SPATIOTEMPORAL ANALYSIS OF HUMAN ACTIONS USING RGB-D CAMERAS

By
Farhood Negin

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
Spring 2013

Spatiotemporal Analysis of Human Actions Using RGB-D Cameras

APPROVED BY

Prof. Dr. Aytül Erçil ....
(Thesis Supervisor)

Dr. Ceyhun Burak Akgül …………...............................
(Thesis Co-Advisor)

Assoc. Prof. Dr. Hakan Erdoğan ....

Assoc. Prof. Dr. Gözde Ünal ....

Assoc. Prof. Dr. Yücel Saygın ....

DATE OF APPROVAL: ....

to my family

Acknowledgments

I wish to express my gratitude to my supervisor, Aytül Erçil, whose expertise, understanding, and patience, added considerably to my graduate experience. I am grateful to her not only for the completion of this thesis, but also for her unconditional support from the beginning.

I owe a special gratitude to my co-advisor Dr. Ceyhun Burak Akgül for his professional support and practical guidance. His encouragement has been of immense significance in completing this work.

I also would like to thank TÜBİTAK for providing the necessary fınancial support for my graduate education[1].

I am grateful to my committee members Hakan Erdoğan, Gözde Ünal and Yücel Saygın for taking the time to read and comment on my thesis.

I owe special thanks to all my friends and colleagues, particularly to, Fırat Özdemir, Shirin Mirlohi, Ali Atabey, Solmaz Çelik for their care, support, friendship and assistance during this period.

Finally, I would like to thank my family Monireh, Masoud, Hamid and Ali for their valuable support, love and encouragement.

[1] This work has been funded in part by TÜBİTAK Project 109E134 Vipsafe.

Spatiotemporal Analysis of Human Actions Using RGB-D Cameras

Farhood Negin.

CS, M.Sc. Thesis, 2013

Thesis Supervisor: Aytül Erçil

Keywords: human motion analysis, action recognition, random decision forest

Abstract

Markerless human motion analysis has strong potential to provide cost-efficient solution for action recognition and body pose estimation. Many applications including human-computer interaction, video surveillance, content-based video indexing, and automatic annotation among others will benefit from a robust solution to these problems. Depth sensing technologies in recent years have positively changed the climate of the automated vision-based human action recognition problem, deemed to be very difficult due to the various ambiguities inherent to conventional video.

In this work, first a large set of invariant spatiotemporal features is extracted from skeleton joints (retrieved from depth sensor) in motion and evaluated as baseline performance. Next we introduce a discriminative Random Decision Forest-based feature selection framework capable of reaching impressive action recognition performance when combined with a linear SVM classifier. This approach improves upon the baseline performance obtained using the whole feature set with a significantly less number of features (one tenth of the original). The approach can also be used to provide insights on the spatiotemporal dynamics of human actions.

A novel therapeutic action recognition dataset (WorkoutSU-10) is presented. We took advantage of this dataset as a benchmark in our tests to evaluate the reliability of our proposed methods. Recently the dataset has been published publically as a contribution to the action recognition community.

In addition, an interactive action evaluation application is developed by utilizing the proposed methods to help with real life problems such as ‘fall detection’ in the elderly people or automated therapy program for patients with motor disabilities.

Derinlik Kameralarindan Elde Edilen Insan Edimlerinin Uzam-Zamansal Analizi

Farhood Negin
CS, M.Sc. Tez, 2013
Tez Süpervizörü: Aytül Erçil

Anahtar Kelimeler: İnsan hareket analizi, hareket tanıma, rastgele karar ormanı

Özet

İşaretleyicisiz insan hareket analizinin, hareket tanıma ve vücut poz tahmini için verimli maliyete sahip çözüm sunma potansiyeli vardır. İnsan-bilgisayar etkileşimi, video gözetlemesi, içerik tabanlı video indeksleme, ve otomatik açıklama da dahil olmak üzere birçok uygulama, bu güçlü çözümden yararlanacaktır. Geleneksel video doğasındaki çeşitli belirsizlikler sebebiyle çok zor olarak kabul edilen otomatik görüntü tabanlı insan hareketi tanıma sorunu, son yıllardaki derinlik algılama teknolojileri sayesinde olumlu değişiklikler gösterdi.

Bu çalışmada, ilk olarak değişmeyen spatiotemporal özellikli büyük bir set, hareket halindeki iskelet eklemlerinden (derinlik sensöründen alınan) elde edilir ve temel performans olarak değerlendirilmektedir. Sonra, bir lineer SVM sınıflandırıcı ile kombine edildiğinde etkileyici hareket tanıma performansına ulaşma kapasitesine sahip bir ayrımcı Rastgele Karar Ormanı tabanlı özellik seçimi çatısı tanıttık. Bu yaklaşım özellik kümesi önemli ölçüde daha az sayıda (orijinalin onda biri) kullanarak tüm özellik kümesini kullanarak elde edilen temel performansa üstünlük sağlar. Bu yaklaşım aynı zamanda insan hareketlerinin mekan-zamansal dinamikleri üzerinde fikir edinmek için kullanılabilir.

Yeni bir tedavi edici hareket tanıma veri kümesi (WorkoutSU-10) sunulmuştur. Önerilen yöntemlerimizin güvenilirliğini değerlendirmek için testlerimizde bir kriter

olarak bu veri kümesinden yararlandık. Yakın geçmişte, veri kümesi hareket tanıma toplumuna bir katkı olmak üzere kamuya yayınlanmıştır.

Buna ek olarak, yaşlı insanlardaki 'düşüş algılama' gibi gerçek hayat problemlerine veya motor engelli hastalar için otomatik terapi programına yardımcı olmak için sunulan yöntemler kullanılarak interaktif bir hareket değerlendirme uygulaması geliştirilmiştir.

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1

Introduction

"Neither the human condition in particular nor our explanatory knowledge in general will ever be perfect, nor even approximately perfect. We shall always be at the beginning of infinity."

-David Deutsch

1.1 Motivation

Markerless human motion analysis has strong potential to provide cost-efficient solution for action recognition and body pose estimation. Many applications including human-computer interaction, video surveillance, content-based video indexing, and automatic annotation among others will benefit from a robust solution for these problems. This raises a high motivation for significant research effort on this domain (Figure 1.1).

The proliferation of new depth sensing technologies in recent years has positively changed the climate of the automated vision-based human action recognition problem, deemed to be very difficult due to the various ambiguities inherent to conventional video. Depth sensors, such as Microsoft Kinect [1] [2] or Asus Xtion [3], and associated computer software have loudly been revolutionizing human-computer interactions by enabling users to control their virtual avatars without requiring any proxies but their own bodies. Therefore there is a high capacity in the field to develop such applications using these raising technologies.

1.1.1 Medical Motivation

Patients with motor disabilities experience dysfunction in motor control, strength and range of motions, which limits their ability to perform daily task and also in integration to community and vocation. Also in brain disorders like Alzheimer disease (AD) which is the most common form of dementia, neuromuscular weakness is one of the prominent symptoms among the patients. Physical activities for these patients have the same effect as for any other population and yield to significant fitness gain for patients. [4] Persons who were treated with both exercise and medical management were less depressed than those who were treated with medical management alone and showed marked improvement in their physical functioning, however studies indicate that just 31% of patients with motor disabilities perform exercises as recommended by the therapist [5] which give rise to other chronic health issues in patients such as obesity-related and cardiovascular problems. Proposed methods have potential and are aimed to leverage a platform either to enhance the recovery quality of the patients with a low cost home-based system and to offer the therapists a monitoring system to keep their patients under a more efficient surveillance.

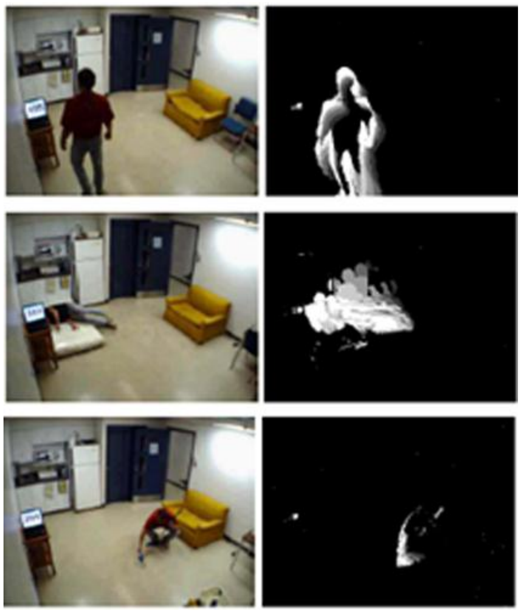


Figure 1.1: Left: Fall detection using 3D head trajectory extracted from a single camera sequence [6] Right: Visual tracking system for behavior monitoring of At-Risk children [7]

1.1.2 Motion Analysis and Depth Sensing Technologies

Human gestures are physical movement of fingers, hands, arms etc. to convey meaningful body motions. There are three classes of these motions that serve three functional roles [8]:

- The intention of ***Semiotic*** class is to communicate meaningful information such as goodbye gesture or the American Sign Language.
- The ***Ergotic*** function of the gestures is the capacity of the human to manipulate or interact with the environment
- The ***Epistemic*** class allows learning from the environment through tactile experience.

Each of these functions may be augmented using various instruments such as a handkerchief for goodbye gesture or a retro-active system to sense the invisible. Motion analysis is the interpretation of these types of movements by computers using underlying algorithms and enable humans to interact with machines with or without using a mechanical device.

However this ability of interpretation of gestures can be achieved through various sensors, vision-based sensors accompanied with computer vision have high ability to transform these gestures into effective input devices.

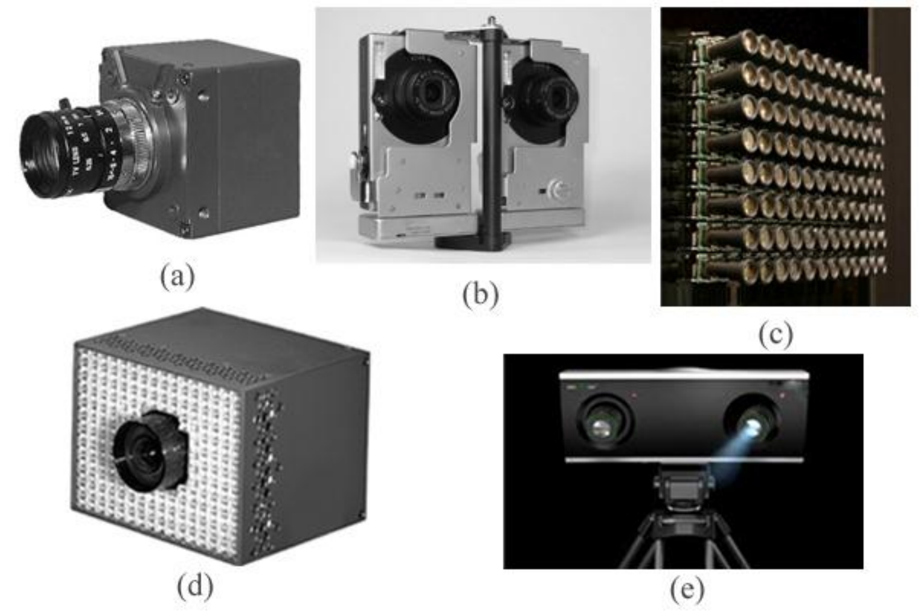


Figure 1.2: (a) RGB camera (b) Stereo camera (c) Camera array (d) Time-of-Flight (ToF), (e) Structured light 3D scanner

Vision-based analysis of human motion can rely on [9]:

- **Single Camera:** Normal RGB camera. Although not necessarily effective as the other types but it allows for wider accessibility.

- **Stereo Camera:** Two or more lenses each with separate image sensor next to each other used to determine depth in the scene by corresponding points in the images taken by each of the sensors.
- **Camera Array:** Is similar to stereo cameras, 3D representation of the environment can be determined using the relation between the cameras in the array. for example cameras can be placed in different corners of the room.
- **Time-of-Flight Camera:** Produce a depth image, each image of which encodes the distance to the corresponding point in the scene using time of a light pulse which can be transferred into distance. These cameras can be used to estimate 3D structure directly [10].
- **Structured-Light 3D scanner:** By using projected IR(infrared light) patterns and analysis of what is being seen, a depth map can be derived to reconstruct an image-based 3D scene [11] Microsoft Kinect sensor is one prominent example.

Each of the technologies differs in accuracy, resolution, range of motion, latency, cost and user comfort. Based on specific purposes of the applications and considering advantages and disadvantages of each, the best technology is selected (Figure 1.2) [9].

Structured-light 3D scanners among the others gain popularity and applicability as they provide the users with markerless detection of human body. These low-cost sensors with their convincing detection accuracy motivate the researchers to ideate about make use of these technologies in various research fields and developers to put them in use in useful applications which was not applicable before. The recognized gestures by these sensors can be used in a variety of human-computer-interaction applications. Gestures can be used as an event trigger and operate as a virtual controller [12]. It makes virtual environments more immersive and interactive. Gestures also can be used to identify the emotional state of the user. They also can be used to train people e.g. to identify wrong movements in dance [13] or sports [14] or can be used for therapeutic purposes (Figure 1.3) [15].

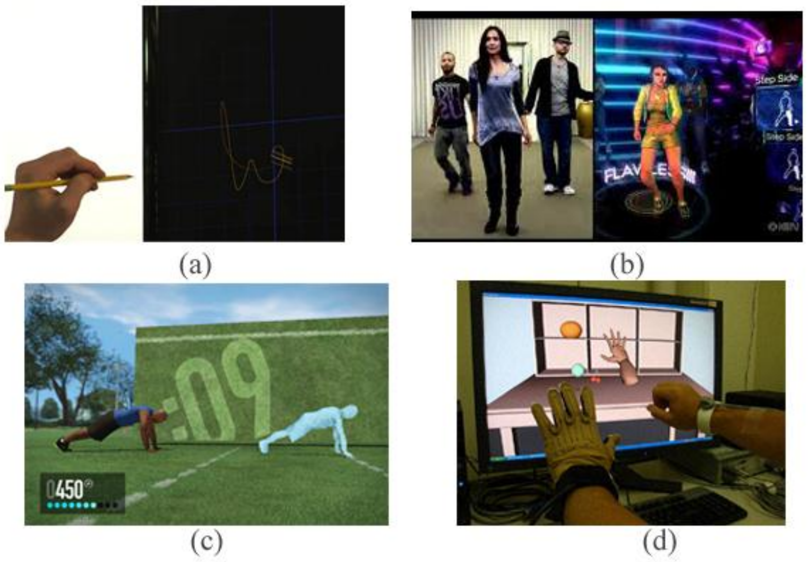


Figure 1.3: (a) *Leap Motion*, motion sensing for interaction in virtual environments (b) Playing and training dance using *dance central* (c) Physical exercise with *Nike*'s *Kinect Training* (d) Treatment of phantom limb pain patients

In this thesis, these kinds of depth sensors are our modality of interest because of the advantages they provide in context of our work.

1.2 Scope of the Thesis

Having all these technologies in hand and aforementioned motivations in mind, in this research study by employing machine learning approaches like SVM classifiers and Random Decision Forests, we attempt to analyze human motion problem and suggest a solution for feature selection problem which is a common problem in this field and finding the most effective features is always a challenge. We want to evaluate the effectiveness of various extracted kinematic features from depth sensors in different action recognition scenarios.

Incorporating ideas from various approaches and algorithms can yield a system which is capable of appropriate automated surveillance and treatment in homes and hospitals. Based on our proposed methods we want to develop a therapy platform for patients suffering from motor disabilities or fall detection in a room inhabited by patients.

1.3 Contributions

Interpretation of human behavior is a very interesting area in biometric and human computer interaction researches. It is important to identify actions of some parts of human body (in lower level) and whole body (in higher level) for some applications. And also developers are so interested to model human behavior in some other applications such as generating natural animation or graphics. One of the challenges is to choose the best features to achieve the best results in motion analysis. There are lots of features in literature which have been tested in different researches. To find the best feature set for specific applications one should do some sort of fine tuning between different features in hand. This notion of finding the best feature set for classifier can be achieved either with expert selection of application dependent features or using dimensionality reduction techniques such as principal component analysis or combination of both. In the area of model based action recognition and classification, time varying sequence of parameters such as velocities, angles, distances (Euclidian, Mahalanobis,…) are the most useful features [16]. Therefore to find out whether a set of features is a good set for an application or not in this work:

- A large set of invariant spatiotemporal features available in the literature, our defined features and combinations of them are extracted from skeletons in motion and are tested.

Related problems to automatic or semi-automatic analysis of complex data such as text, speech, n-dimensional medical images etc. can be categorized into a set of machine learning tasks [17]. Recent popularity of decision trees is due to this fact that ensembles of slightly different trees tend to produce much higher accuracy on previously unseen data, a phenomenon known as generalization [18]. Hence in this work:

- We introduce a discriminative RDF-based feature selection framework capable of reaching impressive action recognition performance when combined with a linear SVM classifier[2].

A large number of applications need to estimate movements of body parts. Interface designer for such systems is responsible for designing of a system which is capable to recognize these movements as embodying meaningful actions [19]. To tackle this problem using a machine learning approach requires a collection of datasets containing all the natural varieties of the movements in the system. Absence of such datasets collected by depth sensors in action recognition community motivated us to address this issue:

[2] A part of this work is published at “International Conference on Image Analysis and Recognition” (ICIAR 2013)

- We present a novel therapeutic action recognition dataset (WorkoutSU-10) to be prospectively used by the action recognition community.
- As an extension of the dataset we also collected a dataset for assessment of fall detection among subject (we call this extension the “fall dataset”)

After performing the entire test and obtaining assessment results, at some point there must be a benchmark to evaluate these developed methods in a real life occasion. In order to do this:

- KinematEval is an application which is under development to record, learn and evaluate actions. Its aim is not only recognition of actions but also analysis of actions in deliberate time points and time spans.

1.4 Thesis Outline

This thesis is organized as six chapters including the Introduction chapter. A brief review of early and recent developments in human action recognition techniques is presented in chapter 2. In chapter 3, we describe the kinematic feature collection we used in this work, the aggregation methods we employed to generate baseline performance and our RDF-based discriminative feature selection approach. In chapter 4, we provide experimental results as well as some insights about selected kinematic features. In chapter five we describe the properties and capabilities of the developed application based on the methods we proposed in early chapters. And finally in the last chapter, the conclusions and future work are presented.

Chapter 2

Background and Related Work

In this chapter, first, we briefly discuss about the background of human motion analysis, related work and recent developments on this active topic. Next, we provide a review of conventional and recent motion capture devices by emphasizing on their applications, especially those are developed for treatment purposes.

2.1 Action Recognition and Its Applications

Human motion analysis has been divided into sub-topics such as gesture recognition [20], facial expression recognition [21] and action or movement behavior recognition [22]. Full-body action recognition will require a unified recognition approach for different body limbs from movement of hands and feet to facial actions. Our focus in general is full-body action recognition. Generally, the process of naming actions in the simple form of action verbs using sensory observations is called action recognition. Technically an action is a four dimensional sequence of movements by human agent during the performance of a task. This way an action is a four-dimensional object which may be further decomposed into spatial and temporal parts [23].

Traditionally, motion capture systems require markers are attached to body but because these systems are obtrusive and expensive a markerless solution would be preferable. Over the last decade development in vision-based motion capture systems provide such a solution, using camera sensors [24].

Additionally, for human motion analysis systems viewpoint play a significant role [25]. Due to limitations derived from this problem, a large number of applications obstructed to reach into a wider applicability. In recent years growing number of researchers pay attention to this problem and a large number of attempts and progresses have been reported [26].

Therefore a markerless view-invariant system seems to be an ideal motion analysis system.

2.1.1 Action Recognition Systems

The major components of an action recognition system and their typical arrangement are illustrated in Figure 2.1 [23].

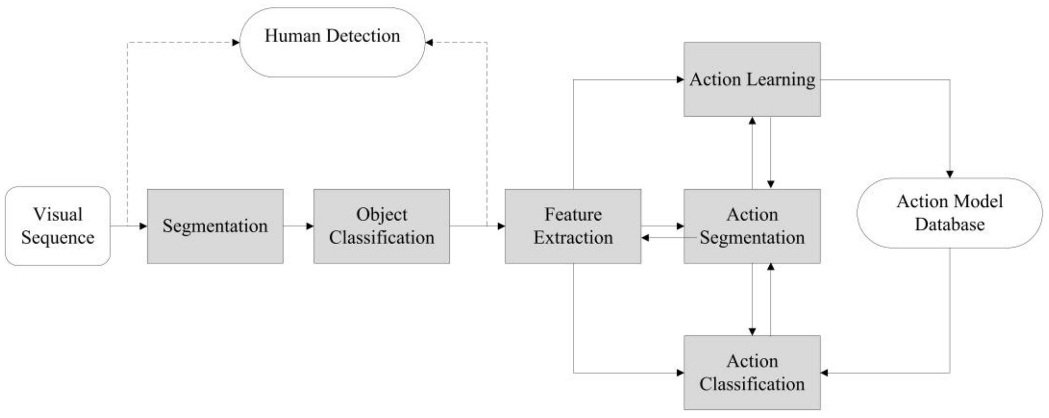


Figure 2.1: Components of a generic action recognition system

We will have a brief overview of each component. **Human Detection** tries to separate people from the background. It is the fundamental component of the human motion analysis system that is the subsequent pose estimation and classification quality considerably depends on performance of this component [22] [27]. The underlying parts of the human detection component are segmentation and object classification. Motion **segmentation** is used to distinguish regions related to moving objects as a potential targets in the scenes. This supply the system with a focal point for later processes such as tracking and activity analyses [22]. Two conventional segmentation methods for human motion analysis are *background subtraction* and *optical flow*.

