculate a separating hyperplane, hence RT-CAM_{KD} does not calculate inverse of $\mathbf{F}^{m \times m}$ to find the unknowns of the system of linear equations. Finally pattern vector \mathbf{w}^{n+1} is calculated, using the \mathbf{x}^m vector. \mathbf{w}^{n+1} is the discriminative value which we calculate instead of the separating hyperplane and is used in the prediction phase to compare against the test data.

Algorithm 4.1. Core training algorithm

```

wi=0 ∀1 ≤ i ≤ t
coreTraining(Cmxn)
for i from 1 to m
    for j from 1 to m
        (p0,p1)=Z(K(c(i,:)),K(c(j,:)))
        fij=p0 · p1
    end for
end for
xm=mult(Fmxm, bm) where bm= { bi = 1, ∀1 ≤ i ≤ m/2
    { bi = -1, ∀m/2 + 1 ≤ i ≤ m
for i from 1 to m
    wn+1=av(wn,mult(K(c(i,:)),xim))
end for
return wn+1

```

featureExtraction procedure merges reference action matrix \mathbf{R}^{mxn} with acceleration matrix \mathbf{C}^{mxn} whose features are extracted after Kernel function \mathbf{K} is applied. Row vectors, which originate from \mathbf{C}^{mxn} , are the data representing action class whereas the ones originating from \mathbf{R}^{mxn} represent the pseudo class. Then, the resulting matrix $\mathbf{D}^{(2m) \times (n+1)}$ shown in Equation (4.5) is processed in *coreTraining* procedure to return \mathbf{w}^{n+1} patterns.

$$\mathbf{d}(\mathbf{i}, :) = \begin{cases} \mathbf{K}(\mathbf{c}(\mathbf{i}, :)), & \forall 1 \leq \mathbf{i} \leq \mathbf{m} \\ \mathbf{K}(\mathbf{r}(\mathbf{i} - \mathbf{m}, :)), & \forall \mathbf{m} + 1 \leq \mathbf{i} \leq 2\mathbf{m} \end{cases} \quad (4.5)$$

Training algorithm is explained in Algorithm 4.2. Pattern $\mathbf{P}_a^{\text{mxn}}$ for each simple action is composed of the features of each training sample $\mathbf{M}_i^{\text{mxn}}$ belonging to that action where $\mathbf{M}_i^{\text{mxn}}$ represents an acceleration matrix. Hence, each action is characterized by a matrix whose rows are feature vectors extracted from training samples.

In the prediction phase elaborated in Algorithm 4.3, features of acceleration matrix \mathbf{C}^{mxn} to be labelled are extracted. Generated feature vector \mathbf{f}^{n+1} is compared against each previously generated pattern $\mathbf{P}_a^{\text{mxn}}$ to calculate an error value for each simple action, which is $\mathbf{g}(a_i)$,

using the *error* function. The simple action yielding the smallest error value is assigned as the detected action.

Algorithm 4.2. Training algorithm

```

Ta={M1mxn, M2mxn, ..., Msmxn } where Mimxn is a training sample for simple
action a
training()
for each a ∈ A
    pa(i,:)= featureExtraction(Mimxn) ∨ Mimxn ∈ Ta
end for

```

A numerical example is given in Appendix A to demonstrate how RT-CAM_{KD} works.

Algorithm 4.3. Prediction algorithm

```

L:set of error values such that g:A→L
Arg min g(ai) := {ai|∃y: g(ai) ≤ g(y)}
g
prediction(Cmxn)
fn+1=featureExtraction(Cmxn);
g(ai)←  $\frac{1}{u} \sum_{i=1}^u \mathbf{error}(\mathbf{p}_{a_i}(\mathbf{i},:), \mathbf{f}^{n+1})$ , ∃ ai∈A given Paiuxv
return Arg min g(ai)
g

```

4.2. SUMMARY

We explained how knowledge discovery module of RT-CAM, which is RT-CAM_{KD}, works. Being an adaptation of SVM, RT-CAM_{KD} performs feature extraction phase and most of the classification phase at once. Unlike SVM, RT-CAM_{KD} can differentiate between multiple classes, comparing the similarity between actual classes and the pseudo action introduced, which is the reference action. As the segmentation policy to partition continuous real time data stream, RT-CAM_{KD} follows explicit segmentation where each segment corresponds to either a simple action or a transition.

5. EXPERIMENTAL SETUP

In this chapter, experimental context for testing RT-CAM is explained. Firstly, ambient assisted living scenario is given to elaborate on the activity set, mentioning the way the actions in the activity set are practiced. Also, the reason why this activity set is chosen is justified. Secondly, analysis of the sensor data is presented, introducing the data collection tool and generation of the data set. Then, the real testbed is described.

5.1. AMBIENT ASSISTED LIVING SCENARIO

The set of simple actions is composed of *eating*, *pouring*, *turningKey*, *drinking* and *toothBrushing*. These actions are selected since they can be the constituents of more complex actions performed in a natural setting of a smart home environment as well as professional contexts such as process control. Being hand oriented activities, these actions are also suitable for our data collection tool since we acquire data using a single 3D accelerometer placed on the right wrist. The following list explains how these actions can form more complex activities in different contexts.

- *Pouring* could be both pouring cornflakes from its package to a bowl for breakfast and pouring food ingredients to a saucepan for cooking in a home setting while it could also be performed for pouring chemicals to tanks in water purification plants.
- *TurningKey* can be for opening the door of a room at home and for starting an engine in the plant.
- People can perform a *drinking-like* movement pattern with a glass for clearing the mouth after *toothBrushing* apart from regular *drinking* while *eating*.

Contextual properties of the simple actions mentioned above enables forming logical combinations of them.

5.2. SENSOR DATA ANALYSIS

In this chapter, sensor data structure and the sensor itself are introduced. The nature of the data set is discussed, elaborating on the conditions under which the activity data are acquired.

5.2.1. Data Collection and Processing Tools

The raw sensor data are composed of X, Y and Z axis acceleration values acquired from the 3D accelerometer built in TI Chronos eZ-430 [81] shown in Figure 5.1-a, which is a CC430 based wearable device. The RF access point illustrated in Figure 5.1-b is connected to the PC with USB interface and enables wireless communication between the sensor and PC. Being the unit where data receipt and classification are carried out employing a Java application, the PC operates with a 3.07 GHz processor and 1.86 GB of RAM.

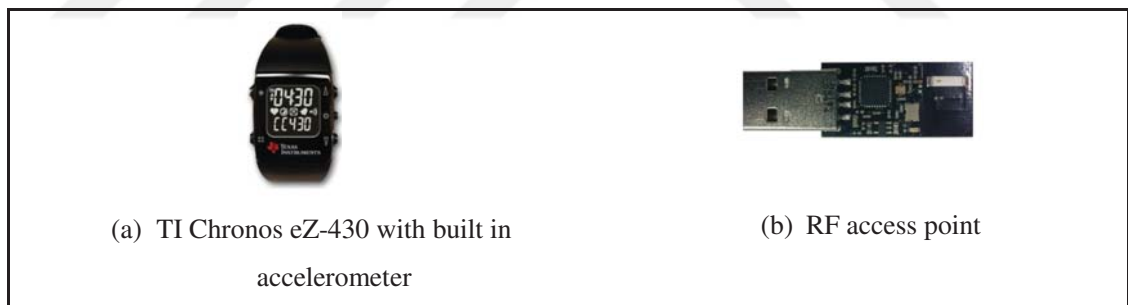


Figure 5.1. Sensing device

5.2.2. Data Set Generation

Fifteen training samples per simple action and ten test samples per simple and composite action are acquired. Each chunk of data lasts for 2.7 s. A chunk corresponds to a simple action or the transition between the simple actions when the activity is composite. The raw sensor data related to simple actions *toothBrushing*, *drinking*, *turningKey*, *pouring* and *eating* are plotted in Figure 5.2-Figure 5.4. The plotted data belong to the training data set. Time axis whose dimension is denoted to be $1/f$ s in these plots represents a virtual notion of time. This means that time axis shows the sequence number of the acceleration vector in the form of $\langle X, Y \text{ and } Z \rangle$ triple; X, Y and Z being acceleration values along corresponding