Background subtraction is extensively used for segmentation, especially in static scenes. It identifies moving object from static background with a pixel-by-pixel comparison of the frames with a reference frame which is called background frame. For example Lo [28] proposed use of the median value of the last n frames as the background model and then try to estimate and update the background. This algorithm also can handle some of the inconsistencies due to lighting changes.

Optical flow on the other hand describes coherent motion of points or features between consecutive frames. This method is vulnerable to image noise, non-uniform light and also is sensitive to motion discontinuities [29] (Figure 2.2 Left).
Detected regions in motion segmentation may correspond to different objects in the scene. This is an issue in realistic scenes. Therefore **object classification** is required under these assumptions to distinguish human from other moving objects. Two main approaches for object classification is *shape-based* and *motion-based* classification. Shape-based approach use different shape information of the moving object like point, blob, circle, etc. to identify object. Since there are large number of varieties to human body motion shape, this approach is unable to accurately identify the human body. Motion-based approaches use periodic property in articulated human body to distinguish human from another moving objects. A hybrid approach shows better results over each of the approaches [30] (Figure 2.2 Right).

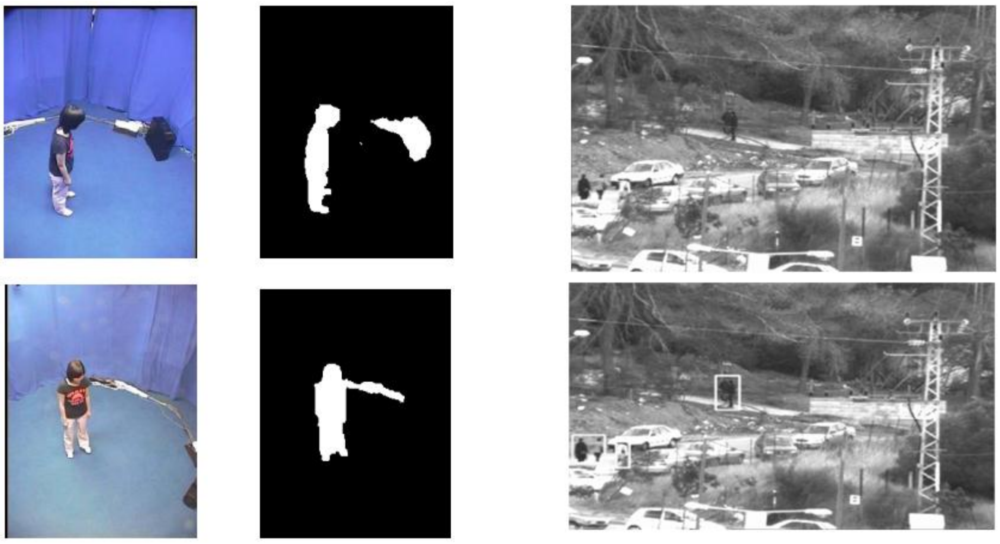


Figure 2.2: Left: background subtraction using Graph-cut method [31] Right: object detection using shape-based and motion-based approaches [30]

Feature extraction is one of the main tasks in action recognition and it consists of extracting posture and motion cues from visual input that are discriminative due to human actions. Various representation methods can be used such as human body models or silhouette images.
Vision-based human action recognition techniques can be classified considering different criteria e.g. image features, statistical models or pose-based models. But one useful classification proposed by [23] which classify different techniques into *spatial* and *tem-*

poral structures of actions. Spatial action recognition can be based on global image features, parametric image features or statistical models describing spatial distribution of image features. Temporal techniques can be based on global temporal signatures such as stacked image features representing action from start to finish or grammatical models. Spatial representation of actions is classified into three groups:

- Body models
- Image descriptors
- Statistical models

Body models: in this approach pose of human body is extracted from consecutive frames and by extraction of variety of features, action recognition takes place. Most of these models describe human body by a kinematic tree composed of linked joints which each of them has a number of degrees-of-freedom. These models can be expressed either in 2D or 3D. 2D models often are suitable for motions parallel to image plane. 3D models represent body as rigid segments; each one has three rotation axes. Large number of degrees-of-freedoms and high variability of human body shape is two major difficulties for these models.

Segments in 2D models are defined with rectangular or trapezoid shape patches [32]. 2D models work for direct recognition approaches where it uses labeled body parts without take them into 3D space. One common example is stick figures [33]. 3D model segments are volumetric or surface-based. In volumetric models, human body's kinematic shape depends on several parameters which describe the model with cylinders [34]or spheres [35]. Parameters of these shapes often are considered fixed. But due to large variability among people, fixed parameters cause inaccurate pose estimation. Some works use an initialization step to adopt observed person to specific pose [36] but this also will not work for applications such as surveillance. A large number of 2D and 3D kinematic and parametric body models represented within years, you can see some examples in Figure 2.3**.**

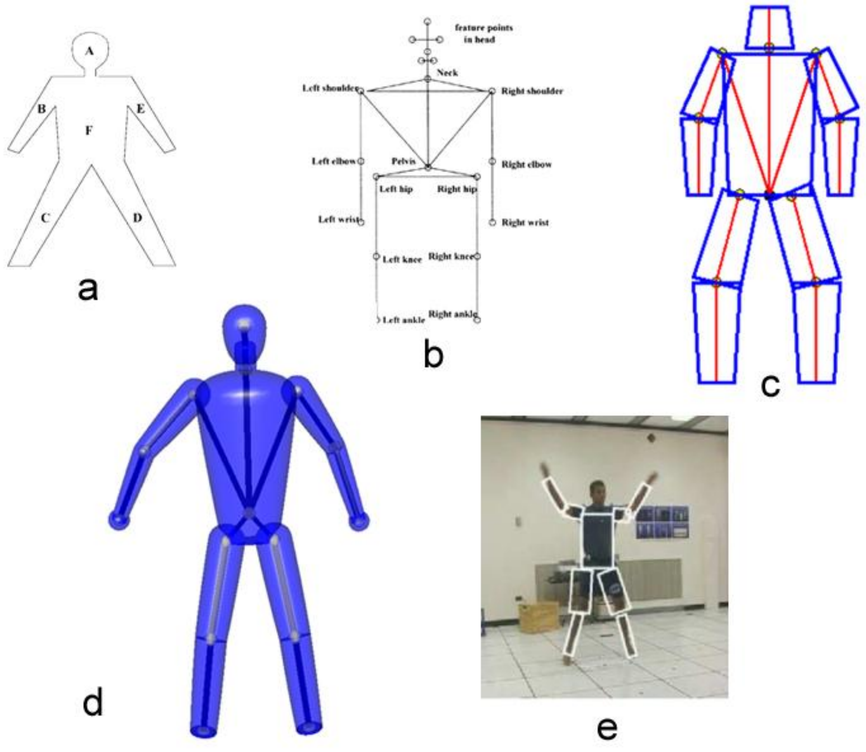


Figure 2.3: Human shape models and kinematic models: (a) 2D contour human model [37] (b) a stick figure human model [38] (c) a 2D model with segments as trapezoid-shape patches [32] (d) 3D volumetric model consisting of superquadrics [39] (e) body model based on rectangular patches [40]

Image descriptors: or appearance-based features. Appearance of people is different in images because of lighting conditions and different texture of cloths. Instead of kinematic features we can take image descriptor of body in a scene. This way, we don't need to know whole of the knowledge about the model that appears in the image. Some examples of these descriptors are silhouettes and contours, edges, 3D reconstructions and color (Figure 2.4). Silhouettes and contours can be recovered robustly when the background is static. Silhouettes are insensitive to texture and color of the surface [41]. Performance of silhouette extraction is limited due to noisy artifacts such as noisy background subtraction and sometimes it is impossible to recover degrees-of-freedom because of lack of depth information. Also extraction of edges can be done robustly. An edge is a substantial change in intensity at different sides of image and is invariant to lighting condition [24]. When multiple cameras are used, a 3D reconstruction can be created from silhouettes. This is not possible with single camera due to lack of depth information. A common technique is volume intersection [42]. Because color and texture of body parts are almost remain unchanged, they can be used to model human body. For example skin color is a good cue to find head and hands. These appearance features can be described by Gaussian color distributions or color histograms.[40]. Combination of these descriptors proves to be more robust than using them individually e.g. the silhouette information combined with color [43] [44].

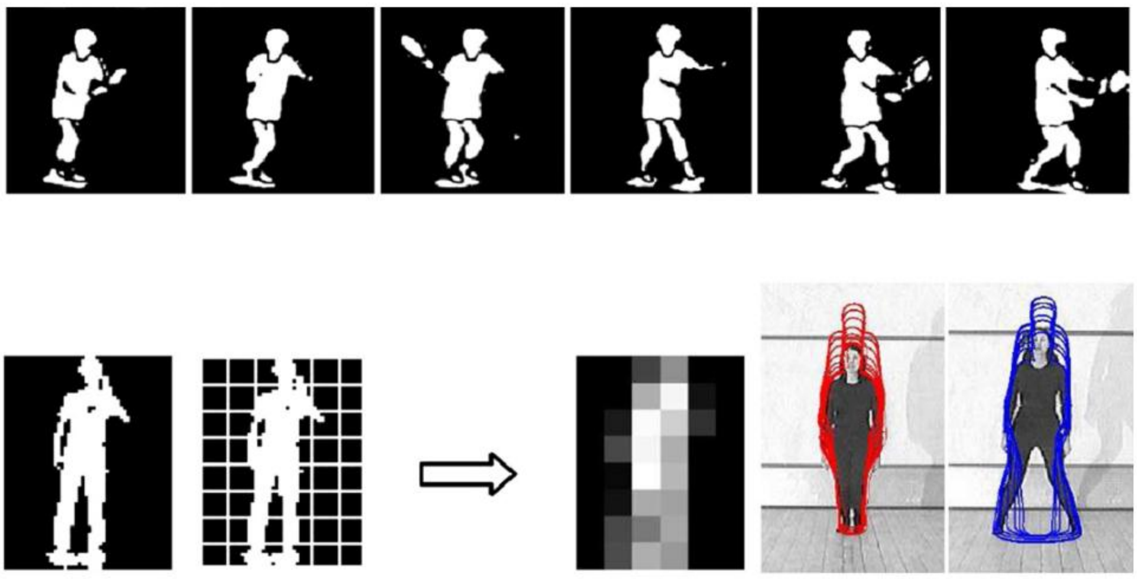


Figure 2.4: Image descriptors, Top: silhouette of strokes by a tennis player [45] down: silhouette pixels are accumulated in a grid and in spline contours [46] [47]

Statistical Models: in this approach visual input (video/image) decompose into smaller regions without getting linked to body parts and then action recognition take place based on statistics of local features from all regions. These approaches are based on bottom-up strategies, where they first detect interest points in the image and then assign these points to a preselected vocabulary features and do classification bag of features approach such as space-time interest points, like the method proposed in [48]. Statistical methods based on local features promise the same advantages as static object recognition and can easily apply to difficult scenes like movie clips from internet which is hard for model based approaches [23] (Figure 2.5 top).

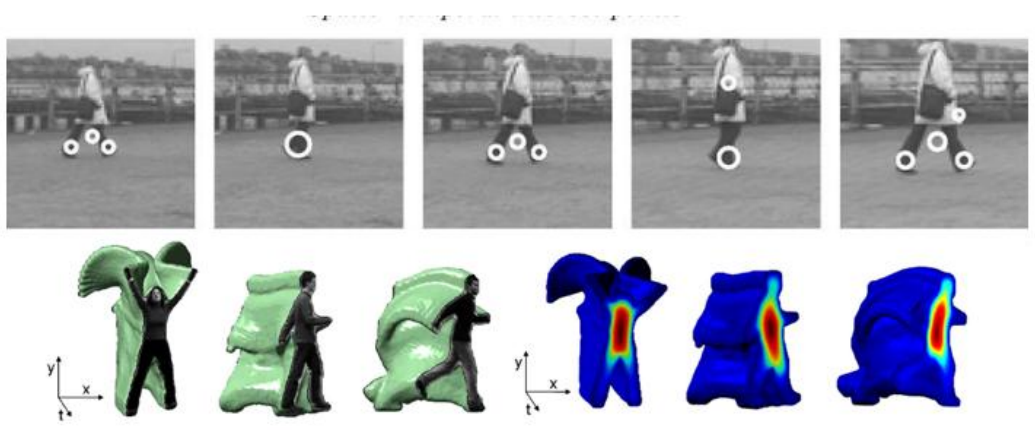


Figure 2.5: Top: Results of spatio-temporal interest point detection for a zoom-in sequence of a walking person [48] Down: Action templates space-time shapes [49]

So far we explained the spatial representation methods and now we will briefly describe the temporal representations. Temporal representation of actions is mainly divided into three categories:

- Grammars
- Templates
- Temporal statistics

A natural way to estimate a dynamic system by feature observations is to group features into similar groups or states and learn how to temporally transition between those states. Such models are so called **grammars** and the most prominent model of this kind is hidden Markov model [50]. Some methods try to learn appearance of complete temporal blocks of actions which called **templates.** Unlike grammars, templates cannot represent variations in speed, time and style of actions and more advanced techniques such as dynamic time warping (DTW) [51] may be used to deal with this issue. **Temporal statistic** approaches attempt to build statistical models of the appearance of actions, without an explicit model of their dynamics. Typical examples of this approach are methods that learn an appearance model of action from a single characteristic keyframe as in a photograph [52].

In most of the cases for both training and testing, the action recognition approaches work on the visual streams where each one shows a single action from start to end. Finding a generic method to **action segmentation** is a difficult issue for breaking the input visual streams into segments. Boundary detection and sliding windows are common approaches to deal with this difficulty.

Action learning and **classification** components of the recognition system are the steps of learning statistical models from the extracted features, and using those models to classify new feature observations. The big challenge in managing of large statistical data is to deal with the considerable variation of an action especially when it is done by different subjects where those subjects are of different size, gender, speed and style. Simple actions such as walking and waving etc. which might look clear and defined to us can have very large variation in practice. And also semantically similar motions may not necessarily be numerically similar [53]. So the designed system should contain an action model which enables to identify characteristics of each action and be adaptable to all forms of variations of actions [23].

2.1.2 Kinematic Performance Assessment Systems

The term "kinematics" lies stress on that these types of features are independent of any forces that taking action on that object or mass of that object. And they only capture motion information of that particular object. This kind of feature is useful for recognizing the actions in a way that its representation is independent of the physical features of the subjects while they are performing the actions. Examples of kinematic features are velocity, position, height and width of a set of bounding boxes, which contains the subject in every frame of the sequences. These features can be useful in recognition of actions by exploiting the spatial and temporal relationship among the features. Here we describe some examples of the systems that take advantage of this kind of features for their evaluations.

Hernández et al [54] propose an action recognition system to recognize human actions in 2D sequences. The system is based on real-time tracking of the subject and extraction of kinematic features from human activities in video sequences. It consists of several modules which they are responsible for particular tasks starting from preprocessing of the video sequences to action recognition module (Figure 2.6).

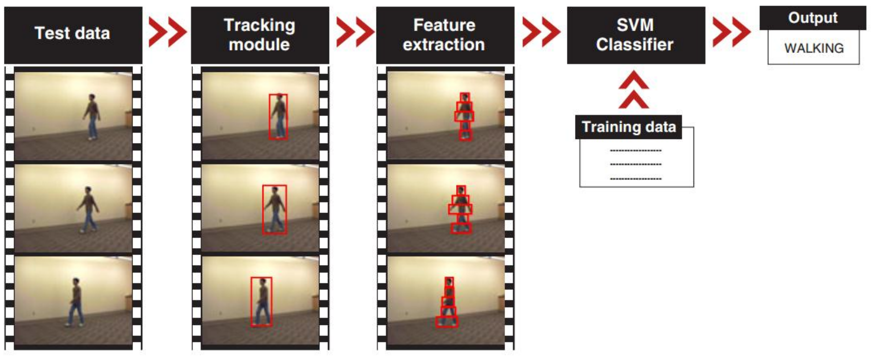


Figure 2.6: System modules overview [54]

In the first module a hybridization of a particle filter and a local search procedure used to speed up the weight computation process. In feature extraction module, the tracked person is represented by dividing his silhouette into rectangular boxes. Then, the system computes the statistic of the evolution of these rectangles over time and finally in the action recognition phase it passes these statistics into a support vector machine classifier to classify the actions. The feature selection process in the system is based on expert human knowledge and it uses heuristic rules. The rule model can be validated and check for consistency of the rules and furthermore the rule set can be completed by adding new rules progressively. For example the following are two instance of the rules used to select features: The bend down action implies change in the height of the bounding box and the jumping jack action produces changes in the bounding box's height and width. They experiment their system by three available dataset: Weizmann dataset, UIUC and the IxMas that in total they consist 566 sequences. On Weizmenn dataset they obtain 96.66% success rate. On UCIC dataset they obtain 99.58% success rate and they beat the other available performance results in the literature. They also obtain 94.44% accuracy on IxMas dataset and beat the other available reported results. State-of-the-art experimental results of the system reveal that the proposed system has potential to be applied into 3D context.

In [55] they propose a set of kinematic features that are extracted from the optical flow and they use it for human action recognition in videos. The extracted kinematic features include: vorticity, divergence, symmetric and anti-symmetric flow fields, second and third invariants of flow gradient and rate of strain tensor. Each of these features is computed from a sequence of optical flow images, creates a spatiotemporal pattern. The representation of dynamics of optical flow is captured by these spatiotemporal patterns in the form of kinematic modes, where these kinematic modes computed by applying principal component analysis (PCA) on the spatiotemporal volume of kinematic features instead of optical flow data itself. They use multiple instances learning (MIL) for classification, in which, each video is represented by a bag of kinematic modes. You can see the architecture of their system and its data flow between system components in Figure 2.7.

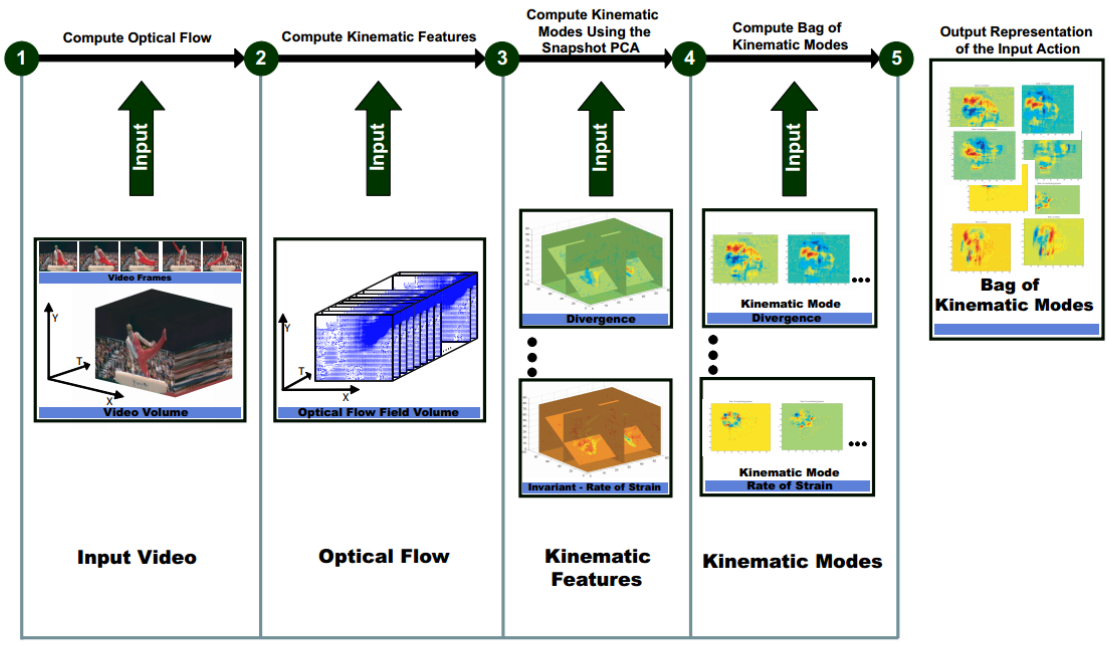

Compute Optical Flow
Compute Kinematic Features
Compute Kinematic Modes Using the Snapshot PCA
Compute Bag of Kinematic Modes
Output Representation of the Input Action
1
2
3
4
5
Input
Video Frames
Video Volume
Optical Flow Field Volume
Divergence
Invariant - Rate of Strain
Kinematic Mode Divergence
Kinematic Mode Rate of Strain
Bag of Kinematic Modes
Input Video
Optical Flow
Kinematic Features
Kinematic Modes

Figure 2.7: Process of representing a video in terms of modes of kinematic features [55]

In the first phase of the pipeline a video containing an action is fed into the system as input, which computes the optical flow between the consecutive frames of the video and produces a stack of optical flow fields. Then, in second phase these stack of optical flow field is taken as input and the system extracts the kinematic features out of it and produces a separate spatiotemporal volume for each feature. Next step is applying of the principal component analysis on the extracted features and construction of the kinematics modes out of the PCA components. Finally, the input video is represented by a bag of kinematic modes pooled from of all the kinematic features. Next, each video is embedded into a kinematic mode based feature space and the coordinate of the video in that space is used for classification by using a nearest neighbor classification algorithm. They evaluate their proposed recognition algorithm on two publically available dataset: Weizmann and KTH action datasets. They show that how using these kinematic features improves the classification performance by comparing them to optical flow classification alone. In 5-mode kinematic features they obtain 94.75% classification accuracy and beat the 85.8% accuracy of optical flow classification performance. On KTH dataset that consists of 6 actions they achieve mean accuracy of 87.7%. They obtain their best accuracy when they use all of the kinematic features. Their algorithm has two major flaws. First, the proposed kinematic features are not view-invariant, different view will produce different optical flow on the images. The solution can be taking the view into account and produce separate kinematic feature based representation for each view. The second problem is occlusion, especially when an important body part is occluded; it affects the performance of the system severely.