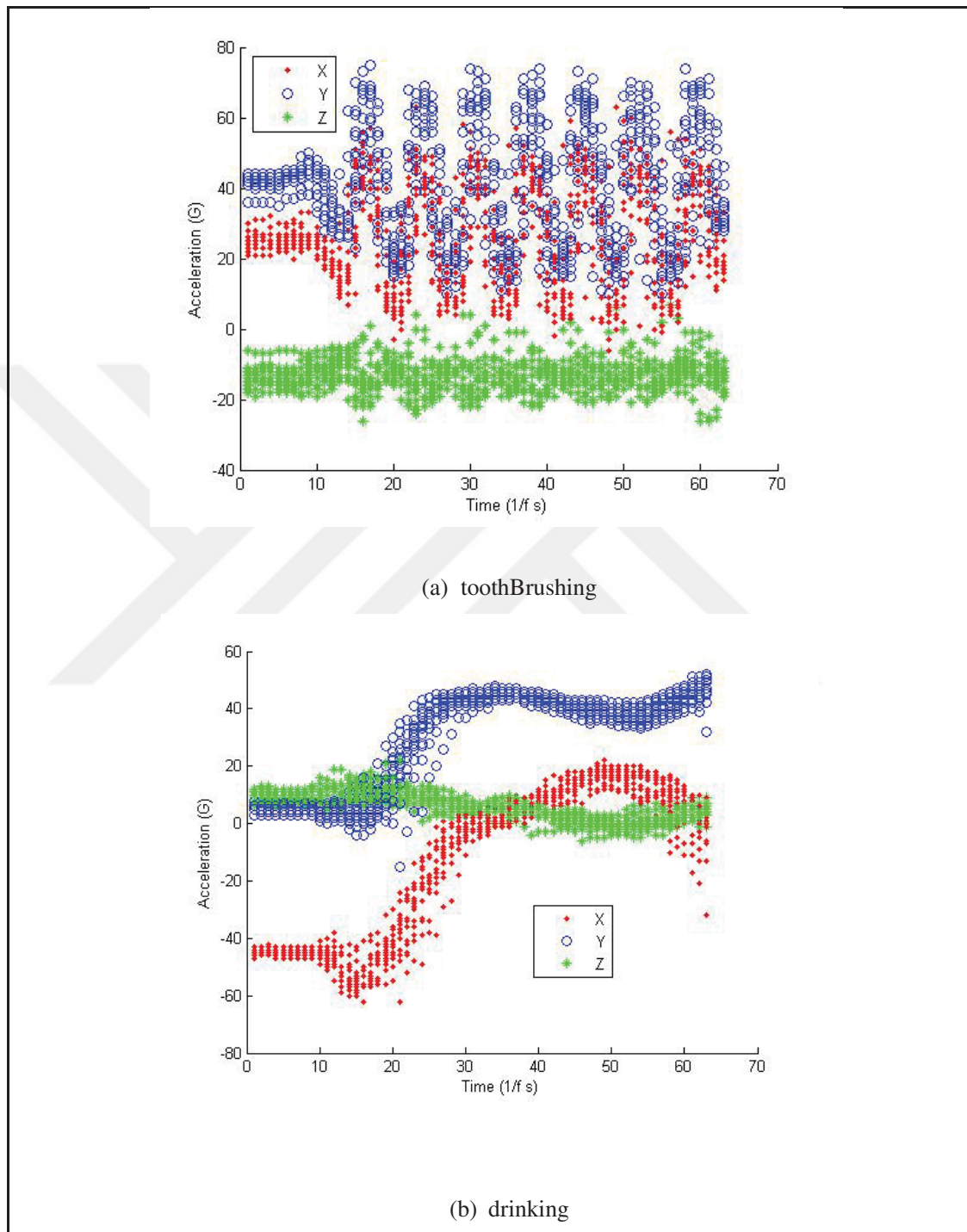


Figure 5.2. Raw sensor data for toothBrushing and drinking

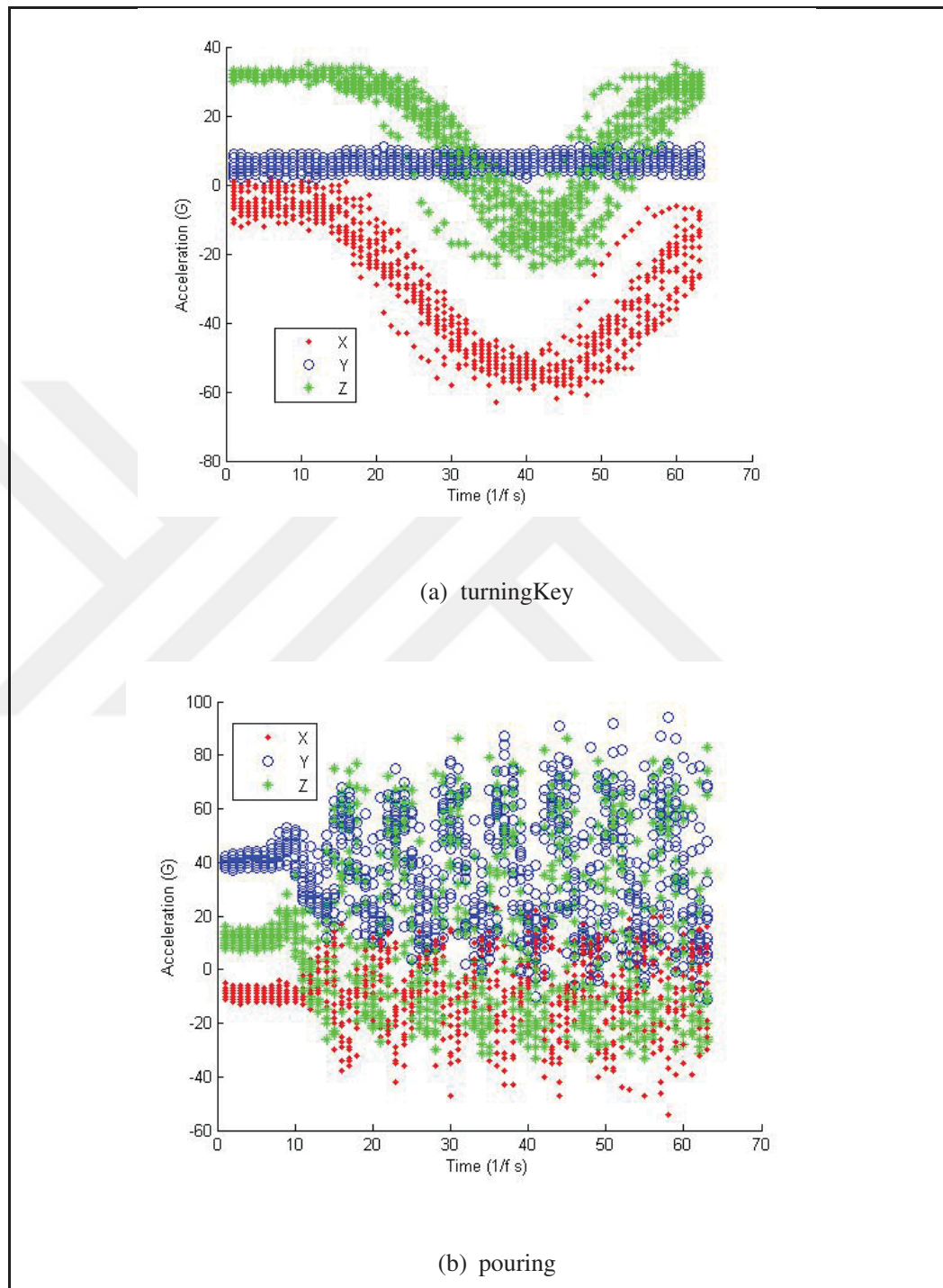


Figure 5.3. Raw sensor data for turningKey and pouring

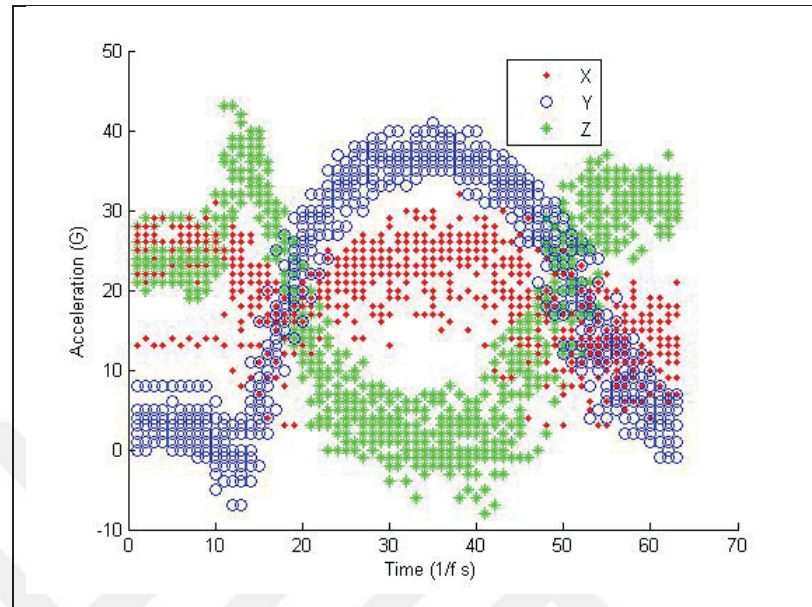


Figure 5.4. Raw sensor data for eating

axes. Though time axis designates sequence number, it can be regarded as a way of representing time, considering a vector with sNo_{i+1} arrives at the classification unit $1/f$ s later than the vector with sNo_i , where f is the transmission frequency. All real time continuous monitoring experiments are implemented with $f=23.26$ Hz. Mentioned transmission frequency value is calculated by averaging the number of vectors across training data set for a simple action, containing 15 training samples. Figure 5.5 illustrates the histogram related to the data points from which f is derived. Horizontal axis indicates the number of 3D vectors acquired as a sample whereas the vertical axis shows how many training samples contain corresponding number of vectors. Number of 3D vectors acquired as a sample are the data which manage to reach the classification unit.

5.3. REAL TESTBED ENVIRONMENT

In this chapter, testbed model is elaborated and testing methodology is reviewed, addressing the test categories. Then testbed setup is presented, emphasizing how the

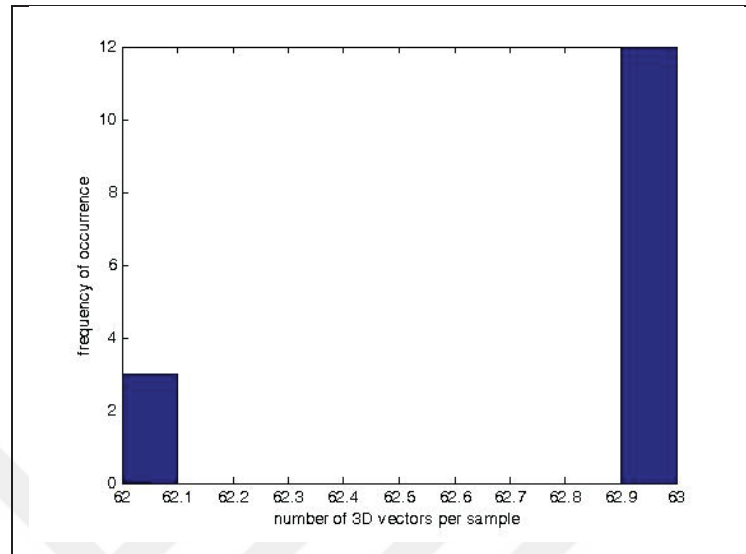


Figure 5.5. Histogram modelling the data set from which f is derived

actions are practised. Also, the application of testbed is explained, illustrating the output of the training phase.

5.3.1. System Model

Experimental context is summarized in Figure 5.6. The subject sits by the table, wearing the sensor on the right wrist. He uses an auxiliary object depending on the action being performed. The watch (TI Chronos ez430), incorporating the sensor, wirelessly communicates with the USB access point, which is connected to the PC. The watch with sensing and transmitting capability comprises the data acquisition unit whereas the PC is the module where knowledge extraction takes place.

5.3.2. Methodology

We tested our prototype with two kinds of tests named as T_iP_i and T_iP_j tests. T_iP_i tests are for evaluating intra-person classification accuracy, meaning that system predicts activities of the person who provides the training data. T_iP_j tests are for assessing inter-person classification accuracy, which means the system is trained with the training data acquired

from person i and predicts activities of another person j . Apart from classification accuracy, real time prediction delay is also measured.



Figure 5.6. Testbed overview

5.3.3. Testbed Setup

Training and test data are acquired in an office setting. The subject sits at a table to perform the actions with a single sensor worn on right wrist. A glass, a knife, which are made of plastic, a punched pocket and a key, shown in Figure 5.7, are used to perform the actions. As shown in Figure 5.8, *drinking* action is composed of taking the glass from the table, carrying it upwards then leaving it back to the table. When the glass is at the closest point to the mouth, the head slightly leans backwards. The time allocated for a chunk expires when the hand is hooked in the air during the downwards movement.



Figure 5.7. Auxiliary objects used in performing the activities

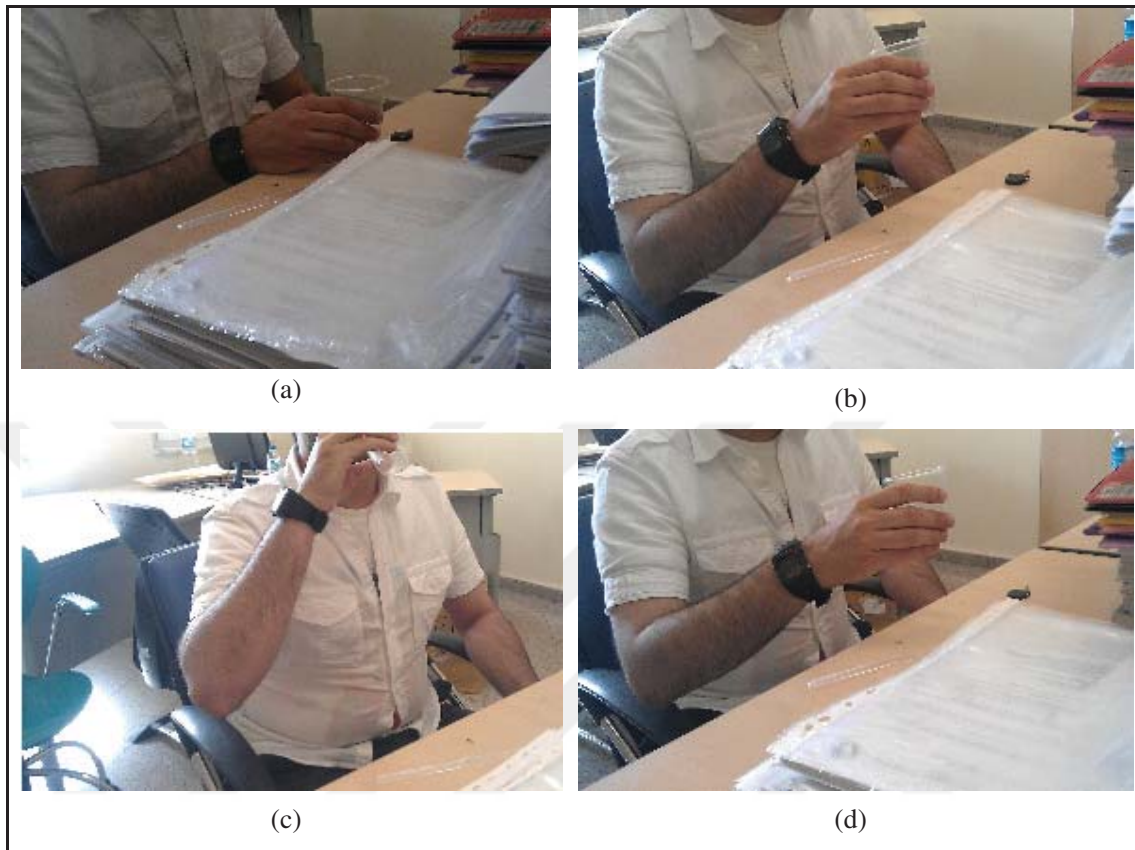


Figure 5.8. Drinking action as the combination of (a), (b), (c) and (d) with given order

Pouring action, illustrated in Figure 5.9, is shaking a punched pocket up and down with a regular rhythm. The subject, who provides the training data, reports that he performs the upwards and downwards shaking of the punched pocket seven or eight times at a chunk period. *Eating* action, shown in Figure 5.10, is carrying a knife upwards, and then downwards to the point from which it was picked. *TurningKey* action, illustrated in Figure 5.11, is turning a key placed on the table to the right and then to the left to reach back to

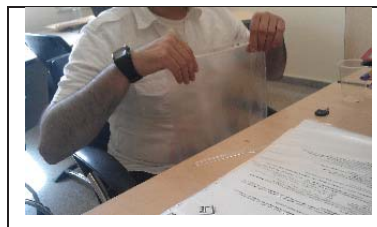


Figure 5.9. Pouring action

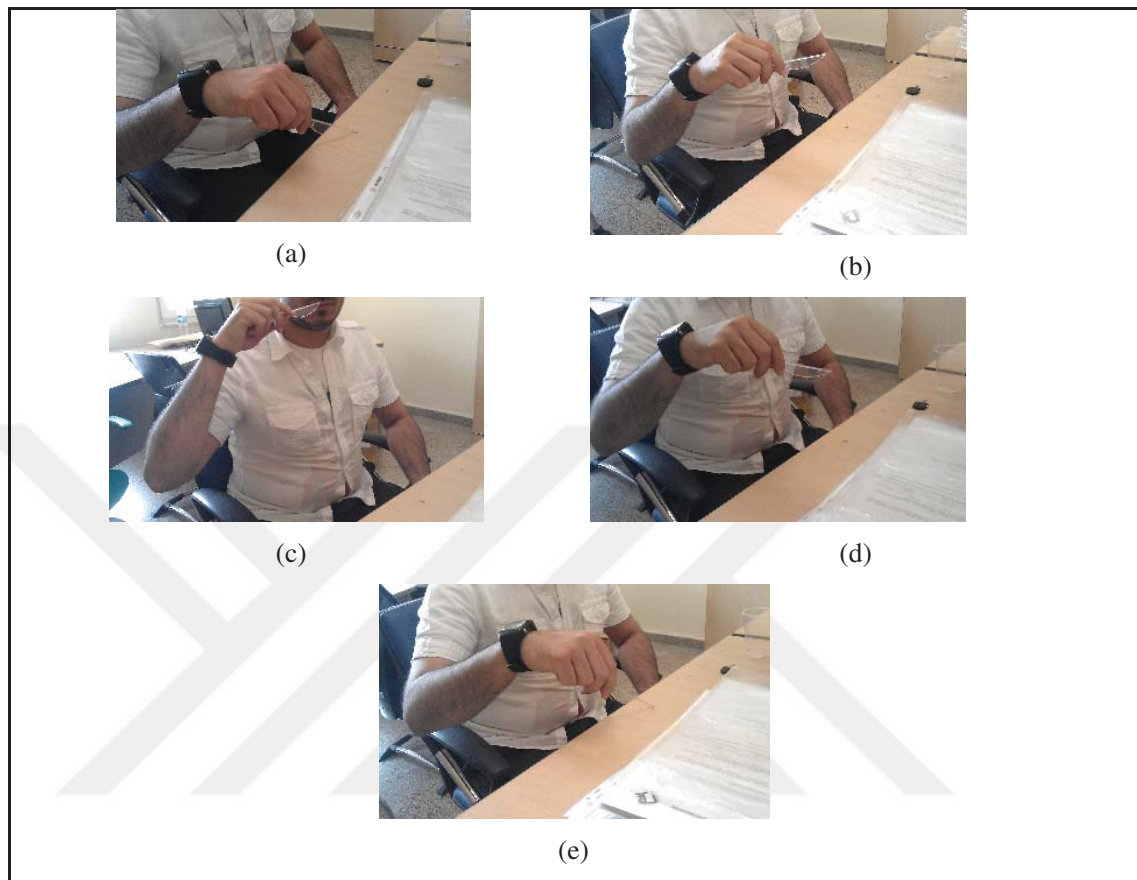


Figure 5.10. Eating action as the combination of (a), (b), (c), (d) and (e) with given order



Figure 5.11. TurningKey action

the initial position. *ToothBrushing*, illustrated in Figure 5.12, is shaking the knife upwards and downwards right in front of the mouth, considering the knife as a toothbrush. The



Figure 5.12. ToothBrushing action

subject providing the training data reports that he performs the upwards and downwards shaking of the knife seven or eight times at a chunk period.

5.3.4. Examination

Figure 5.13 to Figure 5.15 illustrate plots related to training output for the simple actions *toothBrushing*, *drinking*, *turningKey*, *pouring* and *eating*. Since training phase is also the feature extraction phase, the data points shown in these plots are the features generated for simple actions. A set of features with five elements are generated per training sample, which constitute the simple action patterns. Observing the patterns are close to each other, we can conclude that our training and feature extraction modules yield consistent activity models.

5.4. DISCUSSION

Using the testbed we set up, we verify that training module of RT-CAM can produce discriminative features. Also, we observe that RT-CAM is applicable for the ambient assisted living scenario we formed.

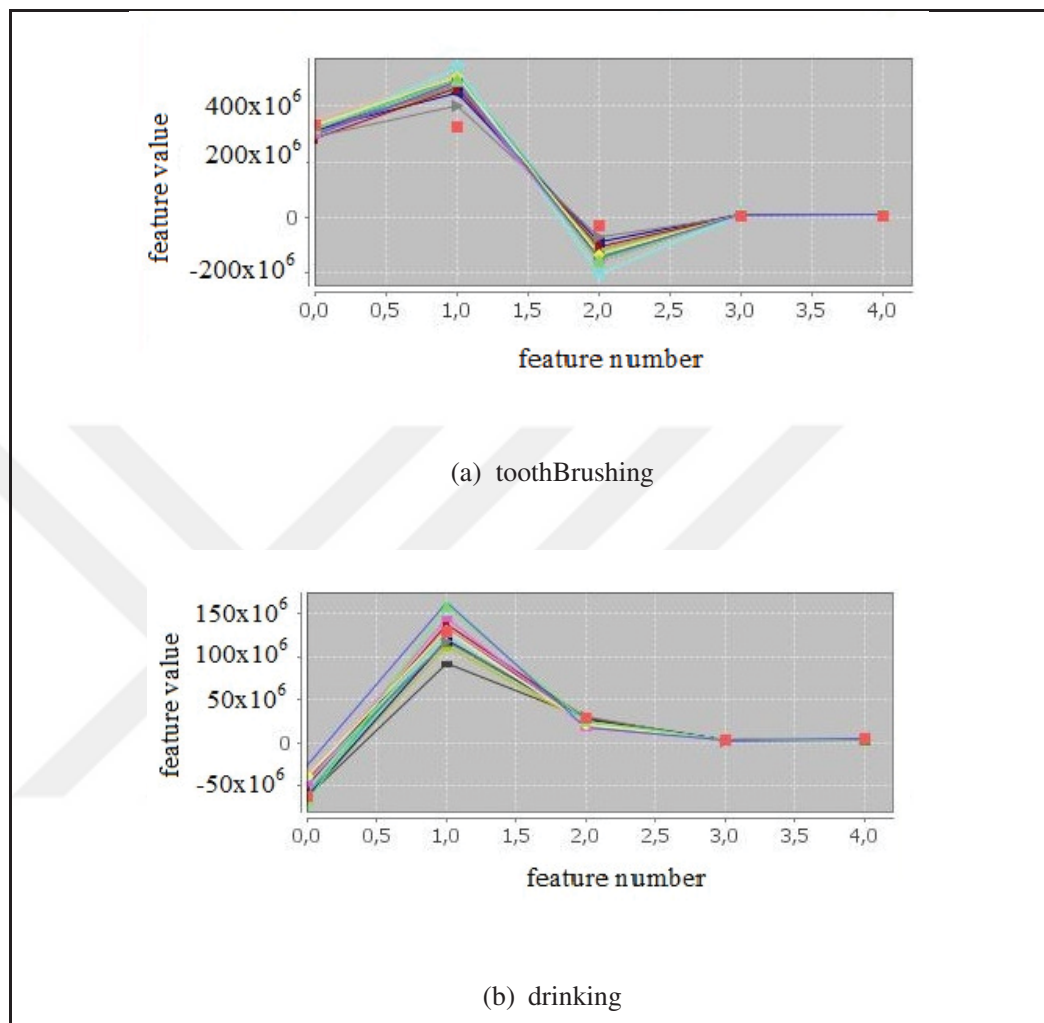


Figure 5.13. Training output for toothBrushing and drinking

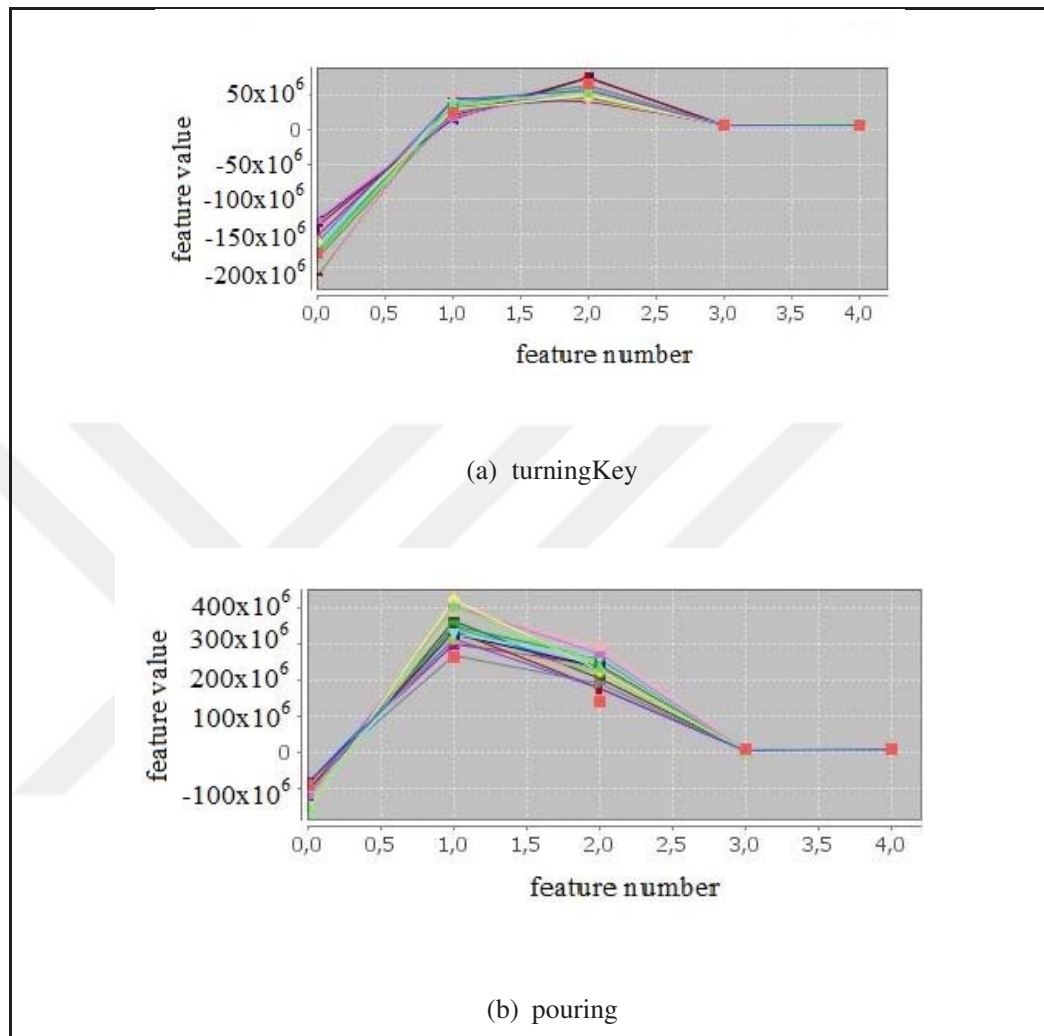


Figure 5.14. Training output for turningKey and pouring

5.5. SUMMARY

We overviewed the ambient assisted living scenario for which we defined our action set. Then, we analyzed the sensor data, presenting the tools we used for data collection and processing, followed by the description of data set generation process. Finally, we described our testbed and examined how our proposed method processes the experimental data.

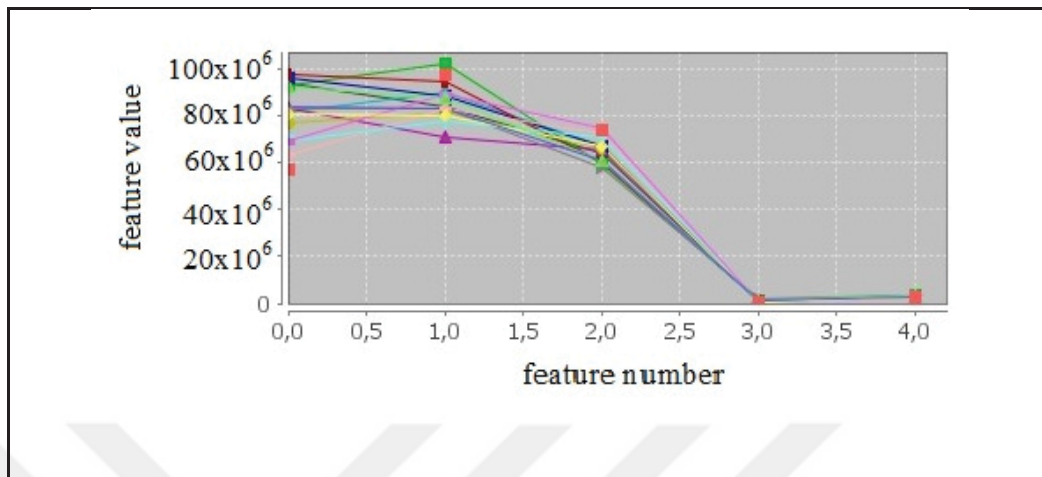


Figure 5.15. Training output for eating

6. PERFORMANCE ANALYSIS & EVALUATION

In this chapter, we present results obtained upon testing RT-CAM. We evaluate our system in terms of intra-person accuracy, inter-person accuracy and real time delay. We test for accuracy in real-time classification for two metrics:

- **Simple action accuracy (SA):** Correct recognition rate for simple actions
- **Composite action accuracy (CA):** Correct recognition rate for composite actions as a whole. A composite action is recognized correctly if all simple actions forming that composite action are successfully detected.

We tested RT-CAM across four subjects, one female and three male, being numbered from 1 to 4 and referred with these numbers throughout the text. One of the male subjects is the training subject, who is subject 1, and the female subject is the subject 3. Male subjects, though their physical profiles are similar, can be ordered from tallest to shortest as subjects 1, 2 and 4 and they are in the same age range. Detailed results of T_1P_1 tests regarding accuracy and real time delay can be seen in Appendix B and C respectively.

Since there are numerous real time tests to be maintained, a testing tool is programmed so that errors resulting from manual maintenance can be eliminated. However, using this tool costs 70-90 minute setup for each testing session and a group of 10 tests can be acquired in approximately 30 minutes. For subjects to practice until they can correctly imitate the actions, even more time is necessary. Due to the fact that tasks mentioned can not be completed at a single session, testing phase spreads across many days, leading to a great number of sessions. Since acquisition of test samples is such a time consuming operation, the number of tests presented is restricted.

6.1. T_1P_1 DETECTION ACCURACY

T_1P_1 is the term we use to formalize intra-person accuracy. It means that training data are acquired from subject i and the model is used to predict and classify the activities of

subject i as well. We carry out T_iP_i tests on subject 1, meaning that our intra-person tests are formalized as T_1P_1 .