2.1.3 Fall Detection

The quality of life of individuals is highly dependent on their motor and functional abilities. A lot of research has been done in this regard to develop systems and algorithms for enhancing the motor ability of elderly and patients. The advancement of the camera, sensors and computer technologies make the development of such systems a feasible scenario.

Population of elderly people is increasing in recent years and without receiving enough care they will lose their independence into a high degree and their health would have been in a great risk. Thus an intelligent monitoring system that allows elderly people to live safe and to have more independence is more than needed. Increase in fall and fall related injuries and decrease in qualified staff hires to prevent these injuries has reported in recent years for example in England 32% of incidents related to safety of patients account for fall events and fall incidents are 40% more likely to happen in hospital than in other locations or industries [56]. For elderly people population possibility of falling is approximately 50% more than general population [57]. Some traditional approaches such as using belt size button which alarms when patient pushes a button experimented but those are not a robust solutions for this problem since for example that will not help in case of unconsciousness fall incidents .Therefore developing such automated intelligent systems will minimizes such incidents while requires less staff employment.

Fall detection approaches are categorized in three classes: wearable devices, ambience sensors and camera based (vision based) methods. Here we will have a brief review of vision based fall detection methods which can be categorized as: body shape change, inactivity and head motion.

As mentioned before image analysis for action recognition requires accurate shape modeling methods. Shape modeling using spatiotemporal features supplies the required important information for event recognition algorithms. In [58] they describe an address-event vision system to detect accidental falls in elderly. They extract changing pixels from the background and calculate motion contrast. This value is equivalent to the change of the image reflections under constant lighting. Finally they detect fall event by calculating an instantaneous motion vector. Also Foroughi et al. in [59] propose a fall detection system using the combination of the Eigenspace and integrated time motion images (ITMI). ITMI is a type of spatiotemporal database that includes information about motion and time of motion occurrence. This combination leads to extraction of eigen-motion and finally a MLP Neural Network is used for an accurate classification and determination of fall event. Unlike other fall detection systems only take a limited action patterns into account, they consider a wide range of actions in their system such as normal daily life activities and abnormal behaviors and unusual movements.

Some systems are based on analysis of shape change and inactivity detection. Miaou et al [60] propose an approach using an MapCam omni-camera. They take some personal information such as weight and height of the subjects into account in image processing phase. For object segmentation they apply a background subtraction algorithm on the images and for more accuracy, a noise reduction is applied to remove the noise during the segmentation. In order to use shape change they employ a bounding-box method which surrounds the subject. Changing in the ratio of width to height of the bounding-box in consecutive frames is a clue which indicates how much the fall event is likely to occur. [61] propose a robust shape matching method to classify fall detection motion by analyzing human body shape deformation. The system can works with one uncalibrated camera or multi-camera setup using an ensemble classifier to improve the detection results. They characterize fall by large movement and change in human shape. In common human activities shape of the body change progressively and slowly but in fall the change will happen drastically. Using this, they detect fall during video sequences by quantifying human shape deformation following these steps: first a silhouette edge point extraction is performed. Silhouette is obtained by a foreground segmentation method and edge points are extracted by a Canny edge detector. In second step the detected edges are matched through video sequence. In third step a shape analysis perform by two deformation measure (mean matching and Procrusts distance). Finally, they detect fall using Gaussian mixture model (GMM) classifier. The error rate of the system reduced to 4.6 and 3.8 percent by using the two deformation measures respectively (Figure 2.8).

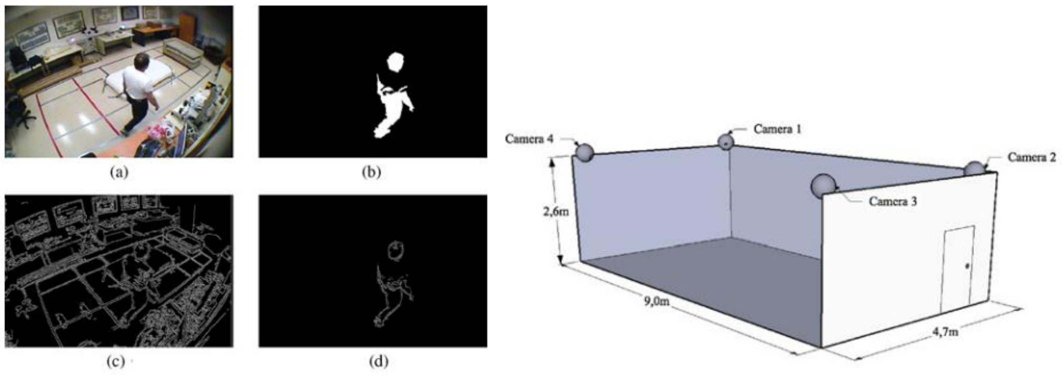


Figure 2.8: System’s camera configuration and its algorithm’s feature extraction steps

In addition, Posture information also can lead the detection system toward accurate results. There are systems such as [62] and [63] where they use this information to achieve better detection results. in [62] they analyze behavior by classifying the posture of the monitored person and consequently detecting the corresponding event and alarm. First, the projection histograms of each person are computed in each frame and then, a comparison with probabilistic projection maps stored during training phase, perform for each posture. Average accuracy of their method reaches to 95% and the experimental results indicate the system is also good in dealing with challenging conditions like occlusion. In [63] they develop a two-layered Hierarchical Hidden Markov Model based (HHMM) motion modeling. The first layer consists of two states, a standing and lying pose. 3D angle features and image plane projection also has taken into account in first layer to track the orientation of the subject. After an image rectification process they drive theoretical properties which make it possible to bind the error angle introduced by the image formation process for standing posture. This allows them to identify the non-standing poses and thus robustly analyze pose sequences against a given model.

Head tracking is another method that is used for determine fall. Usually state models are used to track the head based on the magnitude of the movement. Rougier et al.’s method [6] is based on three steps: head tracking, because head usually is visible in the image and has large movement during the fall. 3D tracking; they track head with a particle filter to extract a 3D trajectory of the movement and fall detection, where they report fall using 3D velocities of the head computer from the trajectory. A 3D ellipsoid is used for bounding around the head and to compute the trajectory on 2D image frame (Firgure 2.9).

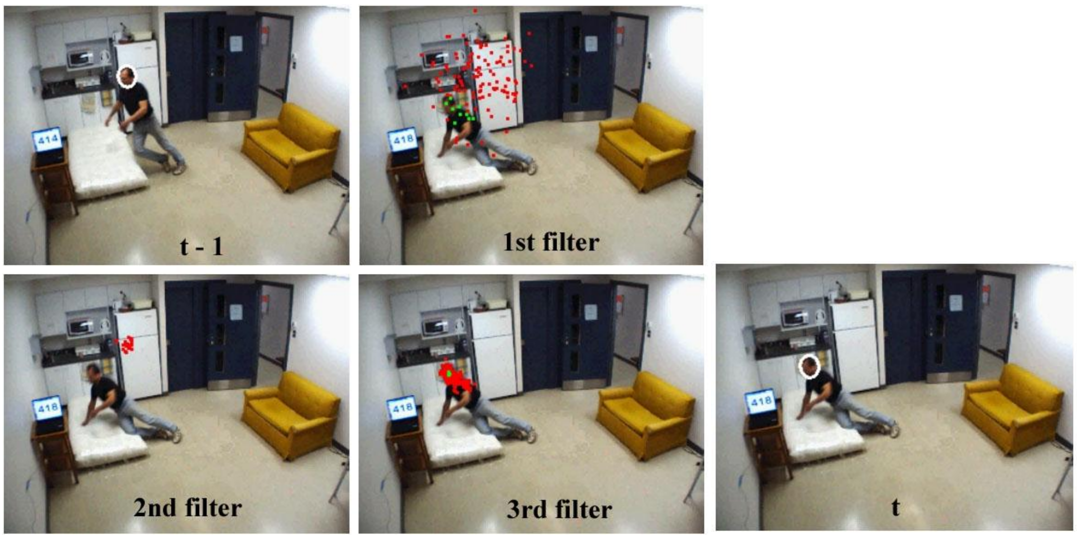


Figure 2.9: steps of the head detection algorithm using trajectory of the head position

Hazelhoff presents a system design, aiming at detecting fall incidents in unobserved home situations by using two fixed, uncalibrated, perpendicular cameras. The system consists of five modules. The first module is object segmentation where at each moment coming from both of the cameras foreground obtained by background subtraction. Then, an object detection algorithm is applied on the images to find non-human objects in the scene. For human objects direction of the principal component and variance ratio computed from both of the cameras. By using the features from previous frames, fall can be determined using a multi-frame Gaussian classifier. The head position is tracked by skin color information. This head position is used to reject false detection (Figure 2.10).

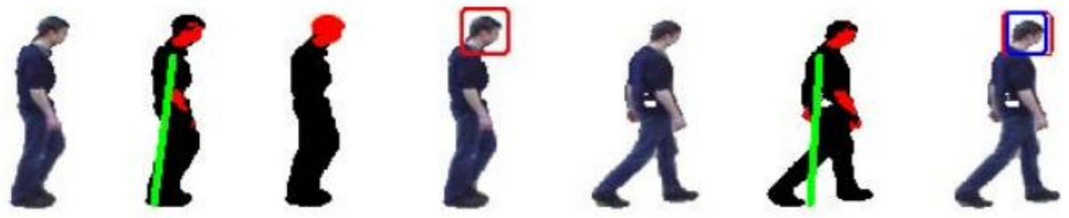


Figure 2.10: Feature extraction of the fall detection algorithm, head detection, PCA calculation and skin color detection

The system obtains accuracy level of 100% for un-occluded video sequences but occlusion reduces the accuracy to 90%.

In [64] they combine information about the subject’s orientation and with inactivity information extracted using a contextual model. The system interprets fall occurrences differently depend on the location, duration and time of the events. The context model is learnt during the monitoring task without human intervention and automatically adapts to the changing activity patterns of the monitored subject.

2.2 Motion Capture

In 1973 Johansson which was a psychologist, conducted an experiment to visually percept biological motion. The experiment was called Moving Light Display (MLD) [65] and Johansson used reflective markers in position of skeletal joints of human subjects and recorded their motion. Next, he asked subjects to identify known body movements such as walking or running etc. just by watching the joint movements (Figure 2.11). This was the beginning of the motion capture.

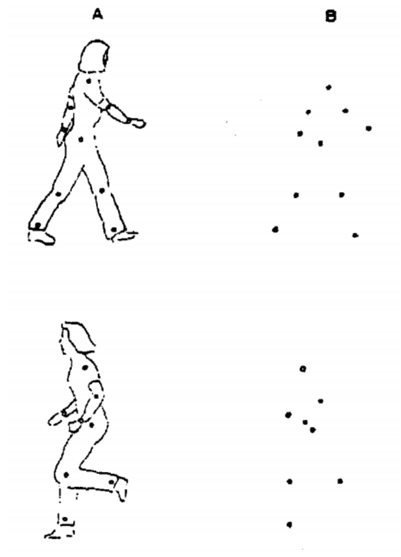


Figure 2.11: Johansson’s MLD experiment in 1973

Motion capture is analysis of a scene, resulting in a mathematical description of the movement or as Menache defines: “Motion Capture is the process of recording a live motion event and translating it into usable mathematical terms by tracking a number of key points in space over time and combining them to obtain a single 3D representation of the performance”. [66] simply defines motion capture as the process of capturing the large scale body movements of a subject at some resolution. Development of motion capture technologies in the past three decade gave birth into a lot of applications and advances in research field of human motion analysis. Here we will have a very brief review of traditional and new kinect-based motion capture technologies.

2.2.1 Traditional MoCap

Motion capture technologies are focused on three main approaches; electromechanical, electromagnetic and optical tracking systems. In general, motion capture devices are based on active sensing or passive sensing. In active sensing, some devices and sensors are placed on subject's body which they transmit or receive real or artificial signals. In passive sensing the device do not effects the surroundings and do not need to generate new signals or wearable hardware e.g. visual light or electromagnetic wavelengths [67].

Electromechanical systems are wearable body suits which have a variety of sensory or measurement devices at fixed part of the suit. When the subject changes his position the sensory devices detect and measure these small changes and report the results. These systems report accurate results but instead they are restrictive because of their weight and size that sometimes restrict the freedom of movement of the subjects. This is the major drawback of these systems which hold back the subject from performing the actions. This disadvantage can be serious when the system is planned to deal with scenarios like clinical applications.

Electromagnetic approaches are capable to capture greater range of motions. They are placed at key points of the body and they are responsible for extract the position and also the orientation. These systems are lighter than electromechanical systems but still they have disadvantages like connection between sensors and transmitters or their attached wires. Advances in active sensors like magnetic trackers, accelerometers, acoustic sensors and optic fibers have been made them cheaper, lighter and easier to use but they are still cumbersome because of need for special hardware. Therefore, touch-free computer vision based approaches could be an attractive alternative. Here we will describe some of the active sensing devices.

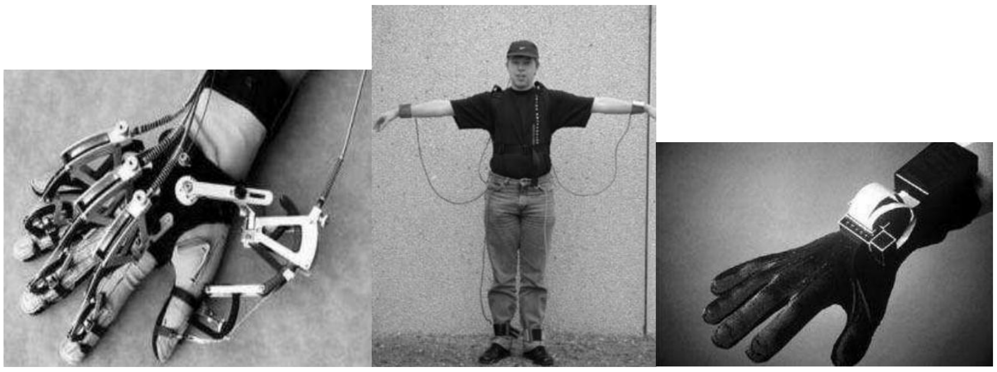


Figure 2.12: Active sensing motion capture devices Left: Mechanical glove, Middle: Electromagnetic sensors Right: Fiber optic Glove

Mechanic devices are attached to some movable parts and when they moved or bend, they generate signals which reflect the configuration of the parts. Accelerometer is a device which measure acceleration of the object it is attached to. The device calculates it by measuring the deflection caused by movement and converts it into electrical signal. Acoustic devices use various set of sound sensors to capture sound waves transmitted from a sensor attached to the subject. By triangulation or use of sound waves phase calculation the 3D position of the sensor can be found. Optical fibers are mechanical devices that placed along with the subject's limbs and when the subject bends the limb/fiber [68] they signal. These devices mainly used in kinematic systems where the goal is to find the position of the joints over the time (Figure 2.12).

2.2.2 Kinect-Based MoCap

In this section we mostly will emphasize on applications of the depth sensors because our work stress over motion analysis systems that predominantly put these sensors to use. In the second part of this section we will describe applications which they apply machine learning for treatment or training purposes.

The problem with active sensing devices motivates the use of passive sensor capturing devices. In passive sensing the idea is to use the images obtained from a camera or depth sensor (chapter one) and capture the motion based on those images. To reduce the difficulties of these sensors many of the developed systems use markers. Markers can reduce amount of information streams from the sensor. Even though the use of markers was good idea but it is still cumbersome in many of the situation and because of that the researches move to more pure MoCap systems which they use raw data to capture the movements [66]. However, due to difficulties arise from projection of the 3D scene into 2D image and the amount of visual information, the idea of depth sensing seems to be a noble and at the same time a difficult one to handle. The new depth sensor technologies that we introduced some of them in chapter one comes to rescue. Depth sensors like Microsoft Kinect[3] captures the visual information of the environment and by using machine learning algorithms tries to detect human body in the scene. In recent years lots of research has been done using these sensors. Here we will describe some of the applications and systems and the researches have been done in the field of human motion analysis using Kinect and depth sensors.

[3] You can find details about the kinect sensor and its underlying algorithms in appendix 1

One of the popular applications in this field is choreography or the evaluation of dance performances. Essid et al. [69] propose a dance training and evaluation framework in an online virtual environment. A dance expert delivers trainings to online students and evaluates their performance and provides them with meaningful feedback. Skeleton movements of the teacher and students are acquired using the Kinect sensor and aligned for score calculation. For rating they compute the Quaternionic Correlation Coefficient (QCC) for each pair of joint position signals. Another choreography framework using Kinect sensor presented by Alexiadis et al. [70] they provide a novel system that automatically evaluates dance performances. They use their so called gold-standard to evaluate the performance and feedback the user visually in a 3D virtual environment. The system is based on an online interactive scenario where dance choreographs can be set, altered, practiced and refined by users (Figure 2.13).

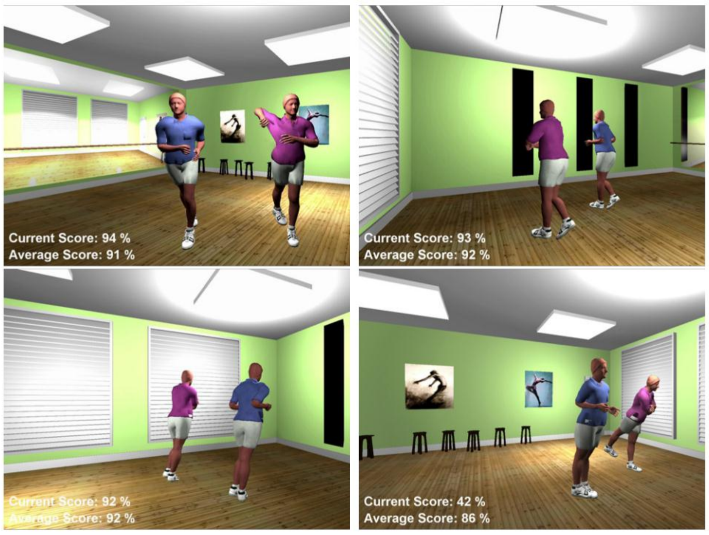


Figure 2.13: Dance performance evaluation

For alignment of the signals before evaluation, they perform a three step preprocessing. Then, they compute three score to evaluate dancer's performance: joint position score, where they use quaternionic correlation coefficient (CC) to calculate the joint position score for position signals. Also a velocity score is calculated and consequently used for calculation of a 3D flow score. They use a weighted mean by assigning different weights to different joints. The final score for dancer's performance is computed and compared to professional choreographs grand truth scores to find out how good the dancer has performed the dance actions.

Sport training is one of the applications that there is an emphasis over it by emerging of the Kinect depth sensor. For example the [71] propose a Kinect-based system that automatically recognizes sequence of complex karate movements and measures the quality of the performed movements (Figure 2.14). Their system consists of four modules: skeletal representation, pose classification, temporal alignment and scoring. They use dynamic time warping (DTW) for alignment and scoring of the action sequences. The system obtains competent recognition accuracy in tests. The recognition accuracy for actions executes in fixed stances is 97% and for actions starting and ending stances is 97.98%.

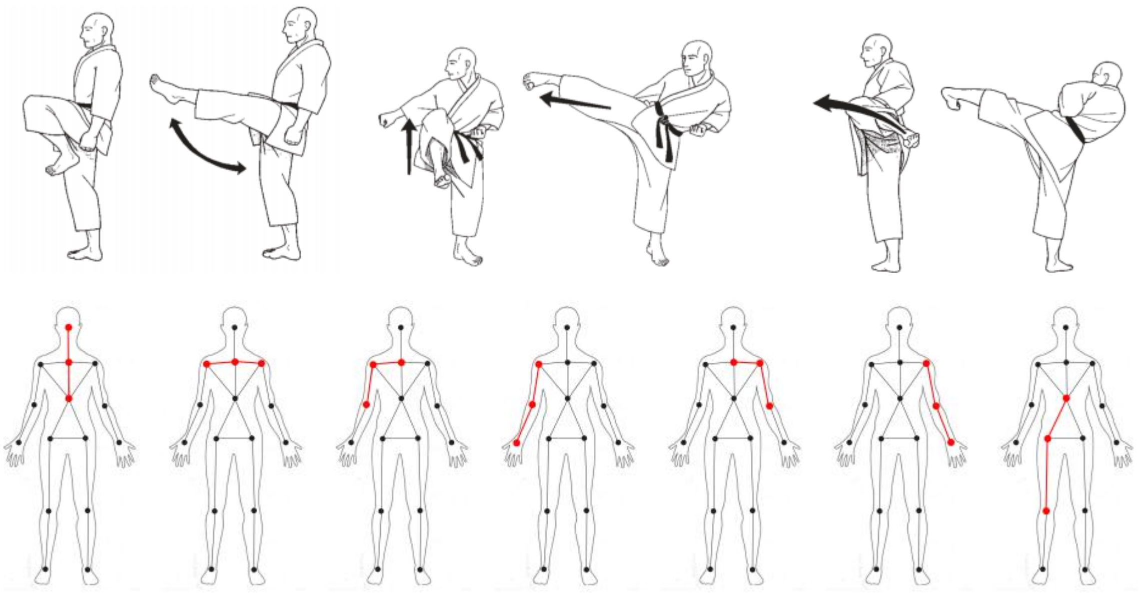


Figure 2.14: The selected techniques (up) and the skeletal representation (down) in proposed system

For analysis of motions parallel to the image plane, the work in [72] uses 2D kinematic cardboard models to model limbs as planar patches. Each of the patches has different parameters to rotate and scale according to 3D motion. Another approach is to model the body in 3D as rigid segments with three orthogonal constraint rotations on each joint. The work in [73] defines the constraints on limb ends. As color and texture of the body remain unchanged during the motion, the work in [74] uses color histograms to describe edges and appearance cues of individual body parts.