6.1.1. Simple Action Accuracy (SA)

Recognition accuracy for simple actions are shown in Table 6.1. *Number of tests* column indicates the number of samples used to calculate accuracy. We also carried out 10 tests on *eating* action, obtaining 80% accuracy, but we excluded it from our analysis. Because of the nature of this action, a single sensor on right wrist is not enough to produce accurate results though we obtained 80% accuracy. Actions *pouring* and *turningKey* are detected with lower accuracies than *eating* but they are not excluded since number of tests for *pouring* and *turningKey* are sufficient. The rest of the analysis is based on the actions shown in Table 6.1 and their combinations.

Table 6.1. Real time classification accuracy of simple actions (T_iP_i)

Simple action	Accuracy (%)	Number of tests
toothBrushing	100	130
drinking	100	130
pouring	61.82	110
turningKey	72.73	110

Composite actions we tested include the simple actions except *eating* and the accuracy for these simple actions is calculated incorporating the successful detection accuracy of the simple actions within the composite action they constitute.

6.1.2. Composite Action Accuracy (CA)

Table 6.2 shows the accuracy of recognizing composite actions with several number of transitions. Each composite action is repeated ten times to form the test data set and the

Table 6.2. Recognition accuracy of composite actions (T_iP_i)

Number of transitions	Composite action	Accuracy (%)	Number of tests
1	drinking_pouring	60	10
	drinking_toothBrushing	100	10
	drinking_turningKey	60	10
	pouring_turningKey	80	10
	toothBrushing_drinking	100	10
	toothBrushing_pouring	100	10
	toothBrushing_turningKey	90	10
	turningKey_pouring	40	10
2	drinking_toothBrushing_pouring	80	10
	drinking_toothBrushing_turningKey	70	10
	toothBrushing_drinking_pouring	30	10
	toothBrushing_drinking_turningKey	70	10
3	drinking_toothBrushing_pouring_turningKey	20	10
	drinking_toothBrushing_turningKey_pouring	20	10
	toothBrushing_drinking_pouring_turningKey	70	10
	toothBrushing_drinking_turningKey_pouring	30	10

shown results are the mean of ten tests. Comparing the T_iP_i results related to *pouring* action and the composite action *drinking_toothBrushing_pouring* which contains *pouring*, it can be observed that 61.82% and 80% accuracy are achieved for them respectively. The fact that a composite action results in a better accuracy than a simple action it contains may seem contradictory. However, as explained before, accuracy for simple actions are calculated including the successful detection accuracy of simple actions within the composite actions they constitute.

Considering Table 6.2, we can observe that if a composite action contains one of *pouring* and *turningKey* actions, its accuracy deteriorates compared to its subactions where *pouring* and *turningKey* are excluded, which agrees with the results shown in Table 6.1. The fact that *toothBrushing_drinking_pouring* and *toothBrushing_drinking_pouring_turningKey* are detected with 30% and 70% accuracy respectively is contradictory in this aspect.

However, even the same person can not perform the actions the same way all the time, resulting in such a contradictory observation.

6.2. T_iP_j DETECTION ACCURACY

T_iP_j is the term we use to formalize inter-person classification tests. For the tests in this category, we evaluate RT-CAM, acquiring training data from subject 1 and classifying the activities of subjects 2, 3 and 4, formalizing our T_iP_j tests as T₁P₂, T₁P₃, and T₁P₄ respectively.

6.2.1. Simple Action Accuracy (SA)

Successful recognition accuracy of simple actions are shown in Table 6.3. Height of the subject also affects successful recognition rate according to our observations. Because *turningKey* action was poorly recognized in T₁P₄ tests and successful recognition rate for this action is observed to increase as can be seen in Table 6.3 when subject 4 performs *turningKey* action on a higher chair as training subject is taller than subject 4. The column named as # indicates number of tests acquired to obtain the corresponding accuracy value. Also in Table 6.3, normalized results calculated for each simple action are shown as **norm**. Normalized results are determined using the formula given in Equation (6.1). Test categories T₁P₂, T₁P₃ and T₁P₄ are shown as c₁, c₂ and c₃ respectively. Also, *a* and *n* designate accuracy and number of tests respectively for the corresponding test categories. If the terms in the formula match the group of tests marked as “Not tested” in Table 6.3, those terms are ignored in normalized result calculation. Observing the normalized results, we can say that RT-CAM can predict the activities of different people in a training data independant way.

$$\text{norm} = \frac{1}{n_{c_1} + n_{c_2} + n_{c_3}} (a_{c_1} n_{c_1} + a_{c_2} n_{c_2} + a_{c_3} n_{c_3}) \quad (6.1)$$

6.2.1.1. Discussion

For SA results, T₁P₂ tests are consistent with T₁P₁ case. Though *turningKey* results in higher accuracy with T₁P₂ than T₁P₁, it is still a similar value to what is generated in T₁P₁

Table 6.3. Recognition accuracy of simple actions (T_iP_j)

Action	T_1P_2		T_1P_3		T_1P_4		norm
	Accuracy (%)	#	Accuracy (%)	#	Accuracy (%)	#	
toothBrushing	100	50	42.86	21	100	5	84.2
drinking	82.86	70	100	11	57	7	82.9
pouring	100	10	100	2	Not tested		100.0
turningKey	80	20	Not tested		50	8	71.4

case. The fact that *pouring* results in a much higher accuracy with T_1P_2 than T_1P_1 is contradictory. Nevertheless, test data are collected from subject 1 a couple of weeks later than acquisition of training data. Therefore, subject 1 may have forgotten the exact way of how he performed the actions during training data acquisition, which can explain the reason why subject 2 performs the actions more similarly to the training samples than training subject.

In T_1P_3 tests, *toothBrushing* results in much lower accuracy than *drinking* though these two actions are expected to generate high accuracy yielding similar values as the T_1P_1 case is considered. This could be explained regarding that *toothBrushing* is performed differently from the way the training subject performs during training data acquisition. The fact that *pouring* results in such higher accuracy than T_1P_1 case could result from either the number of tests being small or training subject having forgotten the exact way he performed this action.

T_1P_4 results agrees with T_1P_1 category since in the case of subject 4, *toothBrushing* is detected with high accuracy and *turningKey* is detected with accuracy deteriorated. The reason why *drinking* is detected with such a poor accuracy compared to *toothBrushing* in T_1P_4 category looks contradictory but subject 4 had much difficulty in performing the exact action which could be the reason for it.

6.2.2. Composite Action Accuracy (CA)

Table 6.4 shows inter-person recognition accuracy for composite actions. Subject 2 is exposed to testing before and after watching the video recordings of training subject, hence T_1P_2 tests are marked as A and B, which indicate after-watch and before-watch cases

Table 6.4. Composite action accuracy results (T_iP_j)

Number of transitions	Test category	Composite action	Accuracy (%)	Number of tests
1	T_1P_2	toothBrushing_drinking	B:0 A:70	B:10 A:10
	T_1P_2	drinking_toothBrushing	B:0 A:80	B:10 A:10
	T_1P_3	toothBrushing_drinking	45	11
2	T_1P_2	drinking_toothBrushing_pouring	B:0 A:80	B:10 A:10
	T_1P_2	drinking_toothBrushing_turningKey	B:0 A:80	B:10 A:10
	T_1P_2	toothBrushing_drinking_turningKey	B:0 A:60	B:10 A:10

respectively. The fact that after-watch success rate is higher than before-watch success rate is that same action is performed differently by subject 1 and 2 in the before-watch case. Observing that before-watch case results in false detection and after-watch results are high, we can say that our method is affected by how the activities are performed. As mentioned in Section 6.2.1, height of the subject is observed to affect successful recognition rate according to our observations. Same effect is observed for *turningKey* tests in T_1P_2 category as training subject is taller than subject 2. Because *turningKey* action was poorly recognized in T_1P_2 tests, subject 2 is made to perform *turningKey* action on a higher chair, hence successful recognition rate for this action is observed to increase as can be seen in the field marked as A in Table 6.4.

6.2.2.1. Discussion

It is expected that T_1P_1 accuracy becomes lower compared to T_1P_1 accuracy since different people are monitored. Hence, T_1P_2 results are consistent except *drinking_toothBrushing_pouring* and *drinking_toothBrushing_turningKey* which are observed to be the same and a little higher compared to their T_1P_1 counterparts respectively. However, the difference mentioned is not extreme, hence a contradiction is not introduced. In the case of T_1P_3 tests, the action *toothBrushing_drinking* is expected to be detected with an accuracy around the obtained value since T_1P_3 tests for *toothBrushing* yields a similar value as can be seen in Table 6.3. Comparing the CA results and values illustrated in Table 6.3 related to T_1P_2 , it can be observed that CA values do not exceed the SA values for simple actions contained in the composite actions yielding the corresponding CA values which agrees with our expectations.

6.3. REAL TIME OVERHEAD

Real time overhead introduced by RT-CAM is illustrated in Figure 6.1 for simple actions and in Figure 6.2 for composite actions, bars showing the average real time overhead and lines indicating the confidence intervals with 95% confidence level. Indicating time taken for knowledge discovery, real time overhead values illustrated in these plots belong to the tests whose accuracy results are shown in Sections 6.1 and 6.2 where duration of each segment is 2.7 s. Figure 6.2 evaluates real time overhead separately for each type of segment where a segment type is either a simple action or a transition.

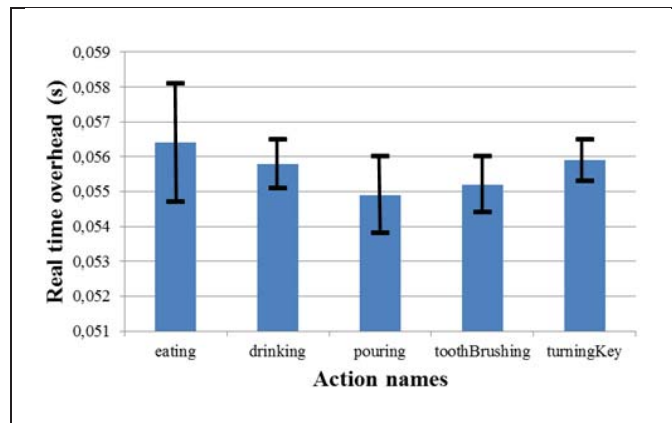


Figure 6.1. Real time overhead for simple actions

S_i and T_i designate i th simple action and i th transition in the composite action respectively. The real time overhead is almost the same across simple actions and segment types as given in Figure 6.1 and Figure 6.2 respectively.

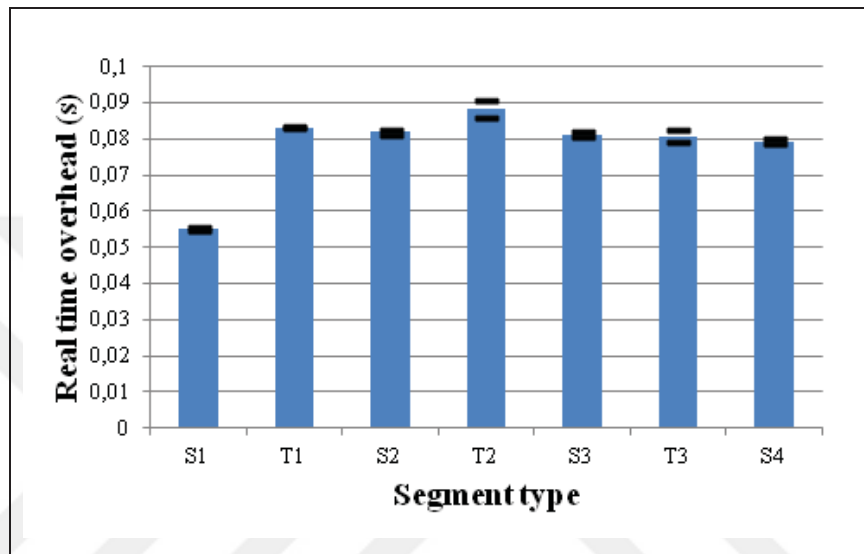


Figure 6.2. Real time overhead for composite actions

6.4. COMPARISON OF RT-CAM

Finding a study which is evaluated on top of a setting which is equal to ours in terms of sensor count, sensor locations, etc., is difficult and to present a comparison, we have to choose a work which we find closest to our study. Hence, we compare RT-CAM with the work by Aiello et. al. [62] as shown in Table 6.5. Their work employs a pair of 3D accelerometers, one of them positioned on the waist and the other one located on the thigh whereas RT-CAM utilizes only a single 3D accelerometer on the right wrist. To create a better basis of comparison, we tested RT-CAM with the actions shown in Table 6.5 which are also tested in the work by Aiello et. al. [62]. The results regarding their work belong to their experiments where transition duration is less than or equal to one second.

Table 6.5. Comparison of RT-CAM with the work by Aiello et. al. [62]

Action name	Accuracy (%)			
	RT-CAM/single sensor	Aiello et. al. [62]/two sensors	SA ₁	SA ₂
lyingDown_sitting	100	70	100	100
sitting_lyingDown	50	0	100	50
sitting_standingStill	40	90	100	40
standingStill_sitting	30	0	30	90
walking_standingStill	0	0	60	0
standingStill_walking	50	100	50	90

The results belonging to RT-CAM are obtained with duration of each chunk being set to one second. We acquired 15 training samples and 10 test samples for each action. SA₁ and SA₂ columns denote the SA accuracy of simple action 1 and 2 respectively for RT-CAM. Namely, if the composite action is *lyingDown_sitting*, SA₁ and SA₂ correspond to *lyingDown* and *sitting* respectively. The comparison between RT-CAM and the work by Aiello et. al. [62] yields that in actions which are easier to distinguish with waist or thigh sensors, their method performs better as expected. The only case which contradicts with our expectations is *standing_sitting* action. RT-CAM performs better in *standing_sitting* action which is the point contradicting with our expectations but this is still a valid case. Because though RT-CAM performs better in *standing_sitting* action, it generates a similar result to what it produces for the reverse version which is *sitting_standing* as expected.

6.5. SUMMARY

For evaluating RT-CAM, we selected an action set, which is suitable for being captured via wrist motions. We showed that the selected activities can be distinguished in real time though they are similar to each other. The successful detection rate is influenced by the way activities are performed and the physical profile of the subjects such as height. The proposed method achieves activity recognition in reasonable amount of time, yielding a light-weight activity recognizer.

7. CONCLUSION AND FUTURE WORK

We proposed a method, which we call RT-CAM, addressing real time continuous monitoring of activities which are combinations of a set of hand oriented actions. We modelled the action set as the simple and composite actions and formalized the transition detection problem as identifying the sequence of simple actions within a composite action which they constitute. We carried out data collection using a 3D accelerometer to present a non-invasive solution. RT-CAM carries out transition detection without being trained with patterns of transitions. Also, RT-CAM does not require large number of subjects who should provide training data. We showed that the selected activities can be distinguished in real time though they are similar to each other. All these features of RT-CAM make it an applicable solution in real time continuous activity monitoring.

As the future work, we will be addressing the following points: In the literature, various ways of windowing is studied [63] for partitioning the continuous sensor input, in other words, for determining the start and end point for simple actions within a data stream containing composite actions. Contrary to these studies, we are after a method to abandon the importance of windowing by means of establishing a robust way of simple activity detection. By doing that, it is expected to achieve correct identification of segments no matter how improperly the window size is selected. To evaluate our system in this aspect, we will test our system with the subject performing actions in his natural pace and not trying to fit in a timely restriction.

Another case which we will study as the future work is training AMS without human supervision since after deployment, people's needs may change or extending their functionality may be desired. AMS should be able to extract knowledge even when training data include unnecessary, perhaps misleading information since it is expected to train itself within the home environment of the end user without the support of a supervising person. Therefore, we will improve RT-CAM targeting this requirement.

Chunk duration is the same for training and prediction data. Hence, it can be questioned whether it is possible to handle the case of people performing the actions at a different

speed compared to the training data. To modify our model for handling this case, we can avoid explicit segmentation by doing the following: A threshold interval can be generated for each action in the training set, applying our algorithm on the training samples. Then, the real time activity data are partitioned into much smaller chunks, being a couple of vectors long only. Then, each segment is exposed to our algorithm to see whether it corresponds to the threshold interval. As the real time activity data are received, they are appended to previously processed chunks and the newly obtained chunk is again exposed to our algorithm for being evaluated in terms of threshold compliance. Segment checking lasts until the segment is detected to be in the threshold interval for a simple action which determines its type. Naming the identified chunk as chunk i , the second step is to identify chunk $i+1$ which could be either a transition or continuation of chunk i . If it is the continuation of the previous chunk, it should be remaining within the specified threshold interval. If it is a transition, since transitions are not stored in the training set, difference between the value our algorithm generates for it as the threshold and threshold values for the simple actions in the training set should become larger and larger as real time activity data are continuously received. When the difference stops becoming larger and starts decreasing, then transition is over and chunk $i+2$ is identified in a similar way to that of chunk i . It is obvious that successful operation of this alternative is possible with obtaining unique threshold intervals for simple actions and thresholds responding to situations described as expected which requires further research and development.

Our model infers the existence and type of a transition based on the simple actions which are predecessor and successor of the transitions. Considering chunks i , $i+1$ and $i+2$, where i is equal to 1, if chunk i and chunk $i+2$ are detected to be different, then chunk $i+1$ is identified to be a transition. It may be questioned that if the person performs the same action twice, making the explicit classification of chunk $i+1$ essential. However, since our aim is identifying transitions, we do not need to know the type of chunk $i+1$ as long as its type is identical to that of chunk i . Whether chunk $i+2$ is a transition or simple action should also be determined but this issue will be handled with the enhancements, mentioned in the previous paragraph, are carried out.

REFERENCES

1. Brusey, J., R. Rednic, E. I. Gaura, J. Kemp and N. Poole, "Postural Activity Monitoring for Increasing Safety in Bomb Disposal Missions", *Measurement Science and Technology*, Vol. 20, pp. 75204-75215, 2009.
2. Selvabala, V. S. N. and A. B. Ganesh, "Implementation of Wireless Sensor Network Based Human Fall Detection System", *Procedia Engineering*, Vol. 30, pp. 767-773, 2012.
3. Amft, O. and G. Tröster, "Recognition of Dietary Activity Events Using On-body Sensors", *Artificial Intelligence in Medicine*, Vol. 42, pp. 121-136, 2008.
4. Zwartjes, D.G.M., T. Heida, J.P.P. van Vugt, J.A.G. Geelen and P.H. Veltink, "Ambulatory Monitoring of Activities and Motor Symptoms in Parkinson's Disease," *Biomedical Engineering*, IEEE Transactions, Vol. 57, pp. 2778-2786, 2010.
5. Sazonov, E.S., G. Fulk, N. Sazonova and S. Schuckers, "Automatic Recognition of Postures and Activities in Stroke Patients," *Engineering in Medicine and Biology Society, 2009, EMBS 2009, Annual International Conference of the IEEE*, Minneapolis, Minnesota, USA, 2 September-6 September 2009, pp. 2200-2203, 2009.
6. Sazonov, E.S., G. Fulk, J. Hill, Y. Schutz and R. Browning, "Monitoring of Posture Allocations and Activities by A Shoe-based Wearable Sensor", *IEEE Transactions on Biomedical Engineering*, Vol. 58, pp. 983-990, 2011.
7. Warren, J.M., U. Ekelund, H. Besson, A. Mezzani, N. Geladas and L. Vanhees, "Assessment of Physical Activity - A Review of Methodologies with Reference to Epidemiological Research: A Report of the Exercise Physiology Section of the European Association of Cardiovascular Prevention and Rehabilitation", *European Journal of Cardiovascular Prevention and Rehabilitation*, Vol. 17, pp. 127-139, 2010.

8. Arcelus A., C. L. Herry, R. A. Goubran, F. Knoefel, H. Sveistrup and M. Bilodeau, "Determination of Sit-to-Stand Transfer Duration Using Bed and Floor Pressure Sequences", *Biomedical Engineering, IEEE Transactions*, Vol. 56, pp. 2485-2492, 2009.
9. Sivaraman V., A. Dhamdhare, H. Chen, A. Kurusingal and S. Grover, "An Experimental Study of Wireless Connectivity and Routing in Ad Hoc Sensor Networks for Real-Time Soccer Player Monitoring", *Ad Hoc Networks*, Vol. 11, pp. 798-817, 2013.
10. Kasteren T. L. M. V., G. Englebienne and B. Krse, "An Activity Monitoring System for Elderly Care Using Generative and Discriminative Models", *Journal of Personal and Ubiquitous Computing*, Vol. 14, pp. 489-498, 2010.
11. Tolstikov A., X. Hong, J. Biswas, C. Nugent, L. Chen and G. Parente, "Comparison of Fusion Methods Based on DST and DBN in Human Activity Recognition", *Journal of Control Theory and Applications*, Vol. 9, pp. 18-27, 2011.
12. Yang J., J. Lee and J. Choi, "Activity Recognition Based on RFID Object Usage for Smart Mobile Devices", *Journal of Computer Science and Technology*, Vol. 26, pp. 239-246, 2011.
13. Sarkar J., L. T. Vinh, Y. K. Lee and S. Lee, "GPARS: A General-purpose Activity Recognition System", *Applied Intelligence*, Vol. 35, pp. 242-259, 2011.
14. Jihoon, H. and T. Ohtsuki, "A State Classification Method Based on Space-Time Signal Processing Using SVM for Wireless Monitoring Systems", *Personal Indoor and Mobile Radio Communications (PIMRC), 2011 IEEE 22nd International Symposium*, Toronto, 11 September-14 September 2011, pp. 2229-2233, 2011.
15. Lara, O. and M. Labrador, "A Survey on Human Activity Recognition Using Wearable Sensors", *Communications Surveys & Tutorials IEEE*, Vol. PP, pp. 1 - 18, 2012.

16. Turaga, P., R. Chellappa, V.S. Subrahmanian and O. Udrea, "Machine Recognition of Human Activities: A Survey", *Circuits and Systems for Video Technology, IEEE Transactions*, Vol. 18, pp. 1473-1488, 2008.
17. Candamo, J., M. Shreve, D.B. Goldgof, D.B. Sapper and R. Kasturi, "Understanding Transit Scenes: A Survey on Human Behavior-Recognition Algorithms", *Intelligent Transportation Systems, IEEE Transactions*, Vol. 11, pp. 206-224, 2010.
18. Joseph, C.N., S. Kokulakumaran, K. Srijevantham, A. Thusyanthan, C. Gunasekara and C.D. Gamage, "A Framework for Whole-Body Gesture Recognition from Video Feeds", *Industrial and Information Systems (ICIIS), International Conference*, India, 29 July-1 August 2010, pp. 430-435.
19. Ahad, M., J.K. Tan, H.S. Kim and S. Ishikawa, "Human Activity Recognition: Various Paradigms", *Control, Automation and Systems, 2008, ICCAS 2008, International Conference*, Seoul, Korea, 14 October-17 October 2008, pp. 1896-1901.
20. Clarke-Moloney, M., A. Godfrey, V. O'Connor, H. Meagher, P.E. Burke, E.G. Kavanagh, P.A. Grace and G.M. Lyons, "Mobility in Patients with Venous Leg Ulceration", *European Journal of Vascular and Endovascular Surgery*, Vol. 33, pp. 488-493, 2007.
21. Bourke, A.K., J.V. O'Brien and G.M. Lyons, "Evaluation of a Threshold-based Tri-axial Accelerometer Fall Detection Algorithm", *Gait & Posture*, Vol. 26, pp. 194-199, 2007.
22. Bourke, A. K., K. J. O'Donovan and G. ÓLaighin, "The Identification of Vertical Velocity Profiles Using an Inertial Sensor to Investigate Pre-impact Detection of Falls", *Medical Engineering & Physics*, Vol. 30, pp. 937-946, 2008.
23. Godfrey, A., A. K. Bourke, G. M. Ólaighin, P. van de Ven and J. Nelson, "Activity Classification Using a Single Chest Mounted Tri-axial Accelerometer", *Medical Engineering & Physics*, Vol. 33, pp. 1127-1135, 2011.

24. Moore, S. T., H. G. MacDougall, J. M. Gracies, H. S. Cohen and W. G. Ondo, "Long-term Monitoring of Gait in Parkinson's Disease", *Gait & Posture*, Vol. 26, pp. 200-207, 2007.
25. Culhane, K. M., G. M. Lyons, D. Hilton, P. A. Grace and D. Lyons, "Long-term Mobility Monitoring of Older Adults Using Accelerometers in A Clinical Environment", *Clinical Rehabilitation*, Vol. 18, pp. 335-343, 2004.
26. Leonard, M., A. Godfrey, M. Silberhorn, M. Conroy, S. Donnelly, D. Meagher and G. Ólaighin, "Motion Analysis in Delirium: A Novel Method of Clarifying Motoric Subtypes", *Neurocase*, Vol. 13, pp. 272-277, 2007.
27. Kangas, M., A. Konttila, P. Lindgren, I. Winblad and T. Jämsä, "Comparison of Low-complexity Fall Detection Algorithms for Body Attached Accelerometers", *Gait & Posture*, Vol. 28, pp. 285-291, 2008.
28. Karantonis, D. M., M. R. Narayanan, M. Mathie, N. H. Lovell and B. G. Celler, "Implementation of a Real-time Human Movement Classifier Using a Triaxial Accelerometer for Ambulatory Monitoring", *Information Technology in Biomedicine, IEEE Transactions*, Vol. 10, pp. 156-167, 2006.
29. Lyons, G. M., K. M. Culhane, D. Hilton, P. A. Grace and D. Lyons, "A Description of An Accelerometer-based Mobility Monitoring Technique", *Medical Engineering & Physics*, Vol. 27, pp. 497-504, 2005.
30. Godfrey, A., R. Conway, D. Meagher and G. Ólaighin, "Direct Measurement of Human Movement by Accelerometry", *Medical Engineering & Physics*, Vol. 30, pp. 1364-1386, 2008.
31. Chan, M., D. Estève, J. Y. Fourniols, C. Escriba and E. Campo, "Smart Wearable Systems: Current Status and Future Challenges", *Artificial Intelligence in Medicine*, Vol. 56, pp. 137-156, 2012.