The use of 3D geometrical information provides a clear advantage over using 2D image-based features. The work in [16] has investigated the two categories of approaches using a wide range of features and has shown that even with high levels of noise, the recognition process benefits from using pose-based features. As skeletal kinematic models encode key parameters of the limbs, they are considered as very powerful representations for a real-time motion analysis of the human body, although such models are difficult to extract and track from conventional video. The emergence of real-time depth cameras has greatly simplified the extraction of human skeleton models and the tracking of skeletal key points such as joints [2] [3].

In [13], Raptis et al. present a real-time gesture recognition platform for classification of skeletal wireframe to evaluate dance gestures. The system includes a specific angular representation of the skeleton using a spherical coordinate system centered at each joint. They group skeleton into three categories (torso, first and second degree joints) and characterize each joint with azimuth and elevation angles. Correlation and energy profiles have been used for evaluation of the dance gestures. The work in [75] focuses on real-time estimation of body poses using depth images and uses the iterated closest point algorithm to tracking the skeleton of known size. In [76], the authors present a classification algorithm based on logistic regression, which also is capable to cope with the latency problem in interactive action-based systems. Their proposed classifier achieves an average recognition accuracy of 88.7% on MSRC-12 dataset and 90.06% on their own dataset. In [77], by transforming motions into various kinds of Boolean time-varying geometric features describing the relationship between specified body points of a pose, the authors show low dimensional features can be effectively used in matching and retrieving indexed motion-capture streams. However, defining discriminative features and relationships for human motions still stay challenging. In that sense, our work explores the potential of feature selection techniques in identifying discriminative kinematic feature sets for action recognition.

Designing a system for treatment of mentally or physically disordered patients always have been a challenging task. The system not only should have robust technological capabilities, but also it should satisfy the medical criteria.

Recently due to high price of these treatments and either high demand and shortage of rehabilitation specialists, distant commuting routes in metropolitans and long waiting periods, there is an effort in this field to develop efficient, low cost and home-based systems. [78] Presents a virtual reality system for rehabilitation of phantom limb pain. Phantom Limb Pain is a very widespread condition between patients after loss of an arm or leg. They experience a chronic pain and displeasing sensory problems. Studies show that a virtual model of missing limb in computer graphics could help reduce the pain in patient. Pettifer et al. develop this system by using Kinect for tracking limb’s motion in conjunction with wearable sensors to offer the patient this immersive experience and they promise for good results. [7] at University of Minnesota have designed a multi-sensor set-up to look for behavioral disorders. The system could help to early diagnosis on one of very tricky disorders in children, autism spectrum disorder (ASD) while using five Microsoft Kinect depth-sensing cameras along with underlying vision algorithms. The cameras keep track of each child based on shape, color or texture of clothes and store information about each one's activity and how they move their limbs. Then the system could flag up which one of the children are hyperactive and which one is still and candidate them for possible autism disorder.

In [79] they provide the user with full-body control of an animated virtual character. The patient pursuits an in-game goal while controlling a virtual character inside a mine and gathers gems and puts them into a cart. They use Kinect depth sensing camera which gives them full-body markerless tracking of the user's limbs. With features they extract from this resource they developed an interactive game-based rehabilitation tool for balance training of adults with neurological injury. The tool gives patients the ability of experience and record their performance and   clinicians to evaluate these meaningful data. The group tested the tool on the subjects suffering from variety of disabilities and report promising feedbacks from the participants. Also [80] performed a pilot study to assess the possibility of rehabilitating of two young adults with motor impairments (one diagnosed as having acquired cerebral palsy and the other muscle atrophy) using a kinect-based system called "Kinerehab" in a public school. Due to their statistic results the number of correct movements of the patients was significantly improved during the intervention phase compared to baseline movements. Also some orthopedic institutes such as Wardell Orthopedics' Harbour Rehab [81] sing kinect and wiiFit games for physical rehabilitation including balance training and fall prevention. They report a good progress in patients suffering from balance deficits and gait abnormalities by participate them in an engaging, fun and motivational activities.

Chapter 3

Methods

3.1 Methodological Pipeline

In this section we provide an algorithmic overview of our proposed motion analysis methods. Here the aim is to break down the algorithm into logical blocks to clarify the process. The pictorial visualization of the pipeline and its building blocks is provided in Figure 3.1. Given an input data sequence containing an action, these steps are involved in processing, evaluation, classification and assessment of the input information.

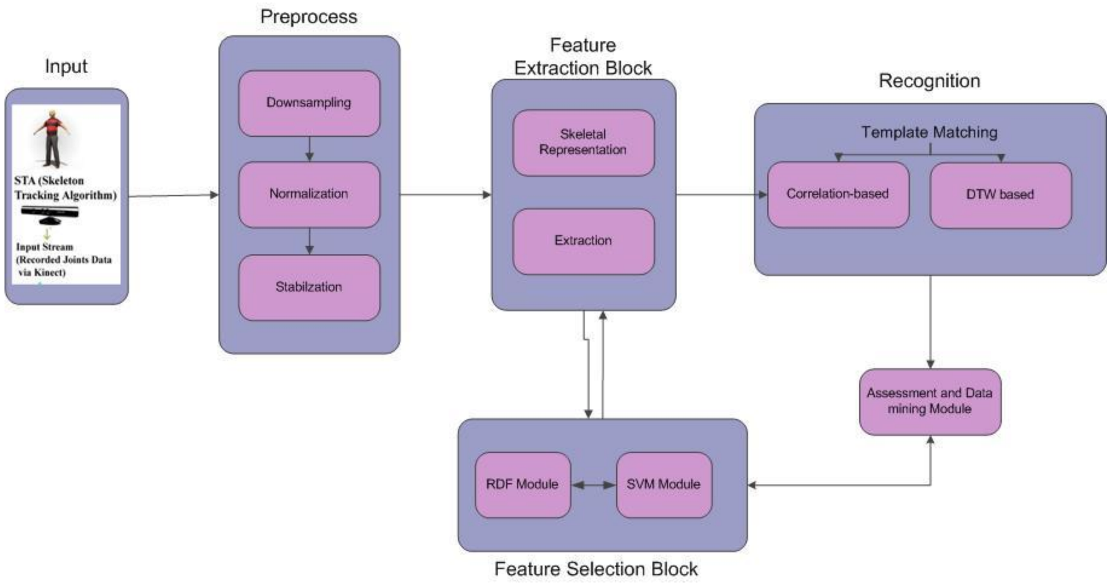


Figure 3.1: Methodological Pipeline

The **preprocessing** is the preparation step which consists of downsampling, normalization and stabilizing of the input data sequence. **Feature extraction** is calculation of defined kinematic features from consecutive pairs of frames of skeletal representation. Classification of the action instances using different template matching methods and analysis of the results is done in **recognition and assessment** block. We claim that the action recognition can be done using a small portion of the extracted feature vectors. In **feature selection** block a feature reduction process by a synergetic method incorporating support vector machine and random decision forests is executed.

3.1.1 Preprocessing Block

To prepare the input data for assessment phase we conduct into some preprocessing steps. Analyzing the data without preprocessing can lead to misleading results. Noisy and unreliable information make the data mining process more difficult. The representation and quality of the data is the first and foremost before running the analysis [82].

3.1.1.1 Downsampling

Downsampling of a signal is the process of reducing the sample rate of the signal. If a reduction in size of the data or data rate is necessary, downsampling is applied on the input signals. In case of our feature time-series which we can treat them as signals, there is different number of frames (time point) in each action instance, some have 60 frame or time points some other has 65, so we downsample and interpolate the time-series to obtain signals with a constant length (Figure 3.2). Doing this, we are also doing an alignment in a very crude manner, later we will do dynamic time warping (DTW) to compensate the misalignment effect.

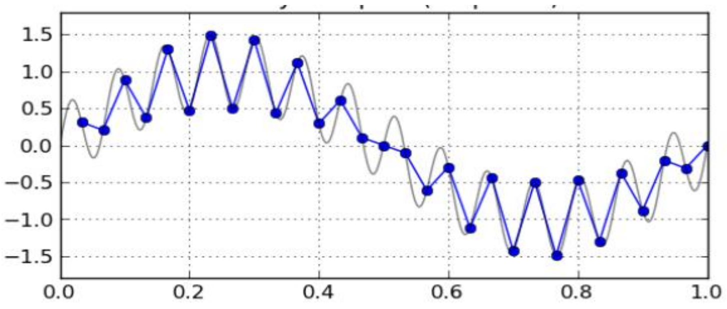


Figure 3.2: Downsampling the features signal to have constant length

3.1.1.2 Normalization

Normalization is a process that follows a procedure to make the data to become closer to the needs of the algorithm. It is a preprocessing procedure which makes the algorithm’s responsibility to be easier. Normalization in the simplest form is the adjustment of values measured in different scales into a common scale. In our pipeline we use this simple normalization method to scale our signal's range to fluctuate between interval [0,1].

3.1.1.3 Stabilization

The data stream coming from the depth sensor suffers from an unavoidable noise in form of jitter. The signal's curve is not smooth and this issue in some point affects the performance of the classification algorithms. To overcome this problem, before all of the processes in the pipeline, skeletal joint coordinate position signals is smoothen by a Gaussian filter (Figure 3.3).

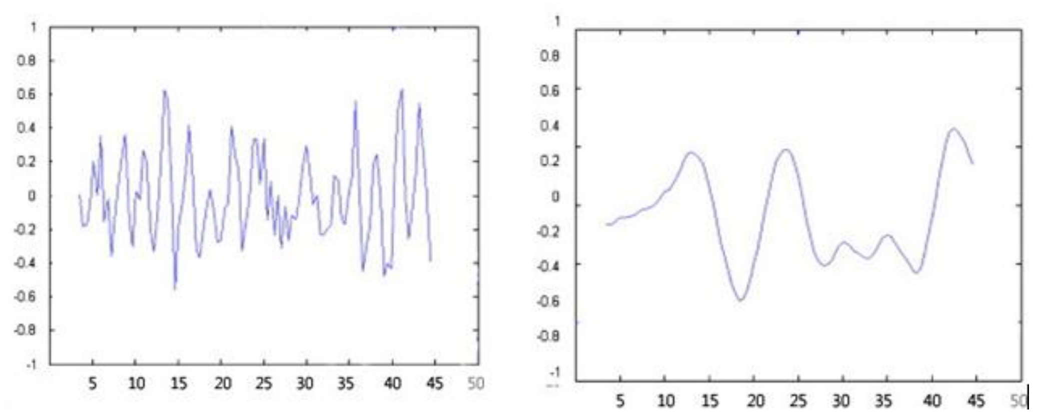


Figure 3.3: Appling Gaussian filter on input signals to overcome the intrinsic jitter

3.1.2 Feature Extraction Block

A good recognition system other than quality of its detection algorithm is one that has a set of features which can satisfy some crucial requirements. We would like to have features that are invariant to the position and the orientation of the sensor. In our case, to make them invariant to Euclidian motion and orientation, we are making them dependent to the torso frame of the skeleton. This torso frame merely moves so that we can treat it as a rigid body. Most of the features we use and test are dependent to this frame which makes them invariant to the position and orientation of the depth sensor. The other beneficiary property that a reliable feature should have is its stability. We want our feature to be stabilized and we want to get rid of noisy fluctuations which are intrinsic to sensory data. As we said before to achieve this we smooth the coordinate position time-series by applying a Gaussian filter prior to the whole process of feature extraction in pipeline. In addition we prefer to have features that they are invariant to intrapersonal and interpersonal variability. Actions and gestures are very variable. They are variable from one person to another person and also from one example to another from the same person (anthropometric variability). invariance of this kind cannot be guaranteed only by feature extraction. Higher-level information gleaned from the classifiers should come into play.

In this phase of the pipeline we try to extract the features from the represented skeleton which are capable to satisfy most of these properties. To this end, we reviewed the literature and test different set of features that are adaptable to our pipeline. We used expert knowledge, reported results on various datasets and our preliminary tests to select the best feature set to continue with.

3.1.3 Recognition and Assessment Blocks

After feature extraction from our skeletal representation of the human body, these features are fed into the recognition block in the pipeline. In recognition block we use two template matching method to classify the action instances. Normalized correlation coefficient (NCC) is a scoring method that calculates the similarity between pairwise time-series. The other similarity metric we use is dynamic time warping (DTW) where it is also capable of rectify the misalignment effect between the time-series. These two methods are employed to acquire a baseline classification performance for our system which we later in feature selection block will improve upon. We also evaluate the classification results by data mining to figure out the properties of the selected features and the classifier's mechanism.

3.1.4 Feature Selection Block

In this block our aim is to maintain the classification accuracy obtained by template matching methods while we reduce the number of features used for classification. We want to select the features and time points that have the most effect on recognition accuracy. With a synergetic method incorporating the SVM and RDF classifiers, we select the discriminative features from the feature set and with these features we try to outperform or maintain the recognition accuracy.

3.2 Methods

Microsoft Kinect SDK, provides markerless full-body tracking by extracting 20 joints of the user's body at 30fps and establishes a gliding wireframe skeleton. Using kinematic features extracted from such a powerful representation, statistical learning algorithms can interpret the user's gestures in order to control the interaction [1, 13, 19].

In template matching we use normalized correlation and dynamic time warping as our distance metric. Then, we classify the action instances in the dataset to find baseline performance. In feature selection, our aim is to find the most discriminative subset of kinematic features that represent an action, which is understood here as a sequence of movements generated by a human agent during the performance of a task. Representation of the human motion via a moving skeleton plays a crucial role in the overall action recognition pipeline. Even though not so much as with conventional video, skeletal data obtained from the processing of raw data acquired from a depth camera are still prone to uncertainty due to anthropometrical differences across users. In addition, geometrical invariance issues, arising due to camera-user position variability are also present. We employ translation and rotation-invariant features to overcome the Euclidean-part of the invariance problem (Section 3.2.1). To alleviate user variability due to inherent action-performance differences and minor scaling, we rely on discriminatively selected features.

To find the most effective and efficient subset of features for a given set of actions from a high-dimensional spatiotemporal feature space, we first discriminatively optimize a random decision forest (RDF) [83] model over a forest-specific set of hyper parameters and then, we collect all the unique features from the nodes of each tree in the optimal forest. We feed this selected feature set into a linear support vector machine training procedure to learn the final classifier.

We test our classifier on two datasets acquired using Microsoft Kinect platform. The first one is the Microsoft Research Cambridge dataset (MSRC-12) [19], where our classifier attains an average accuracy of 94% on MSRC-12. We additionally test the classifier on the WorkoutSU-10 dataset [84] collected in our laboratory, Sabancı University VPALAB. WorkoutSU-10 contains 10 exercise gesture classes, selected by professional sport trainers for therapeutic purposes. Our classifier reaches an average accuracy of 98% on this dataset.

In this work first we use a large set of invariant spatiotemporal features extracted from skeletons in motion and we evaluate them on presented datasets and obtain a baseline classification result and next, we introduce a discriminative RDF-based feature selection framework capable of reaching impressive action recognition performance when combined with a linear SVM classifier.

3.2.1 Kinematic Feature Extraction

For a faithful representation of the skeleton in motion leading to successful recognition, instead of comparing 3D joints of the skeleton in space and time by matching their time-varying 3D coordinates, we extract relational features of joints in a pose at each time point. The tracking algorithm in the current version of Microsoft Kinect platform can effectively track 20 joints of the active person at up to 2 meters distance from the sensor within the field of view. The center of coordinate system coincides with the position of the sensor. Human motion is enabled by the collective movement of the skeletal joints in space through time. In order to characterize human motion, we calculate the so-called motion or kinematic features, using the joints one by one or a subset of them, during the whole course of the action performed. Given a pose at a given time point, several geometric features can be computed as described in the sequel. The collection of these features computed at each of the 20 joints at each time point during the whole course of the performed action will be data-mined by RDF model optimization and selected features will be fed into SVM training.

As we mentioned before, a good set of kinematic features in our context should satisfy at least some of the following requirements:

- **Invariance to the position and orientation of the sensor.** The features introduced in the sequel are all invariant to Euclidean motion, most of them being also scale-invariant.
- **Stability.** In order to ensure stability against unavoidable noise in the form of jitter to some extent, we smooth the skeletal joint coordinate position time-series by a Gaussian filter prior to the whole feature extraction process.
- **Invariance to intra- and interpersonal variability.** In order to get a more accurate classification, the system should not be so much dependent on the way an action is performed, considering that even the same person’s gesture features change from an instance of the same action to another instance. While informed geometric design can cope with such kind of variability to some extent, intra- and interpersonal invariance cannot be guaranteed by feature extraction and higher-level information gleaned from statistical classifiers should come into play. This is addressed by RDF-based feature selection and SVM training.

With these ideas in mind**,** we first define a torso frame, which consists of seven joints, and apply a PCA on constructed matrix of torso joint coordinate positions [13]. These joints seldom move independently, as such they are instrumental in constructing a canonical coordinate frame for several features described in the sequel. The torso frame is established by the following joints: neck, spine, right shoulder, left shoulder, right hip, and left hip (also indicated in blue in Figure 3.4). The first two basis vectors found by PCA ($\mathbf{u}$ *and* $\mathbf{r}$) and their cross product $\mathbf{t}$ form the torso basis vector.

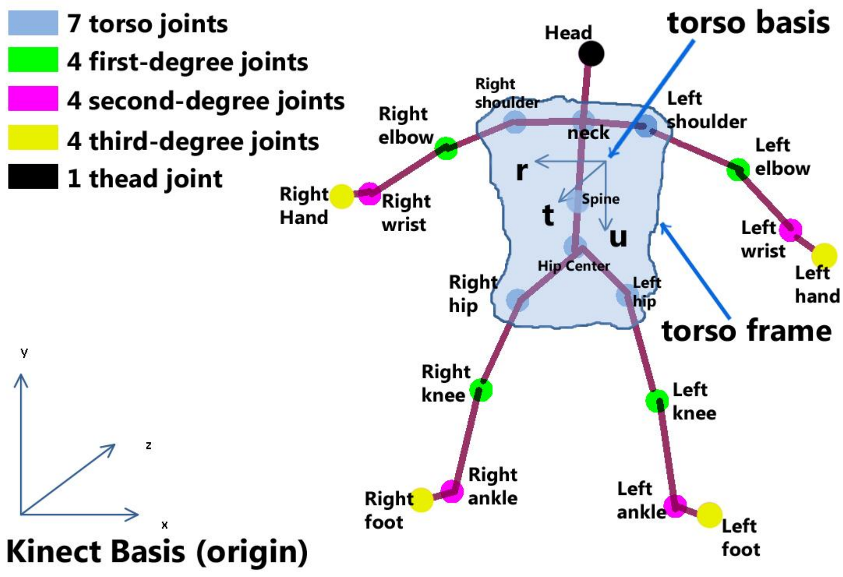


Figure 3.4: skeleton representation

Let furthermore the 20 joints of the skeleton be indexed by the set $J = \{0, \dots, 19\}$. We adopt the following partitioning of the joint index set J as $J = \{Head\} \cup J_0 \cup J_1 \cup J_2 \cup J_3$. The set J_0 indexes the seven torso joints shown in the blue shaded area of Fig. 16. The set J_1 indexes the four first-degree joints – the ones that are immediately connected to a torso joint as shown in green in Figure 3.4. The set J_2 indexes the four second-degree joints – the peripheral joints that are not of first-degree type and shown in pink in the Figure. The set J_3 indexes the four third-degree joints – i.e., the remaining joints shown in yellow in the Figure. This representation is similar to one that is proposed in [13, 85]. Equipped with this notation, we can now define three different types of features as follows:

Type-I features. We define eight pairs of azimuth and elevation (θ^j, φ^j) angles for each joint $j \in J_1 \cup J_2$ with respect to the torso frame (Figure 3.5) to render them rotation-invariant at each time point. Accordingly, there are 16 angular features in total.

Type-II features. Let $p^j \in \mathbb{R}^3$ be the 3D position of the joint j. The interjoint distance $d^{i,j}$ is then defined as the Euclidian distance between joints i and j. We will have $\binom{20}{2} = 190$ such distance features in total.

Type-III features. We also define the three coordinate components of the 3D velocity vector of an individual joint with respect to the torso frame as a new type of feature. There are 20×3=60 velocity features in total.

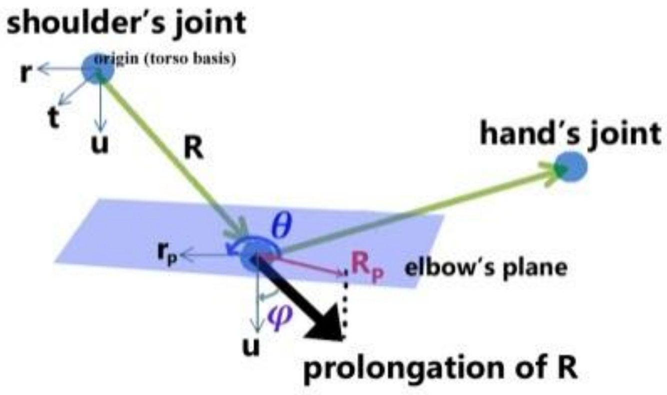


Figure 3.5: Joint representation

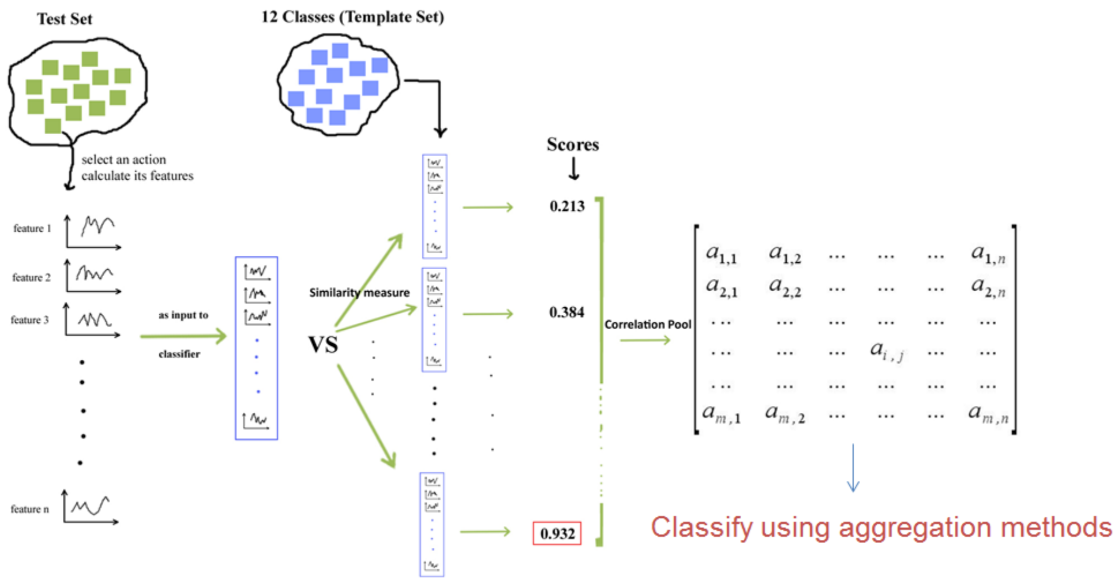
Notice that since all features recalculated during the whole course of the motion, each quantity above forms a time-series. Once the pool of features is constructed, we can denote the set of feature time-series obtained from the skeleton in the course of its motion generically as $f^{(k)} = \{ f^{(k)}(t_1), \dots, f^{(k)}(t_N) \}$, where $k = 1, \dots, K$ runs over the set of features (K=266) and N is the total number of time points. There will be KN feature values in our collection.

3.2.2 Recognition and Assessment

General aim for pattern recognition algorithms is to estimate a sensible answer label for input data instances and to carry out a matching process between the inputs, by considering their statistical variations. For action recognition from input time-series we use two approaches. To obtain a baseline performance we use template matching and then, to improve the recognition results we use our synergetic approach where it consists of a RDF-based classifier in conjunction with a SVM classifier. In this section, first we describe our template matching method and next, our classifier-based recognition methods.

3.2.2.1 Template Matching

A template is a pattern that is used for produce items of that same type. Template matching is comparison of incoming instances with templates stored in long term memory. Figure 3.6 shows the data flow of our template matching algorithm.



Test Set
12 Classes (Template Set)
select an action
calculate its features
feature 1
feature 2
feature 3
feature n
as input to
classifier
Similarity measure
VS
Scores
0.213
0.384
0.932
Correlation Pool
$a_{1,1}$ $a_{1,2}$ ... ... $a_{1,n}$
$a_{2,1}$ $a_{2,2}$ ... ... $a_{2,n}$
$a_{i,j}$
$a_{m,1}$ $a_{m,2}$ ... ... $a_{m,n}$
Classify using aggregation methods

Figure 3.6: Data flow in template matching algorithm

After feature extraction process, similarity score of an action is calculated using correlation coefficient or dynamic time warping. These scores are calculated by comparison against actions in template set. All of the similarity scores are stored in a matrix called correlation pool where it contains all pairwise similarity scores of all actions in the dataset. Using aggregation matching strategies, the system classifies the actions that their scores are available in the pool.

3.2.2.1.1 Correlation-Based

Given a description for each action instance, we can compare two action instances U and V and by computing the similarity between their respective feature sets $F = \{f^{(k)}\}$ and $G = \{g^{(k)}\}$. A natural basic procedure to do that is as follows:

— First, compute the normalized correlation coefficient (ncc) between each corresponding pair of features $f^{(k)}$ and $g^{(k)}$ in F and G respectively, that is, $ncc^{(k)} = ncc(f^{(k)},\ g^{(k)})$. We have $K = 266\}$ such correlation values.

— Then, aggregate the K correlation values to obtain a unique similarity or agreement value between sets $F$and G (hence action instances and. There can be several options for aggregation, such as average, median,, maximum operators. At this "feature" level fusion stage, we prefer the Average operator in our tests, that is, our similarity function will be as follows:

$$SIM(F,G) = Average(\ ncc^{(k)})$$

Now that we have a method to compute the similarity between two action instances and also a dataset S of labeled action instances, we want to estimate the class of a test action instance V. We assume that in our dataset S, there are $|S|$ labeled instances U^s,each of which belongs to one of the $|C|$ categories denoted by the set C. We further assume that the computational descriptions $F^s$of each such action instances are computed and stored in a pool. Then the basic classification procedure is as follows:

— Compute the feature description of the test instance V, denote this feature time-series set as G.

— Compute the pairwise similarities between G and each of the F^sin the dataset: $SIM(F^1, G), SIM(F^2, G), \ldots, SIM\left(F^{|S|}, G\right)$ and sort the list of similarities in decreasing order,

— Now determine the label of the test instance$V$ based on the labels of the databases instances in the front of the list. For this also, there can be several possible options:

- **K-NN:** Assign the majority label amongst the K-first labels in the ranked list.
- **Classwise score average:** Average the scores separately for each class then assign the label of the class having the maximum average.
- **Classwise score product:** Compute the score product for each class then assign the label of the class having the maximum score product.
- **K-first versions of classwise average and product**.
- **Borda Count.:** Sort all of the scores, then start from the least score and assign 1 point to it and continue until assign N point to the highest score. Sum

up the points for each class and assign the label of the class with the maximum gained points.

3.2.2.1.2 DTW-Based

Sometimes the actions perform in different time frames. For example a recorded template action starts immediately in time t = 0 but the tests action that is performed by the subject starts after some delay in t= 2. In this case we wouldn't have an accurate comparison between source and destination time-series if we use correlation measure.

Another problem we are confronted here is when the test and template actions are performed with different velocities. For example it is possible that the recorded template action is performed faster than the test action. That is maybe the action is done accurately but faster or slower than the reference action. On the other hand the time-series are stretched or shrink and again we won't have a fair comparison between the two series.

There are several approaches in the literature to tackle these problems. At an early stage, some linear normalization techniques were examined, in which timing differences between speech patterns were eliminated by linear transformation of the time axis (Euclidian, Mahalanobis and Manhattan distance measure approaches). Studies indicated that any linear transformation is inherently insufficient for dealing with highly complicated fluctuation nonlinearity. Other methods introduced specially for speech recognition field like Spectrogram Cross Correlation (SCC) which basically involves multiplying the intensity values in spectrographs together, and summing the total. If two signals are very similar, then they should overlap considerably in the spectrograph, and this will be reflected in a high SCC value. But it has been realized for some time that this algorithm did not always accurately reflect differences between signals: slight but consistent deviations in frequency, for example, could reduce the SCC score to 0. The algorithm doesn't react in a consistent fashion to deviations in similarity, and is sensitive to background noise.

"DTW based classification aligns sequential data in a way that preserves potentially existing internal structures of the data, which is beneficial for the analysis of real-world time-series" [86]. DTW algorithm is extremely efficient as the time-series similarity measure which minimizes the effects of shifting and distortion in time in order to detect similar shapes with different phases (Figure 3.7 top). Given two time series $X =$

$(x_1, x_2, \ldots, x_N)$ and $Y = (y_1, y_2, \ldots, y_M)$ DTW yields optimal solution in the O(MN) time, the data sequences can be sampled at equidistant points in time or they can be of different time point samples.

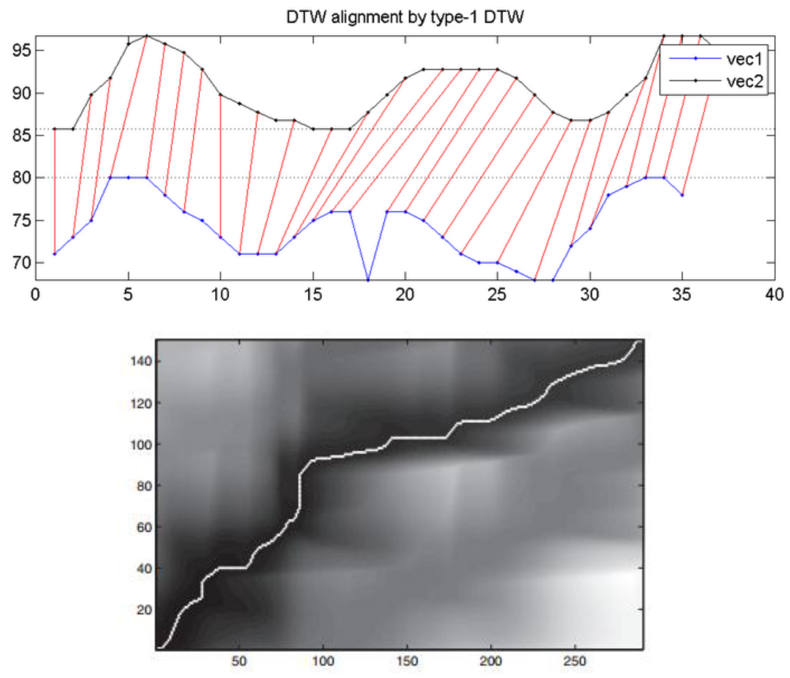


Figure 3.7: Top: Alignment of time-series using DTW. Aligned points are indicated by the red lines, Down: Accumulated cost matrix with optimal warping path.

Sequences are taking values from feature space Φ, we need a function to measure their distance between the two time-series: $d: \Phi \times \Phi \rightarrow R \geq 0$. d have small value if sequences are similar and large value if they are different. Because of dynamic programming nature of DTW this Distance Function is called **Cost Function.**

Using this cost function that can be simple Euclidian distance measure or other measures a matrix called local cost matrix is created which represents all pairwise distances.

$$c_p(X, Y) = \sum_{l=1}^{L} c(x_{nl}, y_{ml})$$

c_p calculates costs of all the paths (p) from X to Y. The optimal warping path (p^*) with minimal cost among all the paths between X and Y is found by DTW using the constructed cost matrix:

$$DTW(X, Y) = c_{p^*}(X, Y)$$

Where c_{p^*} is a path in c_p that has the minimum cost.

To find the optimal path p^* one can test costs of all of the paths in the cost matrix and select the path with minimum cost but this is not computationally efficient and the complexity can grow exponentially with growth of series lengths. To cope with this problem a dynamic programming approach [87] was used which its cost is in order of $O(MN)$ where the M and N is the lengths of the time-series. To this end, we should define the prefix sequences as $X(1:n) = (x_1 \ldots x_n) for\ n \in [1:N]$ and $Y(1:m) = (y_1 \ldots y_m) for\ m \in [1:M]$ and using these we can define accumulated cost matrix D as:

$$D(n,m) = DTW(X(1:n), Y(1:m))$$

Where D defines a $N \times M$ matrix. To compute D matrix we use the following procedure: for the last row and the first column of the accumulated matrix we just put the corresponding row and column from cost matrix c. for the other cells we use this procedure:

$$D(n,m) = \min\{D(n-1,m-1), D(n-1,m), D(n,m-1)\} + c(x_n, y_m)$$

Now using the accumulated cost matrix we can compute the optimal warping path (Figure 3.7 Down). The optimal path $p^* = (p_1, \ldots p_L)$ is computed by reverse ordering (backtracking) of the indices starting from $p_L = (N, M)$. Suppose that we computed the $p_l = (n,m)$ if (n,m) = (1,1) we should have $l = 1$ and the procedure is finished. Otherwise:

$$p_{l-1} = \begin{cases} (1, m-1) & if\ n = 1 \\ (n-1, 1) & if\ m = 1 \\ argmin\{D(n-1,m-1), D(n-1,m), D(n,m-1)\} & otherwise \end{cases}$$

we used a modified version of classic DTW proposed by Sakoe and Chiba [87] in over evaluations.

We use DTW as similarity measure in our work to compensate the misalignment effect that may affect the classification performance. But later we will see from the results that DTW doesn’t outperform the accuracy too much. The reason may lies in the fact that we record our actions in a very controlled environment where we want the subjects to start and stop their action performances in certain moments. But we should take it into account that this is not the case in real world occasions and this approach will be effective in application development.

Again, with DTW we create the score pool of similarity values and apply the same aggregation strategies described in correlation section to classify the actions.

3.2.2.2 Classifier-Based Approaches

For action recognition we make benefit from some of well-known classifiers. We use random decision forests and support vector machines in our classification method. In this section we first, describe these two classifiers and next, we describe how we employed them and proposed a synergetic method for feature selection.

3.2.2.2.1 Random Decision Forests

In this work we utilize Random Decision Forests [83] when we train our data. To get familiar with this classifier we bring here a brief introduction of random decision forests and their application in machine learning analysis tasks.

A random decision forest (RDF) is an ensemble of randomly trained decision trees. For the first time it has been used for handwriting recognition by [88] via randomize feature selection. In later works it showed high-grade generalization quality over other techniques such as boosting.

Decision trees are used for making decisions. They consist of internal (split) nodes and terminal (leaf) nodes. Decision trees can interpret complex problems into hierarchy of simpler ones. Function of decision trees can be separated into two phase: off-line (or training) and on-line (or testing). In **testing phase,** given an unseen data point **v,** on each node of the decision tree starting from the root a *split function* applies to **v** and on the result of this binary test the data led into right or left split. There is a *predictor function* at each leaf which associates an output for the input data points. In **training phase** optimization of parameters for split functions and predictors take place. In case of a forest with **T** trees, the training process is repeated for separately for each tree.

Entropy and Information Gain: Given a training set of data points and their labels we try to choose tree parameters to minimize an energy (entropy) function. As shown in Figure 3.8 (a) distribution of discrete data points in 2D from four different classes is uniform because we have equal number of points in each group. If we split the data horizontally or vertically (Figure 3.8(b) and 3.8(c)) we will end up with lower entropy on each side. The information gain by this split can is computed as:

$$I = H(S) - \sum_{i\,\in\{1,2\}} \frac{|S^i|}{S} H(S^i)$$

Where S is our initial set. In this equation the mathematical entropy defined as:

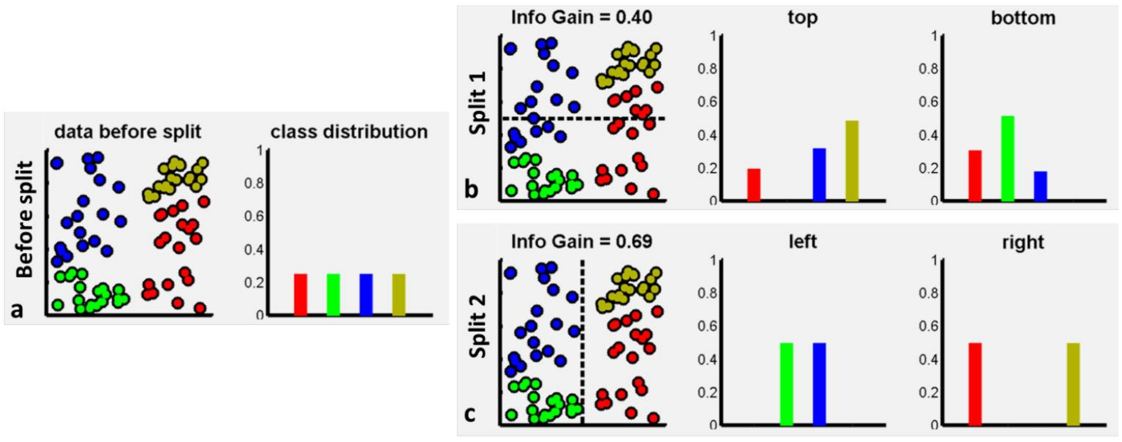


Figure 3.8: Information gain for a discrete dataset: (a) before split (b) after horizontal split and (c) is after vertical split Source: [17]

$$H(S) = -\sum_{c\,\in C} p(c)\log(p(c))$$

In Figure 3.8(b) the horizontal split does not split the data appropriately, and it yields information gain of approximately 0.4. When we use vertical split of Figure 3.8(c) its result is lower entropy of resulted sets and as a consequence higher information gain of 0.69. This simple example is the base concept of forest training algorithm.

Weak learner and Training Objective Function: A random decision forest model is characterized by number of elements. We should choose the split function (or weak learner) for internal nodes and also predictor functions for leaves.

A **split function** for node j is defined as $h(v, \theta_j) \in \{0,1\}$, v is data points arriving at each node and $\theta_j$is parameters of the weak learner in node j where we define it as: $\theta = (\phi, \psi, \tau)$, ϕ is randomly selected features out of entire feature vector v, ψ is the geometric primitive used for split the data and τ the threshold used in binary test of the split function. Data separation function can be defined as linear or non-linear. For example in 2D case the weak learner can be a generic line or axis-aligned line. Weak learner can be complex and non-linear e.g. conic surfaces.

We need to find optimal parameters for split functions at each node. To achieve this we need to optimize the **objective function.** This is done by maximizing the information gain at each node:

$$\theta^* = \arg\max_{\theta_j} I_j$$

Randomness in Forest Trees: One of the important characteristics of forests is that all of the trees in the forest are randomly different from each other. This randomness can be injected into the model during training. two very popular ways to do this is i) bagging [83] and ii) randomize node optimization [89]. While bagging gives us training efficiency, the node optimization technique enables us to margin-maximization property and enables each tree to use the totality of training data. Notice that these two techniques are not mutually exclusive and can be combined together in a fusion implementation.

The Ensemble Model and Stopping Criteria: Information each tree needs to use in testing step is leaned during training. In classification step each leaf contains a probability distribution over the classes associated with the subset of the data has arrived that leaf. The probabilistic predictor for t^{th} tree then is $p_t(c|v)$ where c indexing the class. In a forest with T trees all trees are trained independently and in testing phase the data point v drive into all trees until it reaches to corresponding leaves. Combining all the trees into a single forest can be done by a simple average over the trees [83]:

$$p(c|v) = \frac{1}{T}\sum_{t=1}^{T} p_t(c|v).$$

Or with multiplying tree outputs together:

$$p(c|v) = \frac{1}{Z}\prod_{t=1}^{T} p_t(c|v).$$

Where Z is the partitioning function ensuring the probabilistic normalization.

There are different methods to stop individual trees from growing. One common method is to stop when a maximum depth of tree D has been reached. The other method is to stop when a minimum information gain is evaluated. One also can stop a tree from growing when less than a defined number of data points reach into a node. Avoiding growing full trees has been demonstrated to have positive effects in terms of generalization [17].

3.2.2.2.2 Support Vector Machine

Support vector machines (SVMs) are a class of applications that can be used for classification, regression, density estimation and other applications. Two-class classification is the simplest form of SVMs where, its algorithm finds a hyperplane between the two classes with the maximum margin as possible. This results in good generalization accuracy on unseen data. SVM supports specialized optimization methods that help SVMs to learn from a large amount of data. Over the past decade maximum margin models become popular in machine learning. These techniques were developed in three major steps in last 50 years. First, assuming that two class of training examples can be separated by a hyperplane and this plane separate the training example with maximum margin. Second, the kernel functions get incorporated in maximum margin models and their formulation become more resemble to current form of SVMs. In tests, SVMs shows significant empirical performance compared to neural networks by incorporating transform invariances. SVMs has very strong mathematical basis and they are related to famed statistical theories. They not only do the classification correctly but also they maximize the margin for better generalization. This approach results a hyperplane that only separate the data points laid on the margin. These points called support vectors and because of them the classifier called support vector machine. Since real-world data analysis problems involve nonlinear dependencies, SVMs can easily extend to support non linearity thanks to kernel functions. SVMs obtain state of the art performance in accuracy, robustness and efficiency which they are used abundantly in many applications including computer vision, bioinformatics and finance.

In general case when we have a linearly separable example and we want to find the optimal hyperplane, consider that we have a the training set as $\{(x_i, y_i)\}$ where $i \in \{1, \dots n\}$ and x_i is the feature vector for the i-th input example and $y_i \in \{1, -1\}(positive\ or\ negative)$ is the label of the corresponding example. By assuming that the set of examples are linearly separable into positive and negative value we have a function f that $f(x) = \langle w, x \rangle + b$ where w is weight vector and b is the bias where,

$$\langle w, x_i \rangle + b > 0 \quad for\ y_i = +1$$
$$\langle w, x_i \rangle + b < 0 \quad for\ y_i = -1$$

There can be different separating hyperplanes but they are not equal. There is a concept called margin that is the distance between the hyperplane and the closest example in the training set. The goal of the SVM is to find a hyperplane that its margin is maximized (Figure 3.9).