32. Baek, J., G. Lee, W. Park and B. J. Yun, "Accelerometer Signal Processing for User Activity Detection", in M. G. Negoita, R. J. Howlett, L. C. Jain (eds.), *Knowledge-Based Intelligent Information and Engineering Systems*, pp. 610-617, Springer Berlin Heidelberg, 2004.
33. Maurer, U., A. Smailagic, D. P. Siewiorek and M. Deisher, "Activity Recognition and Monitoring Using Multiple Sensors on Different Body Positions", *Wearable and Implantable Body Sensor Networks, BSN 2006 International Workshop*, Cambridge, 3 April-5 April 2006, pp. 113-116, IEEE Computer Society, Washington, DC, USA, 2006.
34. Ravi, N., N. Dandekar, P. Mysore and M. L. Littman, "Activity Recognition from Accelerometer Data", *Proceedings of the National Conference on Artificial Intelligence*, Pittsburgh, Pennsylvania, 2005, Vol. 3, pp. 1541-1546, AAAI Press, 2005.
35. Nyan, M. N., F. E. H. Tay, K. H.W. Seah and Y.Y. Sitoh, "Classification of Gait Patterns in the Time-Frequency Domain", *Journal of Biomechanics*, Vol. 39, pp. 2647-2656, 2006.
36. Preece, S. J., J. Y. Goulermas, L. P. J. Kenney and D. Howard, "A Comparison of Feature Extraction Methods for the Classification of Dynamic Activities From Accelerometer Data", *Biomedical Engineering, IEEE Transactions*, Vol. 56, pp. 871-879, 2009.
37. Preece, S. J., J. Y. Goulermas, L. P. J. Kenney, D. Howard, K. Meijer and R. Crompton, "Activity Identification Using Body-Mounted Sensors—A Review of Classification Techniques", *Physiological Measurement*, Vol. 30, 2009.
38. Zhenyu, H., L. Zhibin, J. Lianwen, L. X. Zhen and J. C. Huang, "Weightlessness Feature - A Novel Feature for Single Tri-axial Accelerometer Based Activity Recognition", *Pattern Recognition, ICPR 2008, 19th International Conference*, Tampa, FL, 8 December-11 December 2008, pp. 1-4.

39. Mathie, M. J., A. C. F. Coster, N. H. Lovell and B. G. Celler, "Detection of Daily Physical Activities Using A Triaxial Accelerometer", *Medical and Biological Engineering and Computing*, Vol. 41, pp. 296-301, 2003.
40. Mantyjarvi, J., J. Himberg and T. Seppanen, "Recognizing Human Motion with Multiple Acceleration Sensors", *Systems, Man, and Cybernetics, IEEE International Conference*, Tucson, AZ, 7 October-10 October 2001, Vol. 2, pp. 747-752.
41. Pirttikangas, S., K. Fujinami and T. Nakajima, "Feature Selection and Activity Recognition from Wearable Sensors", in H. Y. Youn, M. Kim and H. Morikawa (eds.), *Ubiquitous Computing Systems*, pp. 516-527, Springer Berlin Heidelberg, 2006.
42. Engin, M. , S. Demirağ, E. Z. Engin, G. Çelebi, F. Ersan, E. Asena and Z. Çolakoğlu, "The Classification of Human Tremor Signals Using Artificial Neural Network", *Expert Systems with Applications*, Vol. 33, pp. 754-761, 2007.
43. Parkka, J., M. Ermes, P. Korpiäa, J. Mantyjarvi, J. Peltola and I. Korhonen, "Activity Classification Using Realistic Data From Wearable Sensors", *Information Technology in Biomedicine, IEEE Transactions*, Vol. 10, pp. 119-128, 2006.
44. Zhang, K., M. Sun, D. K. Lester, F. X. Pi-Sunyer, C. N. Boozer and R. W. Longman, "Assessment of Human Locomotion by Using An Insole Measurement System and Artificial Neural Networks", *Journal of Biomechanics*, Vol. 38, pp. 2276-2287, 2005.
45. Begg, R. and J. Kamruzzaman, "A Machine Learning Approach For Automated Recognition of Movement Patterns Using Basic, Kinetic and Kinematic Gait Data", *Journal of Biomechanics*, Vol. 38, pp. 401-408, 2005.
46. Parera, J., C. Angulo, A. Rodríguez-Molinero and J. Cabestany, "User Daily Activity Classification from Accelerometry Using Feature Selection and SVM", in J. Cabestany, F. Sandoval, A. Prieto and J. M. Corchado (eds.), *Bio-Inspired Systems: Computational and Ambient Intelligence*, pp. 1137-1144, Springer Berlin Heidelberg, 2009.

47. Jianxin, W., A. Osuntogun, T. Choudhury, M. Philipose and J. M. Rehg, "A Scalable Approach to Activity Recognition Based on Object Use", *Computer Vision, 2007, ICCV 2007, IEEE 11th International Conference*, Rio de Janeiro, 14 October-21 October 2007, pp. 1-8, 2007.
48. Minnen, D., S. Thad, E. Irfan and I. Charles, "Discovering Characteristic Actions from On-Body Sensor Data", *Proceedings of IEEE International Symposium on Wearable Computing*, 2006, pp. 11-18, 2006.
49. Demir, A. K., K. Irgan, Ş. Baydere and E. Demiray, "Transmitting Objects in Images with Service Differentiation Based Source Coding in Wireless Sensor Network", *IEEE/ACM IWCMC*, Sardinia, 1 July-5 July 2013, accepted.
50. Uslu, G., O. Altun, and S. Baydere, "A Bayesian Approach for Indoor Human Activity Monitoring", *Hybrid Intelligent Systems (HIS), 2011 11th International Conference*, Melacca, 5 December-8 December 2011, pp. 324 – 327, 2011.
51. Uslu, G., H. I. Dursunoglu, O. Altun, and S. Baydere, "Human Activity Monitoring with Wearable Sensors and Hybrid Classifiers", *International Journal of Computer Information Systems and Industrial Management Applications*, Vol. 5, pp. 345-353, 2013.
52. Uslu, G. and S. Baydere, "Support Vector Machine Based Activity Detection", *Signal Processing and Communications Applications Conference (SIU), 2013 21st*, Haspolat, 24 April-26 April 2013, pp. 1-4, 2013.
53. Najafi, B., K. Aminian, A. Paraschiv-Ionescu, F. Loew, C. J. Bula and P. Robert, "Ambulatory System for Human Motion Analysis Using A Kinematic Sensor: Monitoring of Daily Physical Activity in the Elderly", *Biomedical Engineering, IEEE Transactions*, Vol. 50, pp. 711-723, 2003.

54. Godfrey, A., K. M. Culhane and G. M. Lyons, "Comparison of the Performance of the activPAL™ Professional Physical Activity Logger To A Discrete Accelerometer-Based Activity Monitor", *Medical Engineering and Physics*, Vol. 29, pp. 930-934, 2007.
55. Vihriala, E., R. Saarimaa, R. Myllyla and T. Jamsa, "A Device for Long Term Monitoring of Impact Loading on the Hip"., *Molecular and Quantum Acoustics*, Vol. 24, pp. 211-224, 2003.
56. de Vries, S. I., I. Bakker, M. Hopman-Rock, R. A. Hirasing and W. Mechelen, "Clinimetric Review of Motion Sensors in Children and Adolescents", *Journal of Clinical Epidemiology*, Vol. 59, pp. 670-680, 2006.
57. Foubert, N., A. McKee, R. Goubran, and F. Knoefel, "Lying and Sitting Posture Recognition and Transition Detection Using A Pressure Sensor Array", *Medical Measurements and Applications Proceedings (MeMeA), 2012 IEEE International Symposium*, Budapest, 18 May-19 May 2012, pp. 1-6, 2012.
58. Qian, H., Y. Mao, W. Xiang, and Z. Wang, "Recognition of Human Activities Using SVM Multi-class Classifier", *Pattern Recognition Letters*, Vol. 31, pp. 100-111, 2010.
59. Kim, Y. and H. Ling, "Human Activity Classification Based on Microdoppler Signatures Using A Support Vector Machine", *Geoscience and Remote Sensing, IEEE Transactions*, Vol. 47, pp. 1328-1337, 2009.
60. Hsu, C., and C. Lin, "A Comparison of Methods for Multiclass Support Vector Machines", *Neural Networks, IEEE Transactions*, Vol. 13, pp. 414-425, 2002.
61. Krishnamurthy, V., K. Fowler, and E. Sazonov, "The Effect of Time Synchronization of Wireless Sensors on the Modal Analysis of Structures", *Smart Materials and Structures*, Vol. 17, 13 pages, 2008.
62. Aiello, F., F. L. Bellifemine, G. Fortino, S. Galzarano and R. Gravina, "An Agent-Based Signal Processing In-node Environment for Real-Time Human Activity

Monitoring Based on Wireless Body Sensor Networks”, *Engineering Applications of Artificial Intelligence*, Vol. 24, pp. 1147-1161, 2011.

63. Okeyo, G., L. Chen, H. Wang and R. Sterritt, “Dynamic Sensor Data Segmentation for Real-Time Knowledge-Driven Activity Recognition”, *Pervasive and Mobile Computing*, Available online 3 December 2012, ISSN 1574-1192, <http://dx.doi.org/10.1016/j.pmcj.2012.11.004>.

64. Muscillo, R., M. Schmid, S. Conforto, and T. DAlessio, “An Adaptive Kalman-Based Bayes Estimation Technique To Classify Locomotor Activities in Young and Elderly Adults Through Accelerometers,” *Medical Engineering & Physics*, Vol. 32, pp. 849 - 859, 2010.

65. Ganea, R, A. Paraschiv-Ionescu, and K. Aminian, “Detection and Classification of Postural Transitions in Real-World Conditions,” *Neural Systems and Rehabilitation Engineering, IEEE Transactions*, Vol. 20, pp. 688 - 696, 2012.

66. Krishnan, N. C. and D. J. Cook, “Activity Recognition on Streaming Sensor Data”, *Pervasive and Mobile Computing*, available online 2012, in press.

67. Junker, H., O. Amft, P. Lukowicz and G. Tröster, “Gesture Spotting with Body-Worn Inertial Sensors to Detect User Activities”, *Pattern Recognition*, Vol. 41, pp. 2010 - 2024, 2007.

68. Krishnan, N. C. and S. Panchanathan, “Analysis of Low Resolution Accelerometer Data for Continuous Human Activity Recognition”, *Acoustics, Speech and Signal Processing, 2008, ICASSP 2008, IEEE International Conference*, Las Vegas, NV, 31 March-4 April 2008, pp. 3337-3340, 2008.

69. Wang, L., T. Gu, X. Tao and J. Lu, “A Hierarchical Approach To Real-Time Activity Recognition in Body Sensor Networks”, *Pervasive and Mobile Computing*, Vol. 8, pp. 115-130, 2012.

70. Boyd, J. and H. Sundaram, "A Framework to Detect and Classify Activity Transitions in Low-Power Applications", *Proceedings of the 2009 IEEE International Conference on Multimedia and Expo, ser. ICME'09*, New York, NY, USA, 2009, pp. 1712–1715, IEEE Press, Piscataway, NJ, USA, 2009.
71. Jarchi, D., L. Atallah, and G. Z. Yang, "Transition Detection and Activity Classification from Wearable Sensors Using Singular Spectrum Analysis", *Wearable and Implantable Body Sensor Networks (BSN), 2012 Ninth International Conference*, London, 9 May-12 May 2012, pp. 136-141, 2012.
72. Vapnik, V. N., *The Nature of Statistical Learning Theory*, Springer-Verlag New York, Inc., New York, 1995.
73. Vapnik, V. N., *Statistical Learning Theory*, John Wiley and Sons Inc., New York, 1998.
74. Rennie, J. D. M. and R. Rifkin, Improving Multiclass Text Classification with the Support Vector Machine, AI Memo 2001-026, CBCL Memo 210, 2001.
75. Nakajima, C., N. Itoh, M. Pontil and T. Poggio, "Object Recognition and Detection by A Combination of Support Vector Machine and Rotation Invariant Phase Only Correlation", *Pattern Recognition, 2000, Proceedings, 15th International Conference*, Barcelona, 3 September-7 September 2000, Vol. 4, pp. 787-790, 2000.
76. Yang, M. H. and A. Cornuejols, "Introduction To Support Vector Machines", Dr. Gabriela (Serban) Czibula's homepage, http://www.cs.ubbcluj.ro/~gabis/ml/Lectures/6_SVM.pdf [retrieved 24 April 2012].
77. Blanz, V., B. Schölkopf, H. Bülthoff, C. Burges, V. Vapnik, and T. Vetter, "Comparison of View-Based Object Recognition Algorithms Using Realistic 3D Models", in C. Malsburg, W. Seelen, Jan C. Vorbrüggen and B. Sendhoff (eds.), *Artificial Neural Networks - ICANN 96*, pp. 251-256, Springer Berlin Heidelberg, 1996.

78. Schölkopf, B., C. Burges, and V. Vapnik. "Extracting Support Data for A Given Task", *Proceedings of First International Conference on Knowledge Discovery & Data Mining*, Menlo Park, 1995, pp. 252-257, AAAI Press, 1995.
79. Weston, J. and C. Watkins, *Multi-class Support Vector Machines*, Egham, 1998.
80. Ventura, D., "SVM Example",
<http://axon.cs.byu.edu/Dan/678/miscellaneous/SVM.example.pdf>, [retrieved 15 May 2012].
81. Texas Instruments, "eZ430-Chronos™ Development Tool User's Guide",
<http://www.ti.com/lit/ug/slau292d/slau292d.pdf> [retrieved 1 October 2011].