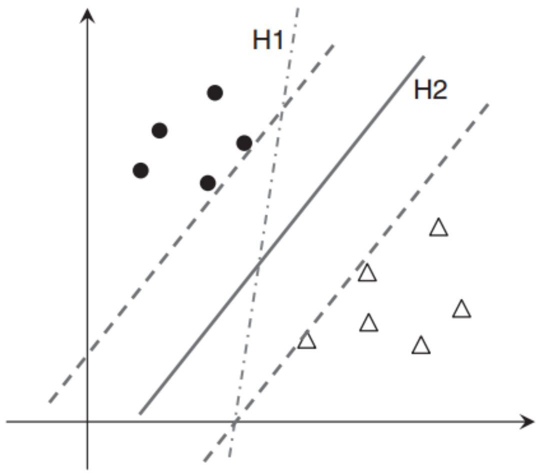


Figure 3.9: Both H1 and H2 correctly separate the set but H2 has wider margin than H1 [90].

Therefore the solution to find this maximum value is an optimization problem that seeks optimal w, b:

$$maximize\ w, b \min \frac{|\langle w, x_i \rangle + b|}{\|w\|}$$

This is a linearly constrained quadratic program which can be solved efficiently and it's the primal form of SVM for linearly separable case. In real life most of the datasets are not linearly separable and therefore no $w\ and\ b$ can satisfy the constraints of the optimization problem. In this case we should allow some data to violate the margin condition and for that we penalize it and use soft margins. In this way we define a slack variable which is called Hinge loss and using this variable we can reformulate the optimization problem into a non-smooth problem. In addition, SVM can be used for non-linear classification to separate data that are not linearly separable by using kernelization, where it use a function (kernel function) to transforms the original data and map them into new feature space (Figure 3.10).

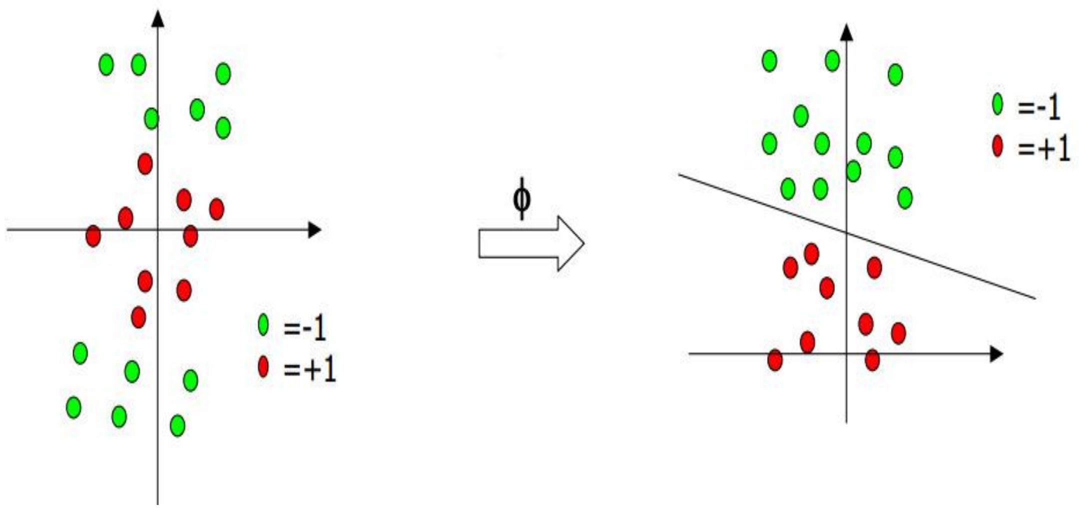


Figure 3.10: Kernel trick for nonlinear classification

SVMs have been widely used in real-world problems. Its first empirical success was when people used it in handwritten digit recognition. It has also been very successful in computer vision applications such as object recognition and detection. With the special advantage in handling high dimensional data, SVMs have witnessed wide application in bioinformatics such as microarray processing, and natural language processing [90]. In this work we use linear SVM in conjunction with RDFs and we introduce a new feature selection mechanism.

3.2.2.3 Synergistic Use of SVM and RDF for Feature Selection and Recognition

Our framework proposes a synergetic use of SVMs and RDFs. We have a pipeline where its input is the whole set of spatiotemporal features. Inside its modules first using different configurations of the forests we find the optimized forest with highest accuracy and least number of features. Then we obtain unique features of the optimal forest and train a SVM by them and find its optimal parameters. We show that this procedure perform the recognition with higher accuracy in some cases and with less number of features (almost one tenth of the initial feature set) (Figure 3.11).

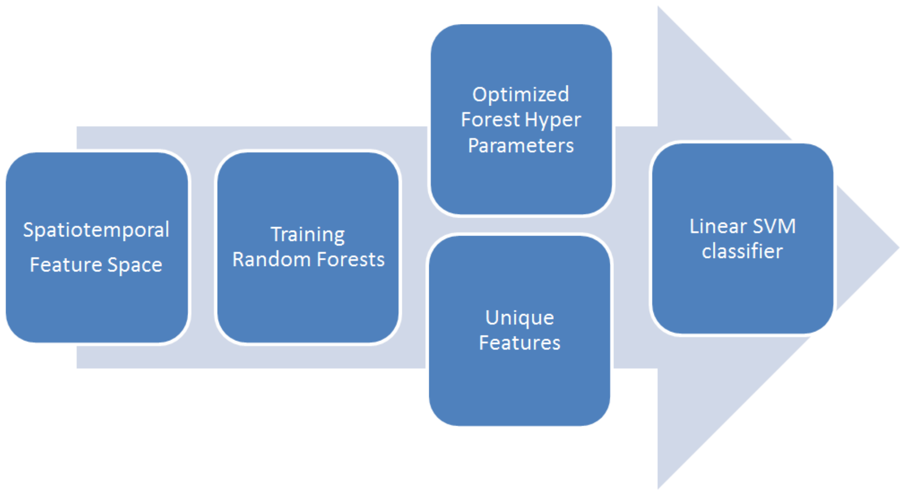

Spatiotemporal Feature Space
Training Random Forests
Optimized Forest Hyper Parameters
Unique Features
Linear SVM classifier

Figure 3.11: Feature selection framework depict by its building blocks

As we mentioned random decision forest (RDF) is a collection of decision trees, where each tree is grown randomly. While a RDF is in general used as a discriminative classifier by itself, in this work we employ it as a discriminative feature selection tool. More specifically, we leverage (i) one of the randomization mechanisms coming into play in RDF training, random node optimization, and (ii) the easily interpretable leaf structure of decision trees as will be described shortly.

There are two means of injecting randomness into the decision forest growing process. The first one is to train each tree on a different random subset of the original training set, in much the same way as in bagging [91]. In the following account, we do not use this mechanism since we did not find any noticeable positive effect as compared to making the whole training set available to each tree in line with the recent arguments in the literature [17]. The second mechanism, that we heavily rely upon, is to optimize each node in each tree over a set of parameters chosen randomly from the whole parameter set as described in the previous section.

In standard decision tree learning, a node is optimized with respect to a purity measure such as information gain [92] or Gini index [93] in order to split the input set of instances into two subsets, each of which is as "pure" as possible in terms of class labels. The split at a node is characterized by the so-called split functions, which, in our context, is instantiated as $s \stackrel{\text{def}}{=} f^{(k)}(t_n)$, where k indexes the feature time-series andt_n the time point. Given a test instance U with description $F = \{f^{(k)}\}$, the action of the split function on U is expressed as follows:

$$\text{Assign } U \text{ to} \begin{cases} \text{the left split if} f^{(k)}(t_n) < \tau, \\ \text{the right split otherwise.} \end{cases}$$

where τ is the decision threshold. As such, the feature index k, the time point t_n and the decision threshold τ are the parameters to be optimized at each node at each tree during forest training. Random node optimization consists of testing only a fraction of these parameters instead of running an exhaustive search over the whole parameter space, which would be computationally prohibitive if performed at each node of each tree in the forest. In our case for instance, there are *K*=266 feature time-series and *N*=50 time points (totaling $K \times N = 266 \times 50 = 13300$ unique features) to test; combined with the number of thresholds to test and the number of all nodes, exhaustive search would be impossible for all practical purposes. That's exactly where random node optimization becomes handy in that not all parameter configurations are tested but only a small subset of them.

A RDF model itself is then parameterized by the number of trees to grow N_{tree}, the number of features to test at each node $N_{feature}$, the maximum depth of each tree D and the number of thresholds to test$N_{threshold}$. After training the forest with a specific configuration$(N_{tree}, N_{feature}, D, N_{threshold})$, one can identify and collect the features selected at each node as the most discriminative ones within the original pool of features. The natural question is then which configuration should be used to grow the forest.

We consider the RDF training model selection problem in conjunction with feature selection as a specific RDF leads to a specific set of features retained. To this end, we follow a discriminative model validation approach. That is, we set a range for the model parameters$(N_{tree}, N_{feature}, D, N_{threshold})$, and then we train a RDF for each configuration on some training set. Then, we evaluate the performance of the RDF classifier for each configuration on a separate set (other than the one used for training). Once we have all validation performances, we choose the forest in view of its validation accuracy and its feature reduction efficiency, which is defined as

$$efficiency = 1 - \frac{K_r}{KN}$$

Where $K_r$is the number of uniquely retained features in the course of training a specific forest and *KN* is the total number of features. As will be seen in chapter 4, this procedure leads to accuracy vs. efficiency curve on top of which we can pinpoint the most effective and efficient feature set amongst tested configurations, i.e., the one which gives “the most bang for the buck”.

After finding the most efficient forest configuration and its hyperparameters, by seeking out the nodes of each tree in the forest, we obtain the features that are uniquely selected in the forest. These unique features then are fed into a linear SVM classifier and the classifier is trained repetitively so that its optimized parameters are found. This optimized classifier then is used for classification of the test set.

Chapter 4

Datasets and Experiments

In this chapter, we first report the results of preliminary assessments we obtained in early stages of the work while we just tested some evaluation methods on single joints of skeleton to test the most simple recognition scenarios. Next, we evaluate our classification algorithm's validity on the datasets we describe in upcoming section. The specification of the system we have used in our tests is as below:

Processor	Intel Xeon CPU X5550 2.67 (two processor)
Memory	32.0 GB of RAM
Coding Software	MATLAB R2010b
Operating System	Windows Vista

Table 4.1: System specification

4.1 Early assessments

In early days of our research we needed to have an intuition about the data stream that kinect sensor provides us with and also with how much accuracy and robustness we can distinguish a joints movement pattern from the others. To figure out this we started with the simplest scenario, simplest actions and feature set. We recorded very simple actions that involve just two or three joints while the other joints are steady (Figure 4.1 left) and then, we evaluate the movement of a selected joint to figure out if we are able to distinguish its movement and with how much accuracy we can do it. If the accuracy is acceptable then, we can move forward and test move complex actions using variety of features. In this section we describe these early assessments on basic action instances we recorded.

In feature extraction phase the primary aim is to find the most representative feature set which is the best possible set to classify our gestures. The main focus in early days was on spatial features of the skeleton as long as the initial information acquired from depth sensor is the Euclidian coordinates of the retrieved joints. This means that by itself, any application using the Euclidian Coordinates as features will differentiate the same gesture performed by two different subjects (intra-personal variety) if they are:

- Positioned at a different location,

- Facing a different angle,
- Having different physical aspects (e.g. height, weight, etc.).

In order to extract features that are going to be invariant to the above occasions, we created a vector skeleton (Figure.4.1 right) and normalized the vector sizes to assure all of them have a Euclidian length of 1 unit. This allows the skeletal joint feature to become invariant to the position of the subject with respect to Kinect Camera and to the physical aspects of the subject.

We then considered every neighbor skeletal vector separately – thus grouping every two consecutive vectors – and formed the normal vector that is created by applying vector multiplication to these two skeletal vectors. We then represented these normal vectors in spherical polar coordinates. This final modification makes our feature set invariant to the possibility of subjects facing different angles. This is due to the fact that every joint is now represented with respect to the neighboring skeletal joints from both sides. In order to wrap up the feature set we extracted within the time of preliminary work, for each normal vector we had:

- Scalar angle α between neighboring vectors (Figure 4.2a)
- Azimuth angle Φ of the normal vector (Figure 4.2b)
- Elevation angle θ of the normal vector (Figure 4.2b)
- Average of these features
- Product of these features

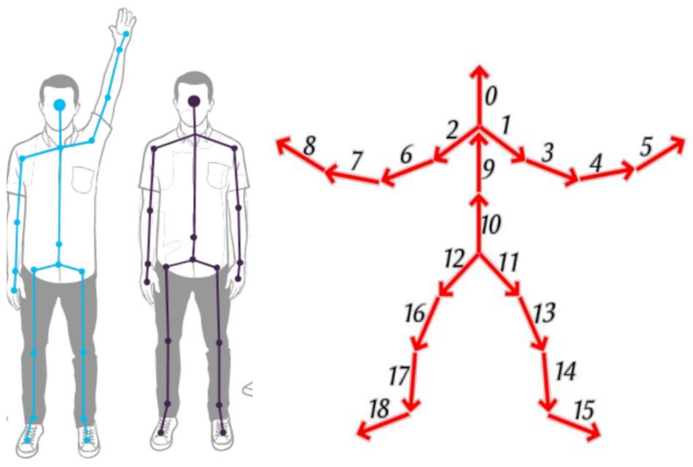


Figure 4.1: Left: An example of basic action Right: Vectors of skeleton

Initially, we didn't have a data to test our algorithms on. Therefore, we recorded a very controlled example data set of our own. This data set consisted of 5 different basic and very simple gestures which only concern two or three joints while the other joints stay steady because we want to test the simplest scenario. We specifically chose a characteristic joint – right elbow joint – for considering the most while observing the performance of our current feature set and the classifier.

As for the classification problem, we computed the correlation coefficients of corresponding features between each instance of gestures and creating a correlation/confusion matrix. Using these correlation matrices, we found the correctness ratio; classify a gesture according to the maximum correlation coefficient bearing value relative to each gesture and ratio of correlation coefficients of each gesture to other gestures with the chosen feature sets. We also computed the entropy of the correlation coefficients of the action blocks – by grouping the instances which belong to the same gesture –.

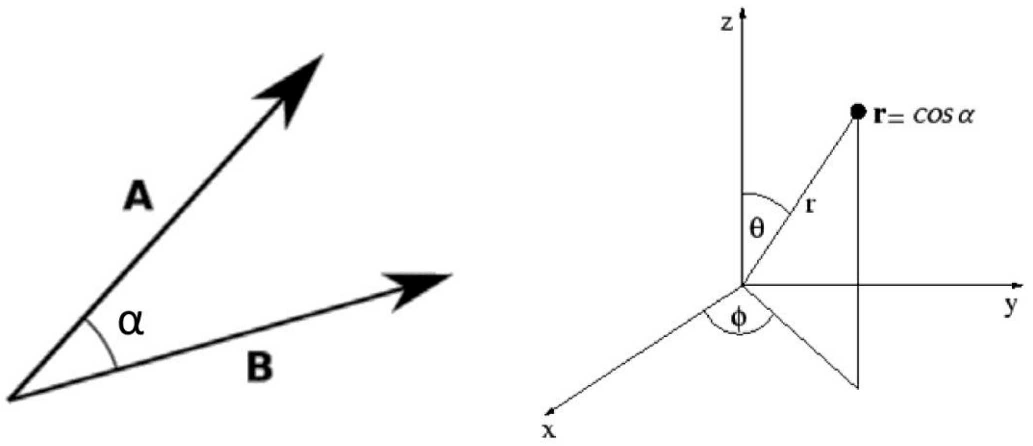


Figure 4.2: (a) Scalar angle between neighboring vectors of vectorized skeleton (b) Azimuth, elevation and magnitude components of the Normal vectors r

We obtained the following results when we calculated the correlation coefficients of the previous 5 features on the right elbow joint.

For scalar angle feature, we've obtained the correlation matrix seen in Figure 4.3 Left. The entropy for scalar angle feature is 10.18. The correctness ratio is 90%. Ratio correlation of each action block is in consecutive order, 0.295, 0.294, 0.438, 0.349, and 0.303. For elevation angle feature, we've obtained the correlation matrix seen in Figure 4.3 Right. The entropy for scalar angle feature is 10.96. The correctness ratio is 98%.

Ratio correlation of each action block is in consecutive order, 0.271, 0.412, 0.283, 0.303, and 0.307. For azimuth angle feature, we've obtained the correlation matrix seen in Figure 4.4 Left. The entropy for scalar angle feature is 11.40. The correctness ratio is 66%. Ratio correlation of each action block is in consecutive order, 0.243, 0.276, 0.282, 0.249, and 0.209. For average of the preceding features, we've obtained the correlation matrix seen in Figure 4.4 Right. The entropy for scalar angle feature is 11.33. The correctness ratio is 98%. Ratio correlation of each action block is in consecutive order, 0.272, 0.315, 0.319, 0.298, and 0.274. For product of the preceding features, we've obtained the correlation matrix seen in Figure 4.5. The entropy for scalar angle feature is 8.26. The correctness ratio is 98%. Ratio correlation of each action block is in consecutive order, 0.469, 0.724, 0.700, 0.574, and 0.508.

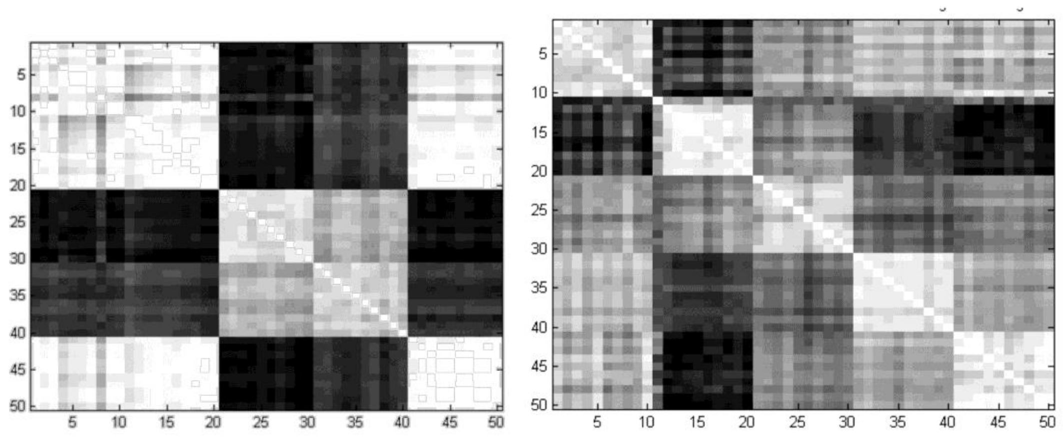


Figure 4.3: Left: Correlation Matrix for Scalar Angle Feature Right: Correlation Matrix for Elevation Angle Feature

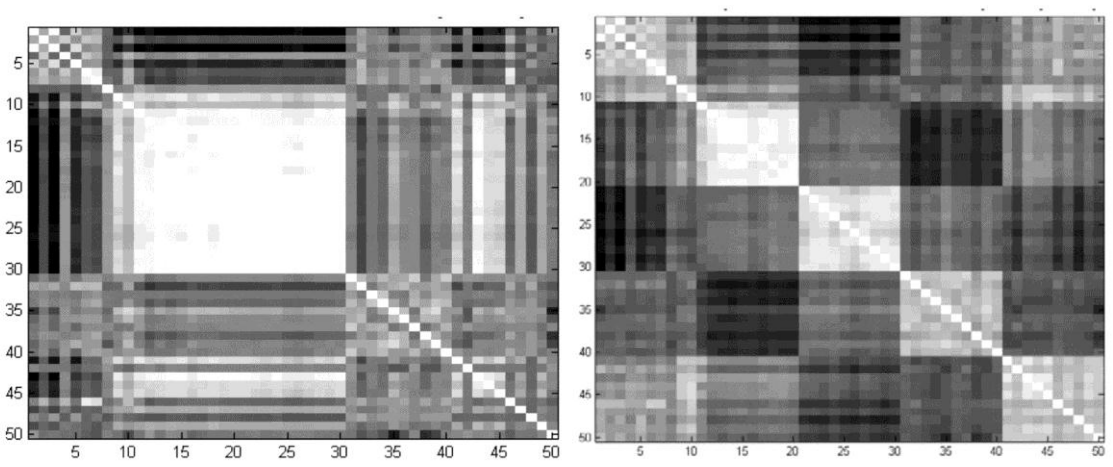


Figure 4.4: Left: Correlation Matrix for Azimuth Angle Feature Right: Correlation Matrix for Average of the scalar angle, azimuth and elevation features

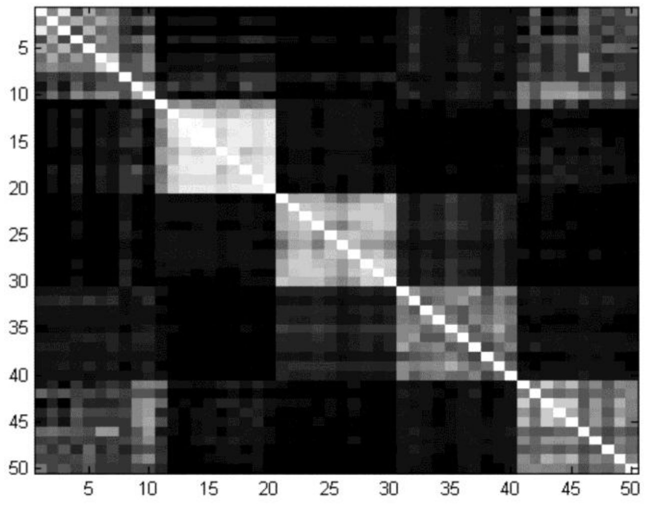


Figure 4.5: Correlation Matrix for Product of the scalar angle, azimuth and elevation features

Although the classification technique in the preliminary work was very primitive, the classification results within the used data set were not exceedingly bad. However, the data set that we have used for testing our algorithm was highly controlled. Now we can use more complicated set of actions to test our proposed methods. To this aim, we use a dataset provided by Microsoft Cambridge Research and later we collected our own dataset. In the remainder sections of this chapter first, we describe the datasets that we used in our tests and then, we report the results of our proposed methods which we applied on these datasets.

4.2 Datasets

In our experiments, we focused on two sets of actions used in MSRC-12 and our novel dataset WorkoutSU-10. In both datasets, we used the provided 3D positions of the joints as determined by Microsoft's markerless motion capture device.

4.2.1 Microsoft Research Cambridge (MSRC-12)

MSRC-12 dataset comprises 12 gesture classes, six of them corresponding to first-person-shooter game actions (iconic gestures) and the other six are gestures of a music player (metaphoric gestures). The gestures are performed by 30 subjects ranging from 22 to 65 years old. The dataset contains 6244 action instances while there are unequal repetitions of each action classes. A list of all gestures is given below:

A. Iconic Gestures

a. Crouch or hide

b. Shoot a pistol

c. Throw an object

d. Change weapon

e. Kick

f. Put on night vision goggles

B. Metaphoric Gestures

a. Start Music/Raise Volume (of music)

b. Navigate to next menu

c. Wind up the music

d. Take a Bow to end music session

e. Protest the music

f. Move up the tempo of the song

Instruction method: To study how the instruction modality affected the movements of the subjects, the data is collected by giving different types of instructions to participants. The participants provided with three familiar, easy to prepare instruction modalities and their combinations that did not require the participant to have any sophisticated knowledge. The three were (1) descriptive text breaking down the performance kinematics, (2) an ordered series of static images of a person performing the gesture with arrows annotating as appropriate, and (3) video (dynamic images) of a person performing the gesture. The goal is that the used mediums to be transparent so they fulfill their primary function of conveying the kinematics. More concretely, for each gesture we roughly have data using: Text (10 people), Images (10 people), Video (10 people), Video with text (10 people), Images with text (10 people).

Descriptive text and static image instructions for iconic and metaphoric gestures come below:

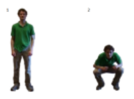
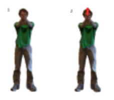
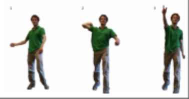
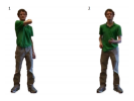
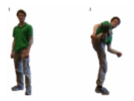
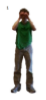
Gesture Outcome	Descriptive Text	Static Images
Crouch or hide	*Squat down or crouch*	
Shoot with a pistol	*Stretching your arms out in front of you and holding your hands together to form a pistol, make a recoil movement*	
Throw an object such as a grenade	*Using your right arm, make an overarm throwing movement*	
Change weapon	*Reach over your left shoulder with your right hand and then bring both hands in front of your body as if you are holding something*	
Kick to attack an enemy	*Karate kick forwards with your right leg*	
Put on night vision goggles to change the game mode	*Bring your hands up to your eyes as if they were goggles*	

Figure 4.6: Descriptive text and static image instructions for iconic gestures [19]

And this is the descriptive text and image of metaphoric gestures:

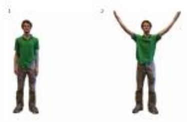
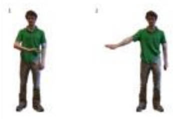
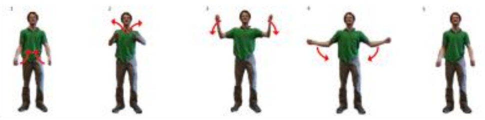
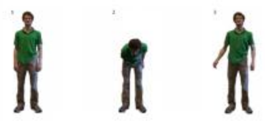
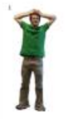
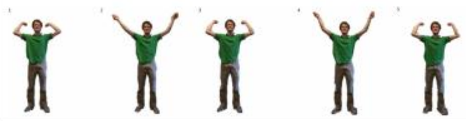
Gesture Outcome	Descriptive Text	Static Images
Start music / raise volume	*Raise outstretched arms*	
Navigate to next menu	*Slide right hand, palm down in front of you, from left to right*	
Wind up the music	*Make circular movements with both arms, in front of your body, clockwise with right hand and counter-clockwise with left hand*	
Take a bow to end the session	*Bend forward at the waist, pause and come back up again*	
Protest the music	Pause and rest your hands on your head	
Lay down the tempo of a song	*Beat the air with both your hands*	

Figure 4.7: Descriptive text and static image instructions for metaphoric gestures [19]

4.2.2 WorkoutSU-10 Dataset

The WorkoutSU-10 exercise dataset comprises a collection of sequences of human body movements represented by 3D positions of skeletal joints. The dataset was collected at Sabancı University in our VPALAB.

The dataset comprises of 1500 sequences in total, collected from 15 people performing 10 different exercises. The motion files contain tracks of 20 joints estimated using the MS Kinect SDK. The body pose is captured at a sample rate of 30Hz.

Participants: The participants were recruited at the Computer Vision Laboratory at Sabancı University (VPALAB). All participants filled a consent form. Each recording session has taken 50 to 90 minutes. Some of the participants were already familiar with the domain of machine learning and computer vision. Relevant statistics of the participants are listed below:

- 73% male,
- 94% right-handed,
- 153-191cm tall with an average height of 179 cm,
- 20 to 30 years old with an average of 24 years of age.

Exercise Types and Repetitions: Exercise types selected by professional coaches are grouped in three categories: Balance, Strength and Flexibility. Some of the initially selected exercises in the preparation steps are eliminated due to incapability of the MS Kinect SDK in tracking the occluded joints of skeleton properly, because of joints occluded by other limbs (the built-in inference mechanism of the algorithm didn’t work satisfactorily). Exercises with least occlusion issues were selected, while one cannot totally get rid of this problem. The list of all exercises is provided in the sequel. Each exercise has been repeated 10 times by each subject.

(A) Balance Exercises

(A1) SL (single leg) Balance with Hip Flexion

(A2) SL (single leg) Balance-Trunk Rotation

(A3) Lateral Stepping

(B) Stretching and Flexibility Exercises

(B1) Thoracic Rotation Bar on shoulder

(B2) Hip Adductor Stretch

(B3) Hip Adductor Stretch

(C) Strengthening Exercises

(C1) DB (Dumbbell) Curl-to-Press

(C2) Freestanding Squats

(C3) Transverse Horizontal DB Punch

(C4) Lateral Trunk/Oblique Stretch

Instruction Method: We have provided the participants with two familiar, easy-to-grasp instructions that did not require any sophisticated knowledge. The modalities were (1) descriptive text breaking down the performance kinematics and (2) video (dynamic images) of a person performing the exercise. All of the videos were of the trainer performing the exercises as defined by application's designer and started and stopped with the beginning and end of the exercise.

The data collection process has been carried out in two phases. In the first phase, we have recorded template exercises that will be our reference in subsequent processes with assistance of professional sport trainers. At the second phase, we have recorded exercises performed by ordinary subjects.

First Phase: Template recording with sports trainers

- One male, one female subjects,
- Record a set which each set contains 10 repetitions of the exercise,
- Recording session should sweep all action classes to complete one set, then proceed with the next set and so on.

Second Phase: Recording with ordinary people

- Multiple subjects (computer vision lab students)
- Record a set which each set should contain 10 repetitions of the action (with 10 class, each subject will end up performing 100 action instances)
- Recording session should sweep all action classes to complete one set, then proceed with the next set and so on

The following data have been recorded from each subject:

- Body Pose sequences (3D positions of the 20 skeletal joints tracked by MS Kinect SDK)

- Depth sequences
- RGB video

RGB video recording has been conducted with a video camera (Canon Legria HD CMOS). I/O streaming processes are slow by themselves therefore the depth values are recorded with Kinect depth cameras but the RGB data have been captured through a separate video camera instead of the Kinect RGB camera.

Data Recording Site: The recording procedure has taken place at VPALAB room at Sabancı University. We tried to maintain the recording environment same during all the sessions (same location, fixed distances and no abrupt changes in lighting conditions). All the recordings have been carried out while the subject performing the exercises stood at ~2.75 meter distance in front of the Kinect sensor (considered as the ideal distance for the Kinect sensor). A green screen used as background during recordings to facilitate the potential use of RGB recordings as well. Figure 4.8 shows the experimental setup for data collection at location.

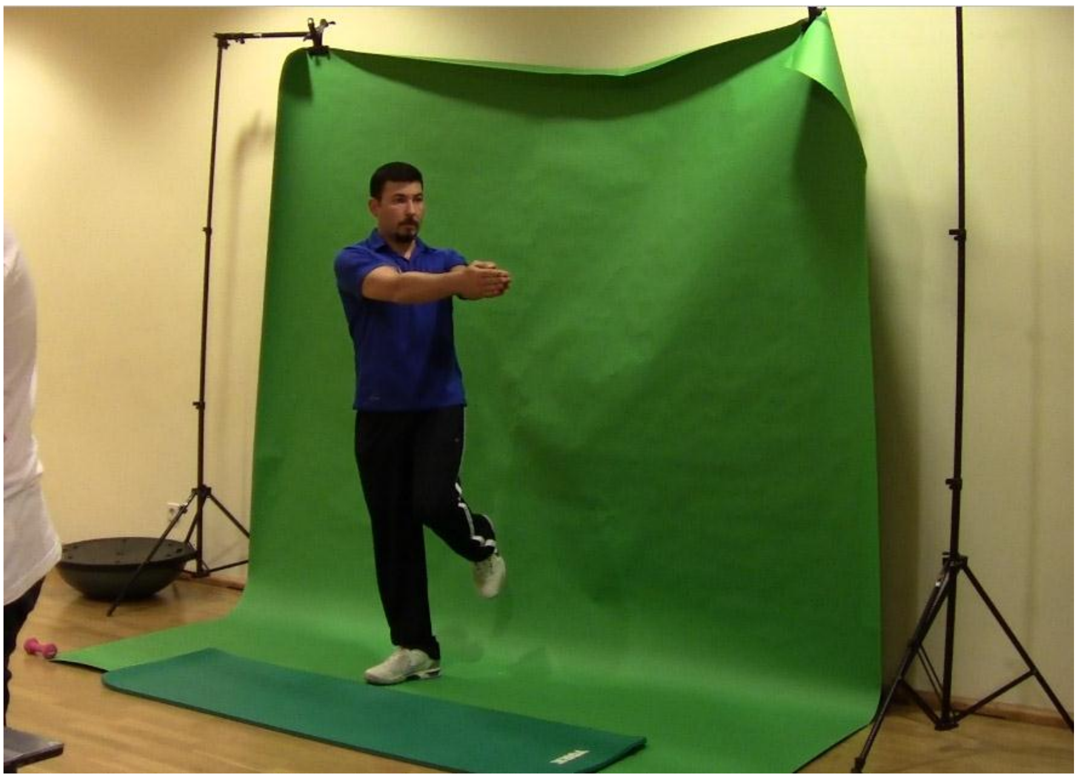


Figure 4.8**:** Data Recording Site (Sabanci University)

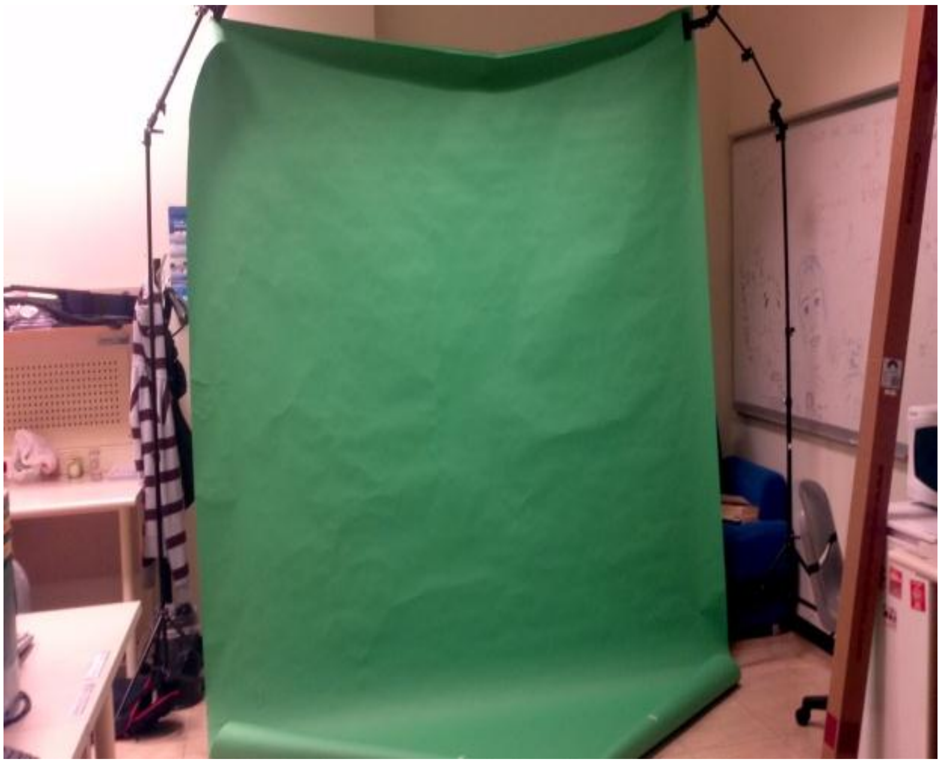
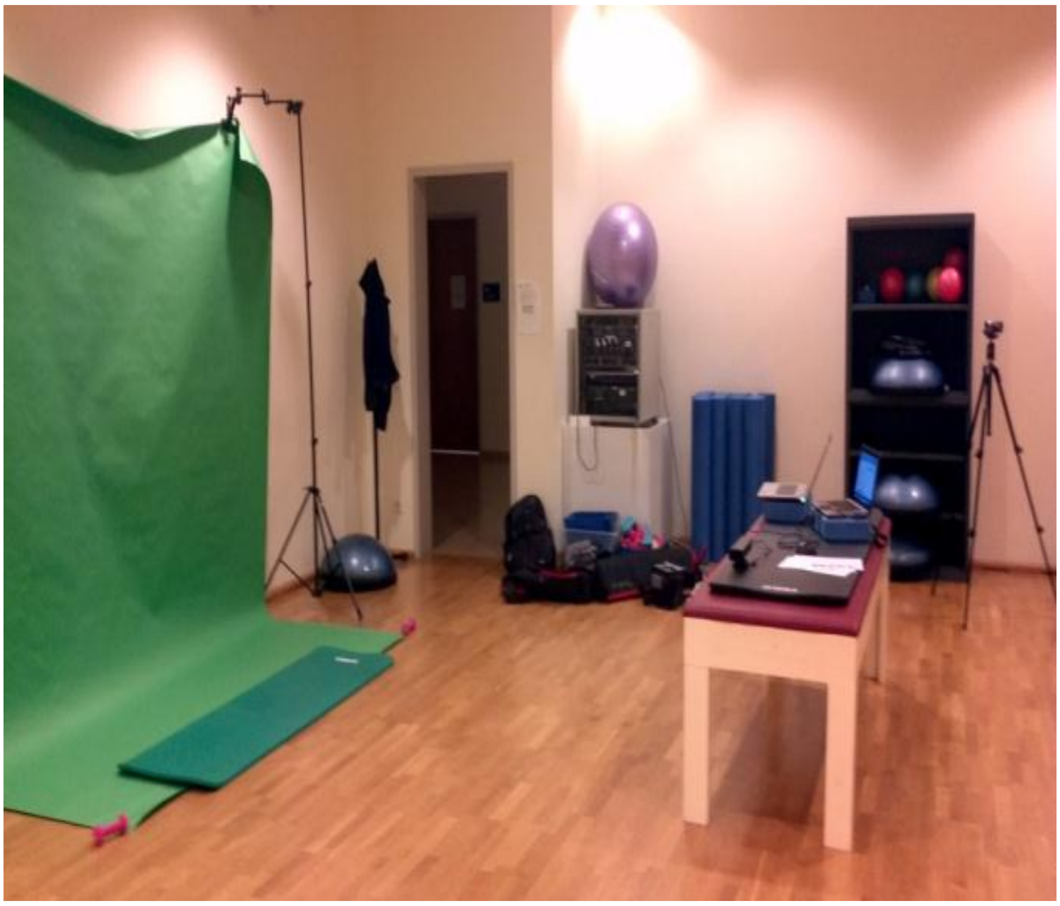
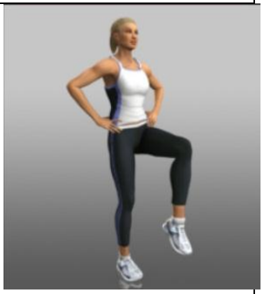
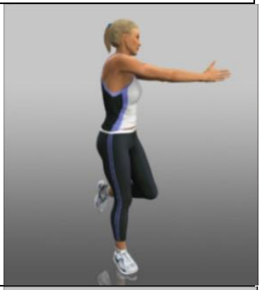
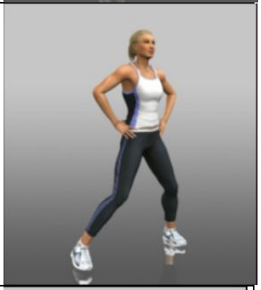
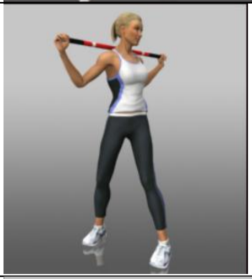
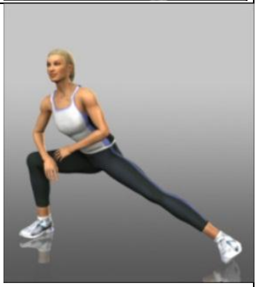
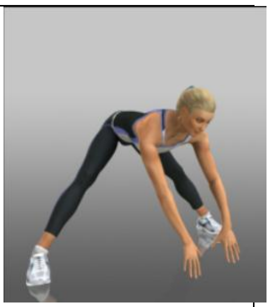
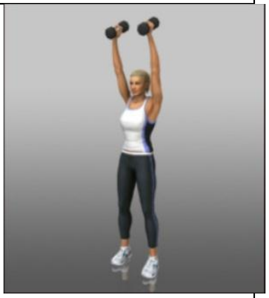
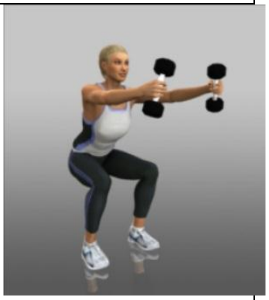
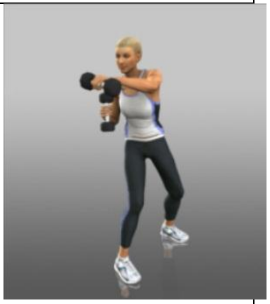
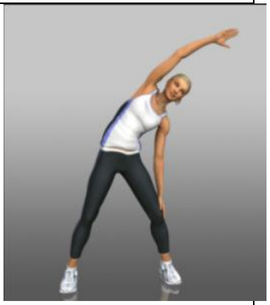



Figure 4.8**:** Data Recording Site (Sabanci University) (continued)

In Table 4.2, you can find all of the exercises and respective instructions of each.

Table 4.2: Exercise Illustrated Description of Exercise Types

Exercise outcome/Code	Descriptive Instruction	Image
SL Balance with Hip Flexion (A1)	• Flex your hip of your non-weight bearing leg up to 90 degrees, bend your knee, and hold. • Use your core & lower extremity muscles to control your center of mass to maintain your balance.	
SL Balance-Trunk Rotation (A2)	• Raise your arms to chest height and clasp your hands together. • Slowly rotate your trunk to one side a comfortable distance, return to the starting position, and then rotate your trunk in the other direction. • Use your core & lower extremity muscles to control your center of mass to maintain your balance.	
Lateral Stepping (A3)	• Slightly bend your knees and begin stepping to the side keeping your toes facing straight ahead. • Use your core & lower extremity muscles to control your center of mass to maintain your balance. • Perform this for a specific number of steps then return back in the other direction.	
Thoracic Rotation – Bar on shoulder (B1)	• Assume standing position with bar across shoulders. • Rotate your trunk to one side. • Hold 30 (s) at end range; then slowly release stretch.	
Hip Adductor Stretch (B2)	• Shift your weight over one leg by bending your knee and straighten the opposing leg to be stretched. • You should feel a stretch on the inside aspect of your thigh and groin of the straight leg. • Hold 30 (s) at end range; then slowly release the stretch.	