APPENDIX A: NUMERICAL EXAMPLE FOR RT-CAM_{KD}

$A=\{a_1, a_2, a_3, a_4, a_5\}$ where a_1, a_2, a_3, a_4 and a_5 correspond to *drinking*, *eating*, *pouring*, *toothBrushing* and *turningKey* respectively. Let's assume that $Ta=\{M_1^{2 \times 3}\} \forall a \in A$ and $M_1^{2 \times 3} = \begin{bmatrix} -46 & 3 & 8 \\ -46 & 3 & 8 \end{bmatrix}$ for $a=a_1$. (We are demonstrating the operations only on a_1 since operations on other simple actions are handled in a similar way.) To generate the pattern for a_1 , we start from the *training* algorithm: $P_{a_1}^{1 \times 5} \leftarrow \text{featureExtraction}(M_1^{2 \times 3}|_{a=a_1})$

Then execution switches to *featureExtraction* module: $C^{2 \times 3} = \begin{bmatrix} -46 & 3 & 8 \\ -46 & 3 & 8 \end{bmatrix}$ and $m=2$ and $n=3$, $R^{2 \times 3} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$, $d(1,:) = K((-46 \ 3 \ 8))$, $d(2,:) = K((-46 \ 3 \ 8))$, $d(3,:) = K((1 \ 1 \ 1))$, $d(4,:) = K((1 \ 1 \ 1))$. Taking $e=2.72$ and using the K function: $d_{14} = \frac{1}{1+2.72^{-46}} + \frac{1}{1+2.72^3} + \frac{1}{1+2.72^8} = 1.047$, $d_{i4} \forall 2 \leq i \leq 4$ are calculated as d_{14} , yielding

$$D^{4 \times 4} = \begin{bmatrix} -46.000 & 3.000 & 8.000 & 1.047 \\ -46.000 & 3.000 & 8.000 & 1.047 \\ 1.000 & 1.000 & 1.000 & 0.806 \\ 1.000 & 1.000 & 1.000 & 0.806 \end{bmatrix} \text{ where } 2m=4 \text{ and } n+1=4.$$

Execution switching to *coreTraining* algorithm with the input being $D^{4 \times 4}$:

$$F^{4 \times 4} = \begin{bmatrix} 2191.804 & 2191.804 & -32.698 & -32.698 \\ 2191.804 & 2191.804 & -32.698 & -32.698 \\ -32.698 & -32.698 & 4.892 & 4.892 \\ -32.698 & -32.698 & 4.892 & 4.892 \end{bmatrix}, b^4 = (1 \ 1 \ -1 \ -1), x^4 = F^{4 \times 4} b^4 = (4449.004$$

$4449.004 \ -75.180 \ -75.180)$, $w^5 = (-409459.000 \ 26543.660 \ 71033.700 \ 9195.024 \ 11462.050)$

which is the pattern for a_1 . After w^5 is computed, $P_{a_1}^{1 \times 5}$ is set to w^5 .

The *prediction* algorithm is demonstrated as the following: Since *featureExtraction* procedure is numerically explained above, how f^{n+1} is generated from $C^{m \times n}$ can be skipped. We assume to have 1 training sample for each simple action as previously mentioned, therefore u becomes one in $g(a_i)$ calculation, yielding:

$$\begin{aligned}
g(a_1) &= \text{error}(p_{a_1}(1, :), f^{n+1}) = \text{error}((-409459.000 \quad 26543.660 \quad 71033.700 \quad 9195.024 \\
& 11462.050), f^{n+1}) \\
&= \frac{\left| \frac{f_1^{n+1} + 409459}{-409459} \right| + \left| \frac{f_2^{n+1} - 26543.66}{26543.66} \right| + \left| \frac{f_3^{n+1} - 71033.7}{71033.7} \right| + \left| \frac{f_4^{n+1} - 9195.024}{9195.024} \right| + \left| \frac{f_5^{n+1} - 11462.05}{11462.05} \right|}{5}.
\end{aligned}$$

Similarly, $g(a_2)$, $g(a_3)$, $g(a_4)$ and $g(a_5)$ are calculated, finally the simple action yielding the minimum of these values becomes the detected action.



APPENDIX B: DETECTED ACTIVITIES IN T₁P₁ TESTS

Table B.1 to Table B.20 illustrate detected activities in T₁P₁ tests. For a more concise presentation, *toothBrushing* and *turningKey* actions are abbreviated as tB and tK respectively within the tables. testId showing the number of the test, action names given in the column names indicate the actual type of action whereas action names within the table entries show detected activity result corresponding to the actual activity whose type is specified by the related column name. Therefore, when a table entry matches the related column name, it means a successful detection. The composite action tests incorporate several columns named as chunk_2, chunk_4 and chunk_6, which represents first, second and third transitions respectively. Since transition type is inferred considering the simple actions right before and after the transition, the types detected for them are ignored.

Table B.1. Drinking

testId	drinking
1	drinking
2	drinking
3	drinking
4	drinking
5	drinking
6	drinking
7	drinking
8	drinking
9	drinking
10	drinking

Table B.2. Drinking_pouring

testId	drinking	chunk_2	pouring
1	drinking	pouring	pouring
2	drinking	pouring	pouring
3	drinking	pouring	pouring
4	drinking	pouring	tB
5	drinking	pouring	tB
6	drinking	pouring	pouring
7	drinking	pouring	pouring
8	drinking	pouring	tB
9	drinking	pouring	pouring
10	drinking	pouring	tB

Table B.3. Drinking_toothBrushing

testId	drinking	chunk_2	tB
1	drinking	drinking	tB
2	drinking	drinking	tB
3	drinking	drinking	tB
4	drinking	drinking	tB
5	drinking	drinking	tB
6	drinking	drinking	tB
7	drinking	drinking	tB
8	drinking	drinking	tB
9	drinking	drinking	tB
10	drinking	drinking	tB

Table B.4. Drinking_toothBrushing_pouring

testId	drinking	chunk_2	tB	chunk_4	pouring
1	drinking	drinking	tB	drinking	pouring
2	drinking	drinking	tB	pouring	pouring
3	drinking	drinking	tB	pouring	tB
4	drinking	drinking	tB	pouring	pouring
5	drinking	drinking	tB	pouring	pouring
6	drinking	drinking	tB	pouring	pouring
7	drinking	drinking	tB	pouring	pouring
8	drinking	drinking	tB	pouring	pouring
9	drinking	drinking	tB	pouring	tB
10	drinking	drinking	tB	drinking	pouring

Table B.5. Drinking_toothBrushing_pouring_turningKey

testId	drinking	chunk_2	tB	chunk_4	pouring	chunk_6	tK
1	drinking	drinking	tB	pouring	tB	tK	tK
2	drinking	drinking	tB	drinking	tB	tK	tK
3	drinking	drinking	tB	pouring	pouring	tK	tK
4	drinking	drinking	tB	pouring	tB	tK	tK
5	drinking	drinking	tB	pouring	tB	tK	tK
6	drinking	drinking	tB	drinking	pouring	tK	tK
7	drinking	drinking	tB	pouring	tB	tK	tK
8	drinking	drinking	tB	pouring	pouring	tK	tB
9	drinking	drinking	tB	pouring	tB	tK	tK
10	drinking	drinking	tB	pouring	tB	tK	tK

Table B.6. Drinking_toothBrushing_turningKey

testId	drinking	chunk_2	tB	chunk_4	tK
1	drinking	drinking	tB	tK	tK
2	drinking	drinking	tB	tK	tK
3	drinking	drinking	tB	tK	tK
4	drinking	drinking	tB	tK	tK
5	drinking	drinking	tB	pouring	pouring
6	drinking	drinking	tB	tK	tB
7	drinking	drinking	tB	tK	tK
8	drinking	drinking	tB	tK	tK
9	drinking	drinking	tB	pouring	tK
10	drinking	drinking	tB	tK	tB

Table B.7. Drinking_toothBrushing_turningKey_pouring

testId	drinking	chunk_2	tB	chunk_4	tK	chunk_6	pouring
1	drinking	drinking	tB	tK	tB	pouring	pouring
2	drinking	drinking	tB	tK	tK	pouring	tB
3	drinking	drinking	tB	tK	tK	pouring	pouring
4	drinking	drinking	tB	tK	tK	pouring	tB
5	drinking	drinking	tB	tK	tK	pouring	tB
6	drinking	drinking	tB	tK	tB	pouring	pouring
7	drinking	drinking	tB	tK	tB	pouring	pouring
8	drinking	drinking	tB	pouring	tK	pouring	tB
9	drinking	drinking	tB	tK	tK	pouring	pouring
10	drinking	drinking	tB	tK	tK	pouring	tB

Table B.8. Drinking_turningKey

testId	drinking	chunk_2	tK
1	drinking	pouring	pouring
2	drinking	pouring	tK
3	drinking	pouring	tK
4	drinking	pouring	tK
5	drinking	pouring	pouring
6	drinking	pouring	pouring
7	drinking	pouring	tK
8	drinking	pouring	pouring
9	drinking	pouring	tK
10	drinking	pouring	tK

Table B.9. Pouring

testId	pouring
1	pouring
2	pouring
3	tB
4	tB
5	tB
6	tB
7	pouring
8	tB
9	tB
10	tB

Table B.10. Pouring_turningKey

testId	pouring	chunk_2	tK
1	pouring	pouring	tK
2	pouring	pouring	tK
3	pouring	pouring	tK
4	tB	pouring	tK
5	tB	pouring	tK
6	pouring	pouring	tK
7	pouring	pouring	tK
8	pouring	pouring	tK
9	pouring	pouring	tK
10	pouring	pouring	tK

Table B.11. ToothBrushing_drinking

testId	tB	chunk_2	drinking
1	tB	tK	drinking
2	tB	tK	drinking
3	tB	tK	drinking
4	tB	tK	drinking
5	tB	tK	drinking
6	tB	tK	drinking
7	tB	tK	drinking
8	tB	tK	drinking
9	tB	tK	drinking
10	tB	tK	drinking

Table B.12. ToothBrushing_drinking_pouring

testId	tB	chunk_2	drinking	chunk_4	pouring
1	tB	tK	drinking	pouring	pouring
2	tB	tK	drinking	pouring	tB
3	tB	tK	drinking	pouring	tB
4	tB	tK	drinking	pouring	tB
5	tB	tK	drinking	pouring	tB
6	tB	tK	drinking	pouring	tB
7	tB	tK	drinking	pouring	pouring
8	tB	tK	drinking	pouring	tB
9	tB	tK	drinking	pouring	pouring
10	tB	tK	drinking	pouring	tB

Table B.13. ToothBrushing_drinking_pouring_turningKey

testId	tB	chunk_2	drinking	chunk_4	pouring	chunk_6	tK
1	tB	tK	drinking	pouring	pouring	tK	tB
2	tB	tK	drinking	pouring	pouring	tK	tK
3	tB	tK	drinking	pouring	pouring	tK	tK
4	tB	tK	drinking	pouring	pouring	tK	tK
5	tB	tK	drinking	pouring	pouring	tK	tB
6	tB	tK	drinking	pouring	tB	tK	tK
7	tB	tK	drinking	pouring	pouring	tK	tK
8	tB	tK	drinking	pouring	pouring	tK	tK
9	tB	tK	drinking	pouring	pouring	tK	tK
10	tB	tK	drinking	pouring	pouring	tK	tK

Table B.14. ToothBrushing_drinking_turningKey

testId	tB	chunk_2	drinking	chunk_4	tK
1	tB	tK	drinking	pouring	tB
2	tB	tK	drinking	pouring	tK
3	tB	tK	drinking	pouring	tB
4	tB	tK	drinking	pouring	tK
5	tB	tK	drinking	pouring	tB
6	tB	tK	drinking	tK	tK
7	tB	tK	drinking	pouring	tK
8	tB	tK	drinking	tK	tK
9	tB	tK	drinking	pouring	tK
10	tB	tK	drinking	tK	tK

Table B.15. ToothBrushing_drinking_turningKey_pouring

testId	tB	chunk_2	drinking	chunk_4	tK	chunk_6	pouring
1	tB	tK	drinking	pouring	pouring	pouring	pouring
2	tB	tK	drinking	pouring	tK	pouring	pouring
3	tB	tK	drinking	pouring	tK	pouring	tB
4	tB	tK	drinking	pouring	tK	pouring	pouring
5	tB	tK	drinking	pouring	pouring	pouring	pouring
6	tB	tK	drinking	pouring	pouring	pouring	pouring
7	tB	tK	drinking	pouring	tK	pouring	pouring
8	tB	tK	drinking	pouring	pouring	pouring	pouring
9	tB	tK	drinking	pouring	tB	pouring	pouring
10	tB	tK	drinking	pouring	pouring	tK	pouring

Table B.16. ToothBrushing

testId	tB
1	tB
2	tB
3	tB
4	tB
5	tB
6	tB
7	tB
8	tB
9	tB
10	tB

Table B.17. ToothBrushing_pouring

testId	tB	chunk_2	pouring
1	tB	pouring	pouring
2	tB	pouring	pouring
3	tB	pouring	pouring
4	tB	drinking	pouring
5	tB	pouring	pouring
6	tB	pouring	pouring
7	tB	pouring	pouring
8	tB	pouring	pouring
9	tB	pouring	pouring
10	tB	pouring	pouring

Table B.18. ToothBrushing_turningKey

testId	tB	chunk_2	tK
1	tB	tK	tK
2	tB	tK	tK
3	tB	tK	tK
4	tB	tK	tK
5	tB	tK	tK
6	tB	tK	tK
7	tB	pouring	pouring
8	tB	tK	tK
9	tB	tK	tK
10	tB	tK	tK

Table B.19. TurningKey

testId	tK
1	tB
2	tB
3	tK
4	tB
5	tK
6	tB
7	tB
8	tK
9	tB
10	tB

Table B.20. TurningKey_pouring

testId	tK	chunk_2	pouring
1	tK	pouring	pouring
2	tK	pouring	tB
3	tK	pouring	tB
4	tK	pouring	pouring
5	tK	pouring	tB
6	tK	pouring	tB
7	tK	pouring	tB
8	tK	pouring	pouring
9	tK	pouring	pouring
10	tK	pouring	tB

APPENDIX C: REAL TIME OVERHEAD IN T₁P₁ TESTS

Table C.1 to Table C.20 show the real time overhead introduced by RT-CAM in T₁P₁ tests. testId showing the number of the test, action name given in the column name indicates the actual type of the action whereas each table entry shows the real time overhead for determining the detected action specified by the related column and test number.

Table C.1. Drinking

testId	drinking
1	0,05310
2	0,05487
3	0,05584
4	0,05663
5	0,05745
6	0,05856
7	0,05664
8	0,05607
9	0,05586
10	0,05762

Table C.2. Drinking_pouring

testId	drinking	chunk_2	pouring
1	0,05495	0,08276	0,08013
2	0,05510	0,08431	0,08223
3	0,05490	0,08153	0,07642
4	0,05458	0,08232	0,07628
5	0,05579	0,08498	0,08002
6	0,05403	0,08135	0,07863
7	0,05771	0,08631	0,08093
8	0,05758	0,08551	0,08174
9	0,05739	0,08064	0,08332
10	0,05401	0,08295	0,08035

Table C.3. Drinking_toothBrushing

testId	drinking	chunk_2	tB
1	0,05393	0,08205	0,07946
2	0,05300	0,08142	0,07672
3	0,05290	0,08311	0,08107
4	0,05786	0,08222	0,08214
5	0,05279	0,08228	0,08348
6	0,05521	0,08515	0,08344
7	0,05401	0,08115	0,07838
8	0,05738	0,08243	0,08404
9	0,05423	0,08104	0,07997
10	0,05894	0,08473	0,07917

Table C.4. Drinking_toothBrushing_pouring

testId	drinking	chunk_2	tB	chunk_4	pouring
1	0,05151	0,08281	0,07960	0,10584	0,07840
2	0,05548	0,08357	0,07975	0,07875	0,07884
3	0,05613	0,08344	0,08201	0,08195	0,08206
4	0,05679	0,08459	0,08590	0,08288	0,08225
5	0,05309	0,08172	0,08010	0,07966	0,07953
6	0,05639	0,08095	0,07963	0,08194	0,07941
7	0,05808	0,08403	0,08186	0,08391	0,08198
8	0,05235	0,07781	0,08012	0,07900	0,07906
9	0,05499	0,08479	0,08104	0,08288	0,08220
10	0,05620	0,08463	0,07963	0,08278	0,08074

Table C.5. Drinking_toothBrushing_pouring_turningKey

testId	drinking	chunk_2	tB	chunk_4	pouring	chunk_6	tK
1	0,08476	0,08151	0,08152	0,08203	0,08042	0,08402	0,08111
2	0,05732	0,08506	0,08175	0,10901	0,08212	0,07850	0,07914
3	0,05409	0,08179	0,08184	0,11050	0,08137	0,08075	0,08610
4	0,05567	0,08430	0,08142	0,10747	0,08041	0,07610	0,07817
5	0,05542	0,08291	0,07816	0,10885	0,07802	0,07868	0,08230
6	0,05659	0,08530	0,08269	0,10722	0,08148	0,08230	0,08308
7	0,05802	0,08514	0,08239	0,10645	0,08443	0,07841	0,08017
8	0,05350	0,08094	0,10976	0,08249	0,08181	0,08216	0,08100
9	0,05806	0,08438	0,08217	0,10875	0,07910	0,07996	0,08087
10	0,05686	0,08078	0,08355	0,11045	0,08089	0,08282	0,08286

Table C.6. Drinking_toothBrushing_turningKey

testId	drinking	chunk_2	tB	chunk_4	tK
1	0,05523	0,08515	0,07961	0,08106	0,07961
2	0,05043	0,08063	0,08072	0,07929	0,07645
3	0,05049	0,08246	0,07848	0,08210	0,08185
4	0,05631	0,08118	0,08216	0,08167	0,07942
5	0,05877	0,08696	0,08210	0,08500	0,07981
6	0,05774	0,08467	0,08279	0,08303	0,07996
7	0,05451	0,08215	0,08206	0,08421	0,08327
8	0,05391	0,08175	0,07836	0,07865	0,07773
9	0,05877	0,08458	0,07796	0,07928	0,08027
10	0,05415	0,08367	0,08108	0,08214	0,08311

Table C.7. Drinking_toothBrushing_turningKey_pouring

testId	drinking	chunk_2	tB	chunk_4	tK	chunk_6	pouring
1	0,05700	0,08145	0,08343	0,08504	0,10927	0,08019	0,07894
2	0,05566	0,08392	0,10766	0,08202	0,08175	0,08032	0,07841
3	0,05536	0,07937	0,08223	0,10931	0,08301	0,08144	0,07932
4	0,05151	0,08026	0,07829	0,10960	0,07886	0,08113	0,07880
5	0,05448	0,08170	0,08071	0,10775	0,08341	0,07793	0,07488
6	0,05500	0,08179	0,07838	0,10996	0,07854	0,07897	0,07643
7	0,05568	0,08093	0,10584	0,08496	0,07882	0,07820	0,07484
8	0,05303	0,08344	0,07937	0,10808	0,08127	0,07940	0,07863
9	0,05741	0,08255	0,08216	0,10852	0,08329	0,08285	0,07998
10	0,05460	0,08224	0,08152	0,10957	0,08141	0,08228	0,08202

Table C.8. Drinking_turningKey

testId	drinking	chunk_2	tK
1	0,05546	0,08482	0,08308
2	0,05681	0,08567	0,08112
3	0,05753	0,08825	0,08320
4	0,05299	0,08187	0,08123
5	0,05641	0,08656	0,08230
6	0,05794	0,08388	0,08037
7	0,05505	0,08302	0,08010
8	0,05591	0,08725	0,08453
9	0,05260	0,08141	0,08049
10	0,05298	0,08341	0,07939

Table C.9. Pouring

testId	pouring
1	0,05385
2	0,05538
3	0,05566
4	0,05556
5	0,05739
6	0,05330
7	0,05658
8	0,05270
9	0,05018
10	0,05621

Table C.10. Pouring_turningKey

testId	pouring	chunk_2	tK
1	0,05238	0,08585	0,08337
2	0,05551	0,08571	0,08239
3	0,04949	0,08220	0,07938
4	0,05145	0,08373	0,08068
5	0,04927	0,07967	0,08034
6	0,05808	0,08109	0,08271
7	0,05671	0,08482	0,08227
8	0,05746	0,08384	0,08340
9	0,05503	0,08167	0,08015
10	0,05416	0,08543	0,08202

Table C.11. ToothBrushing_drinking

testId	tB	chunk_2	drinking
1	0,05451	0,08775	0,07813
2	0,05468	0,08207	0,07666
3	0,05583	0,08578	0,08014
4	0,05371	0,08152	0,07597
5	0,05095	0,08290	0,07879
6	0,05528	0,08506	0,07868
7	0,05489	0,08653	0,08182
8	0,05445	0,08501	0,08109
9	0,05506	0,08318	0,08151
10	0,05591	0,08651	0,08010

Table C.12. ToothBrushing_drinking_pouring

testId	tB	chunk_2	drinking	chunk_4	pouring
1	0,05408	0,08226	0,08092	0,07967	0,07964
2	0,05543	0,08290	0,08241	0,08054	0,08241
3	0,05242	0,08239	0,07679	0,08223	0,07656
4	0,05686	0,08072	0,08079	0,08118	0,07814
5	0,05450	0,08271	0,07893	0,08330	0,07805
6	0,05802	0,08501	0,08023	0,08127	0,08008
7	0,05521	0,08317	0,07923	0,08046	0,07914
8	0,05250	0,08187	0,07812	0,08135	0,07957
9	0,05427	0,08097	0,07890	0,08030	0,07766
10	0,05476	0,08299	0,08191	0,08219	0,08395

Table C.13. ToothBrushing_drinking_pouring_turningKey

testId	tB	chunk_2	drinking	chunk_4	pouring	chunk_6	tK
1	0,05666	0,08396	0,08063	0,08179	0,08156	0,11058	0,08019
2	0,05797	0,08378	0,10757	0,07929	0,08040	0,08154	0,07938
3	0,05615	0,08088	0,08267	0,11057	0,08034	0,08266	0,08123
4	0,05822	0,08302	0,10817	0,07972	0,07902	0,08241	0,08377
5	0,05500	0,08407	0,08155	0,10845	0,07933	0,08146	0,07870
6	0,05379	0,08324	0,10911	0,08086	0,08040	0,08292	0,08023
7	0,05571	0,08027	0,08086	0,10630	0,07965	0,07727	0,07659
8	0,05575	0,08319	0,07592	0,10894	0,07822	0,07937	0,07663
9	0,05619	0,08306	0,10601	0,07929	0,07734	0,07834	0,07945
10	0,05637	0,08134	0,10860	0,08443	0,08036	0,07613	0,07927

Table C.14. ToothBrushing_drinking_turningKey

testId	tB	chunk_2	drinking	chunk_4	tK
1	0,04964	0,08735	0,07904	0,08296	0,08425
2	0,05607	0,08294	0,08025	0,08123	0,08301
3	0,05596	0,08653	0,08242	0,08370	0,08321
4	0,05627	0,08100	0,07964	0,08516	0,07926
5	0,05715	0,08219	0,08191	0,08207	0,08313
6	0,05254	0,08399	0,08291	0,08512	0,08233
7	0,05779	0,08558	0,08094	0,08202	0,08613
8	0,05074	0,08557	0,08180	0,08309	0,08289
9	0,05640	0,08415	0,08192	0,08230	0,08384
10	0,05922	0,08592	0,08119	0,08133	0,08571

Table C.15. ToothBrushing_drinking_turningKey_pouring

testId	tB	chunk_2	drinking	chunk_4	tK	chunk_6	pouring
1	0,05446	0,08193	0,07809	0,07980	0,10863	0,07668	0,07677
2	0,05493	0,08436	0,08093	0,10605	0,08249	0,07809	0,07911
3	0,05696	0,08267	0,07993	0,10894	0,08274	0,08004	0,08215
4	0,05429	0,08355	0,08075	0,10682	0,08233	0,07831	0,07584
5	0,05362	0,08262	0,07930	0,11352	0,08033	0,08050	0,07632
6	0,05531	0,08278	0,10563	0,08062	0,08009	0,07898	0,07801
7	0,05546	0,08236	0,10458	0,08082	0,08018	0,08051	0,07872
8	0,05304	0,08516	0,07909	0,10774	0,07822	0,07875	0,07559
9	0,05259	0,08093	0,07953	0,11246	0,07877	0,08255	0,08089
10	0,05447	0,08250	0,08047	0,10808	0,07884	0,07808	0,07558

Table C.16. ToothBrushing

testId	tB
1	0,05685
2	0,05345
3	0,05206
4	0,05832
5	0,05573
6	0,05247
7	0,05306
8	0,05427
9	0,05720
10	0,05832

Table C.17. ToothBrushing_pouring

testId	tB	chunk_2	pouring
1	0,05312	0,08484	0,07802
2	0,05541	0,08239	0,07863
3	0,05395	0,08162	0,07612
4	0,05720	0,08577	0,08271
5	0,05769	0,08317	0,08172
6	0,05644	0,08157	0,07880
7	0,05564	0,08392	0,08230
8	0,05752	0,08257	0,08233
9	0,05552	0,08377	0,08087
10	0,05803	0,08405	0,08076

Table C.18. ToothBrushing_turningKey

testId	tB	chunk_2	tK
1	0,05868	0,08413	0,08257
2	0,05185	0,08465	0,08252
3	0,05474	0,08288	0,08297
4	0,05864	0,08558	0,08151
5	0,05478	0,08561	0,07925
6	0,05464	0,07851	0,07984
7	0,05666	0,08466	0,08370
8	0,05526	0,08243	0,08061
9	0,05023	0,08564	0,07989
10	0,05481	0,08077	0,07917

Table C.19. TurningKey

testId	tK
1	0,05601
2	0,05339
3	0,05611
4	0,05443
5	0,05871
6	0,05565
7	0,05694
8	0,05820
9	0,05491
10	0,05627

Table C.20. TurningKey_pouring

testId	tK	chunk_2	pouring
1	0,05159	0,08237	0,07853
2	0,05341	0,08304	0,08042
3	0,05963	0,08051	0,08283
4	0,05629	0,08236	0,08208
5	0,05870	0,08420	0,08216
6	0,05567	0,08095	0,07584
7	0,05605	0,08301	0,08370
8	0,05646	0,07856	0,07964
9	0,05576	0,08651	0,08142
10	0,05791	0,08081	0,07829