Hip Adductor Stretch (B3)	• Assume a standing position with your feet straight ahead much wider than shoulder width. • Forward bend your trunk and reach as far forward as possible while your hips and buttocks move backwards in the opposite direction. • You should feel a stretch on the inside aspect of your thigh and groin.	
DB Curl-to-Press (C1)	• Hold dumbbells at your sides with your palms facing in • Brace your lower torso by contracting your abdominals and back muscles • Curl both dumbbells in front of your shoulders, then press them straight up toward the ceiling • Lower the dumbbells back down to the same position in front of your shoulders, and then return them to your sides to complete each repetition	
Freestanding Squats (C2)	• Lower into a full squat position (90 degrees of hip and knee flexion) and allow your arms to rise, thumb up, to shoulder height • Lower your arms to your sides as you return to standing • Try not to allow your knees to cross forward over your toes • Maintain an upright trunk and neutral lumbar spine with abdominals braced	
Transverse Horizontal DB Punch (C3)	• Brace your lower torso by contracting your abdominals and low back muscles. • Rotate your trunk to one side and punching the opposite dumbbell across your body until your elbow is completely straight. • Return to the starting position and repeat the same movement in the opposing direction. • Continue to alternate sides until the set is complete.	
Lateral Trunk/Oblique Stretch (C4)	• Assume a standing position with your feet slightly wider than shoulder width. • Raise one arm overhead and side-bend towards the opposite side without forward bending.	
Illustrations reference: http://www.perfectfitpro.com/		

4.2.3 Fall Dataset

Other than therapy activities, "fall detection" in elderly patients has utmost importance. To pursue our medical motivations we collected a new dataset and we call it fall dataset in addition to our previously recorded WorkoutSU-10 dataset and next we evaluate these data with our proposed method to find out if it's also applicable to this kind of data or not.

This new fall dataset comprises of 180 sequences in total, collected from 10 people performing 3 different exercises. The recorded motions contain tracking of 20 joints estimated using the MS Kinect SDK. The body pose is captured at a sample rate of 30Hz. The participants were recruited at the ISRA Vision Laboratory. All participants filled a consent form and each recording session has taken 30 to 60 minutes. The exercises are designed to recognize the "fall" action in different circumstances in a daily routine of a patient:

Walking:	Falling:	Sitting Down:
- The subject starts from Point A and walk through point B and stops.	- The subject starts from point A and walks thorough point B but falls down at the middle of the path	- The subject starts from point A and walks thorough point B but sits down at the middle of the path on a provided chair

There are three action classes in dataset: "walking", "falling" and "sitting down". Before conducting the actions the subjects provide with instructions of the actions with different modalities (text, video footage and snapshots). In "walking" class, the subject starts walking from point A and stops walking when s/he reaches to point B in experiment's location. In the "falling" action, the subject starts walking from point A through point B but falls down at midway. In "sitting down" exercise the subject starts walking from point A and sits down on a provided chair at midway and stands up and continues until point B (Figure 4.9).

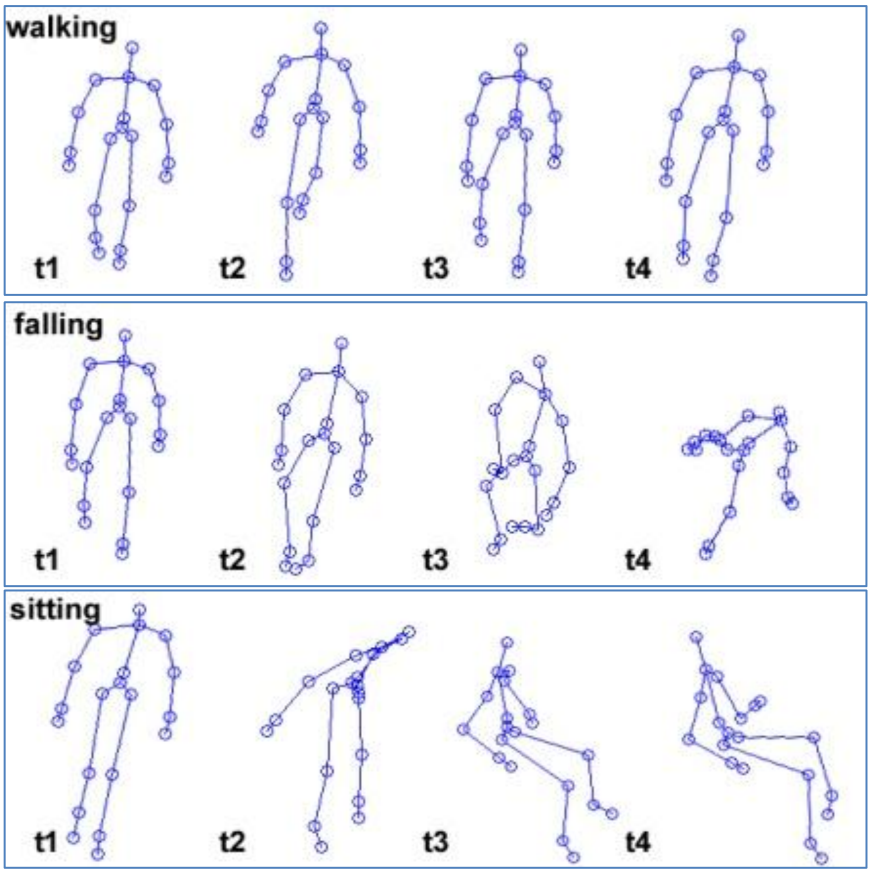


Figure 4.9: Actions in fall dataset

The recording procedure has taken place at VISTEK Company in Istanbul. We tried to maintain the recording environment same during all the sessions (same location, fixed distances and no abrupt changes in lighting conditions). All the recordings have been carried out while the subject performing the exercises stood at ~3 meter distance in front of the Kinect sensor (considered as the ideal distance for the Kinect sensor). Figure 4.10 shows the experimental set-up for data collection at location.

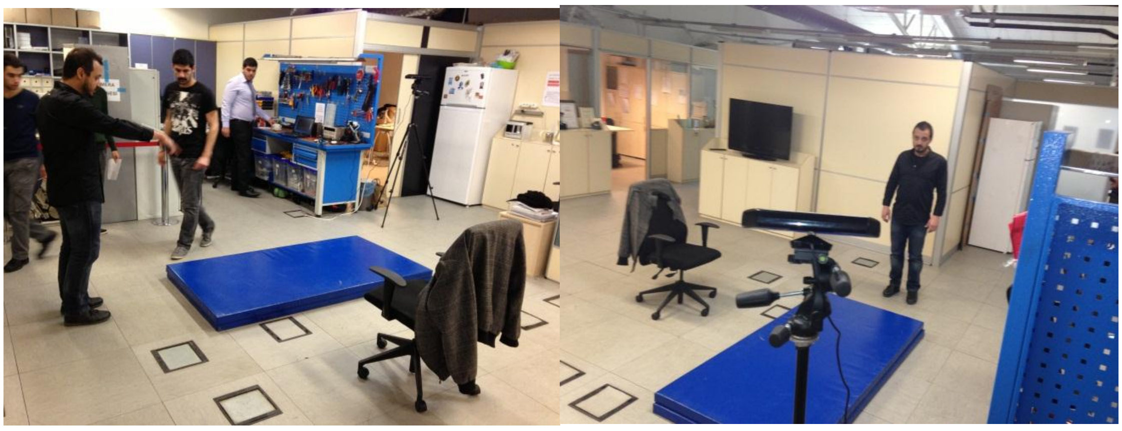


Figure 4.10: Data gathering site (VISTEK Inc.)

4.3 Training and Testing Strategies and Results

In this section we report the test results of applying our methods we described in chapter 3 on the datasets we just explained. First, we report results of template matching strategies and next, the synergetic classifier results. Then we can compare the results and exhibit our methods efficiency.

4.3.1 Template Matching Results

For template matching using the aggregation methods described in section 3.2, after construction of the correlation pool between each pair of action instances in dataset by correlation coefficient and dynamic time warping similarity measure, we have carried out a leave-one-subject-out cross-validation (LOSOXV) test on the MSRC-12 dataset and WorkoutSU-10 dataset. As mentioned before, we try to obtain a baseline performance on these dataset. The evaluation test has been repeated for K-NN, K-first average and K-first product method K=1, 3, 5, and 7. We have linearly interpolated and downsampled all time-series instances to 50 samples. We report the performance from Table 4.3 to Table 4.6. The best aggregation methods turn out to be classwise K-first product and classwise K-first average with K = 3. We also note that all schemes except whole classwise average and Borda count, performances are similar within 0.8% performance points. Aggregation schemes use all the KN=13300 features with a simple but powerful NN-based classifier. As such, they provide a baseline performance that, with our RDF-based feature selection mechanism, we aim at maintaining while reducing the number of features significantly.

Here, we can compare the results of two similarity metric methods. We were expected that the DTW method would improve the classification accuracy and actually it did. But this boost in accuracy is not that significant. As we briefly mentioned this point before this is because of the nature of the time-series. The time-series in the datasets are captured in a very controlled environment with precise start and stop time points. That is, there is not so much misalignment effect or distortion in the form of the signal occurs. But the benefit of this approach will appear in the real-world scenarios where, there is erratic and spontaneous actions (time-series) which we will have no control over them. This is very important subject where we should cope with especially in application development phase.

4.3.2 Synergetic RDF and SVM Based Classification Results

Now that we have all our baseline performances, the classifier-based approaches will apply to investigate the effect of our proposed synergetic method.

In RDF-based feature selection, in order to find the best forest configuration, we have trained 27 forests with different tree sizes (10, 50 and 200 tree), tree depths (4, 6 and 8), and number of selected features at each node (10,100 and 1000). We have observed that the number of thresholds to test at each node had no influence in the performance. In the validation runs, we have used cross-subject cross-validation (CSXV). Accordingly, we have split each dataset (MSRC-12 and WorkoutSU-10) into two groups A $(training)$ and B $(testing)$. For MSRC-12, the number of instances in sets A and B was 3600 and 2645 respectively. For WorkoutSU-10, the split has resulted in 600 instances in both sets A and B. We have reserved the set A for training the 27 RDFs and the set B to evaluate the cross-validation performance.

Figure 4.11 depicts CSXV validation vs. feature reduction efficiency curve on MSRC-12. We picked the configuration giving the most sensible compromise between accuracy (93.2%) and feature reduction efficiency (91%), that is, the forest with number of trees = 50, maximum tree depth = 6 and number of selected features at each node = 100. It's reassuring to see that the 93.2% CSXV performance obtained with this scheme coincides with the 93.0% LOSOXV performance obtained with the full feature set using aggregation on MSRC-12. The nodes of the RDF trained with this configuration contained 1225 unique features, corresponding to a reduction better than one tenth with respect to 13300 features in total. After this selection procedure, we have trained a linear SVM classifier using the 1225 retained features on the set A of MSRC-12. We have applied a 5-fold cross-validation to find the regularization parameter of the SVM. The classification performance on set B of MSRC is shown in Table 4.7-left. It can be observed that the linear SVM using the reduced feature improves the performance even further (by 1%). The results of the same procedure on WorkoutSU-10 are shown in Table 4.7-right, the average accuracy turns out to be 98% with a quite balanced classwise performance. Note that the RDF trained with the best configuration above has yielded 1398 unique features on WorkoutSU-10 corresponding to 89.8 % reduction efficiency. Confusion matrices for individual classes in both datasets are shown in Figure 4.12 top and Figure 4.12 down.

Class	K-NN				Classwise K-First Product			Classwise K-First Average			Whole Classwise Average	Borda Count
	K				K			K				
	1	3	5	7	3	5	7	3	5	7		
Kick	96.1	96.5	96.1	96.5	**96.5**	96.5	96.3	**96.5**	96.5	96.3	91.4	91.6
Beat Both	84.5	84.7	85.7	86.0	**85.5**	86.2	85.9	**85.5**	86.2	85.9	84.1	32.0
Change Weapon	85.7	84.9	84.1	82.5	**85.3**	84.1	83.9	**85.3**	84.1	83.9	70.1	43.8
Had enough	91.5	90.2	90.0	90.4	**90.6**	90.0	90.2	**90.6**	90.0	90.2	67.5	53.9
Throw	95.9	95.5	95.3	95.5	**95.7**	95.7	95.7	**95.7**	95.7	95.7	89.5	81.7
Bow	98.8	98.8	98.4	98.2	**99.0**	98.6	98.4	**99.0**	98.6	98.6	93.9	89.3
Shoot	86.9	85.1	84.5	83.6	**86.5**	85.9	85.5	**86.5**	85.9	85.5	41.5	16.4
Wind Up	95.8	95.7	95.2	95.1	**95.8**	95.4	95.8	**96.0**	95.5	95.8	52.1	81.0
Goggles	96.1	97.5	97.3	97.3	**97.1**	97.3	97.3	**97.1**	97.3	97.3	93.5	92.6
Push Right	91.8	92.9	92.1	91.2	**93.3**	92.7	92.0	**93.3**	92.7	92.0	58.0	46.7
Duck	99.6	99.4	99.4	99.4	**99.6**	99.6	99.4	**99.6**	99.6	99.4	98.2	90.4
Lift Arms	89.2	89.6	89.8	90.2	**90.0**	90.6	90.4	**90.0**	90.6	90.2	83.3	68.7
Average	92.7	92.6	92.4	92.2	**93.0**	92.8	92.6	**93.0**	92.8	92.6	76.3	66.0
standart deviation	5.15	5.52	5.46	5.77	**5.19**	5.30	5.37	**5.20**	5.30	5.40	18.68	26.30

Table 4.3: LOSOXV Performance Results of Aggregation Methods on MSRC-12 Using Correlation Coefficient as Scoring

Class	K-NN				Classwise K-First Product			Classwise K-First Average			Whole Classwise Average	Borda Count
	K				K			K				
	1	3	5	7	3	5	7	3	5	7		
Kick	95.6	97.1	96.3	95.7	**97.4**	95.4	95.7	96.8	96.5	96.1	92.0	93.1
Beat Both	88.9	87.7	85.0	88.1	**84.5**	89.2	85.0	85.9	88.3	87.2	88.1	48.2
Change Weapon	88.3	85.1	86.0	86.9	**87.0**	87.1	88.5	86.4	84.8	84.4	75.2	40.5
Had enough	94.4	90.1	89.9	94.2	**93.5**	92.1	88.2	90.9	89.8	91.3	65.6	59.7
Throw	93.8	96.6	95.8	95.8	**95.0**	94.8	98.6	96.4	96.9	96.4	93.1	88.2
Bow	96.6	97.1	98.5	97.1	**98.3**	98.4	98.1	98.3	98.1	96.8	96.1	87.3
Shoot	90.1	88.4	88.4	89.5	**93.3**	83.1	87.9	88.7	92.7	98.5	51.3	25.1
Wind Up	93.0	95.0	95.5	94.4	**96.5**	95.3	94.8	98.2	95.9	88.5	53.9	80.5
Goggles	93.3	96.9	96.0	97.2	**97.3**	97.5	97.9	97.2	97.1	98.6	96.2	88.2
Push Right	93.7	92.8	91.8	94.9	**94.1**	92.9	94.7	94.8	93.3	92.3	57.1	48.8
Duck	99.6	99.4	99.4	99.4	**99.6**	99.4	99.6	99.6	99.4	99.6	99.4	93.1
Lift Arms	94.5	94.9	94.4	92.9	**92.3**	90.2	91.8	92.3	92.3	92.9	84.1	66.2
Average	93.5	93.4	93.0	93.8	**94.1**	92.9	93.4	93.7	93.7	93.5	79.3	68.2
standart deviation	3.20	4.55	4.78	3.84	**4.49**	4.86	4.96	4.81	4.40	4.93	17.90	23.40

Table 4.4: LOSOXV Performance Results of Aggregation Methods on MSRC-12 Using DTW as Scoring Method

Class	K-NN				Classwise K-First Product			Classwise K-First Average			Whole Classwise Average	Borda Count
	K				K			K				
	1	3	5	7	3	5	7	3	5	7		
Hip Flexion	94.6	94.9	94.6	94.5	95.2	**94.9**	94.6	94.9	94.9	94.5	92.3	92.1
Trunk Rotation	89.3	88.9	88.6	88.1	88.6	**89.7**	88.3	89.9	88.7	90.9	65.7	69.4
Lateral Stepping	92.5	92.3	92.7	91.7	92.5	**93.1**	92.3	93.5	93	93.1	90.4	90.8
Thoracic Rotation	84.5	83.7	83.5	83.9	83.9	**84.9**	83.6	84.9	83.4	83.7	78.3	71.4
Hip Adductor Stretch	86.8	86.8	86.2	85.9	86.7	**87.8**	86.9	86.5	86.8	86.1	72.6	39.2
Hip Stretch	88.3	87.1	88.5	88.1	88.5	**88.5**	88.3	88.1	87.3	88.2	68	41.8
Curl-To-Press	96.2	95.3	94.9	95.3	96.1	**96.4**	96.8	96.2	95.1	96.4	91.4	72.7
Free Standing Squats	87.1	86.3	86.7	86.3	86.7	**87.3**	86.5	84.7	85	84.3	83.7	27.3
Horizontal Punch	93.9	92.2	94	93.3	93.2	**94.2**	93.6	92.5	92.2	93.6	45.1	78
Oblique Stretch	94.3	91.4	91.6	91.4	93.9	**94.3**	94.1	94.1	93	94.1	38.9	92
Average	90.75	89.9	90.13	89.85	90.53	**91.11**	90.5	90.53	89.94	90.49	72.64	67.47

Table 4.5: LOSOXV Performance Results of Aggregation Methods on WorkoutSU-10 Using Correlation Coefficient as Scoring Method

Class	K-NN				Classwise K-First Product			Classwise K-First Average			Whole Classwise Average	Borda Count
	K				K			K				
	1	3	5	7	3	5	7	3	5	7		
Hip Flexion	97.1	98.2	97.1	97.2	**96.9**	95.9	97.4	98.1	97.5	97.3	92.9	95.6
Trunk Rotation	93.2	91.9	93.1	92.2	**93.7**	91.6	94.1	92.8	93.1	91.5	66.9	64.8
Lateral Stepping	96.5	96.2	96.1	96.2	**96.7**	96.1	96.4	95.8	93.2	94.3	90.1	92.1
Thoracic Rotation	88.2	88.1	87.3	87.6	**88.1**	87.3	86.1	88.5	88.1	86.3	82.1	45.4
Hip Adductor Stretc	89.1	86.2	87.5	84.2	**86.9**	86.1	81.2	87.1	86.5	86.2	72.9	40.2
Hip Stretch	90.9	87.3	93.5	92.1	**92.1**	88.8	89.7	92.1	89.3	89.6	61.7	54.8
Curl-To-Press	98.2	97.2	98.6	98.2	**98.6**	98.6	98.2	97.9	94.9	98.2	91.2	75.1
Free Standing Squa	88.3	87.3	83.2	83.8	**88.2**	86.4	87.3	88.1	87.6	87.3	77.1	23.7
Horizontal Punch	95.0	96.1	96.3	96.1	**96.1**	95.7	94.1	97.2	96.0	96.1	58.5	85.2
Oblique Stretch	97.3	95.8	96.1	95.7	**95.2**	96.3	96.1	94.9	97.2	96.1	90.1	89.2
Average	93.29	92.43	92.88	92.33	**93.25**	92.28	92.06	93.24	92.34	92.29	78.35	66.61

Table 4.6: LOSOXV Performance Results of Aggregation Methods on WorkoutSU-10 Using DTW as Scoring Method

We also tried to assure the effect of the SVM classifier on the classifiers performance. After obtaining the unique features, we put these obtained features into RDF and used it as either as classifier. For MSRC-12 dataset using reduced feature set we obtained 93.4% average accuracy using RDF as classifier which is almost coincide with the performance of the SVM classifier. But the effect of the SVM classier appears when we use it for WorkoutSU-10 dataset classification. The RDF classifier obtains 83.6% average classification accuracy on this dataset which when we compare it with 98% accuracy of the SVM dataset, the significant outperform is obvious.

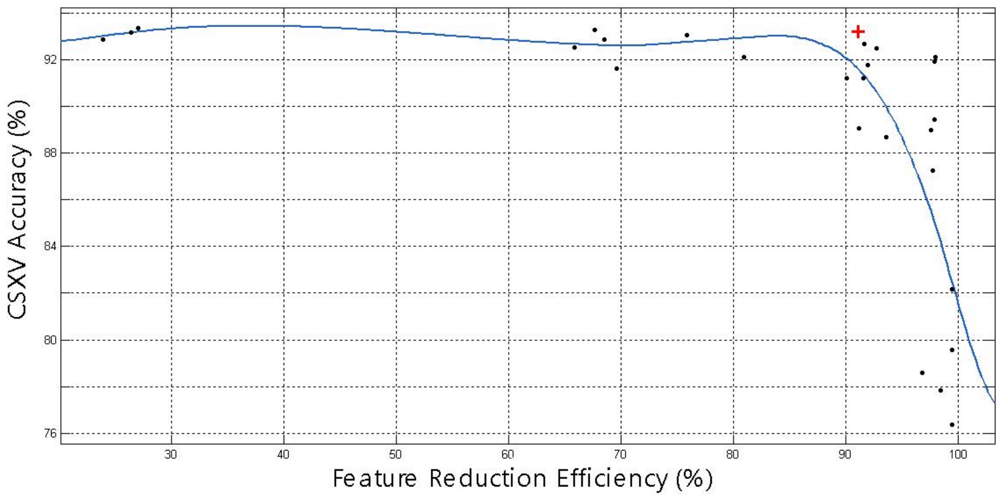
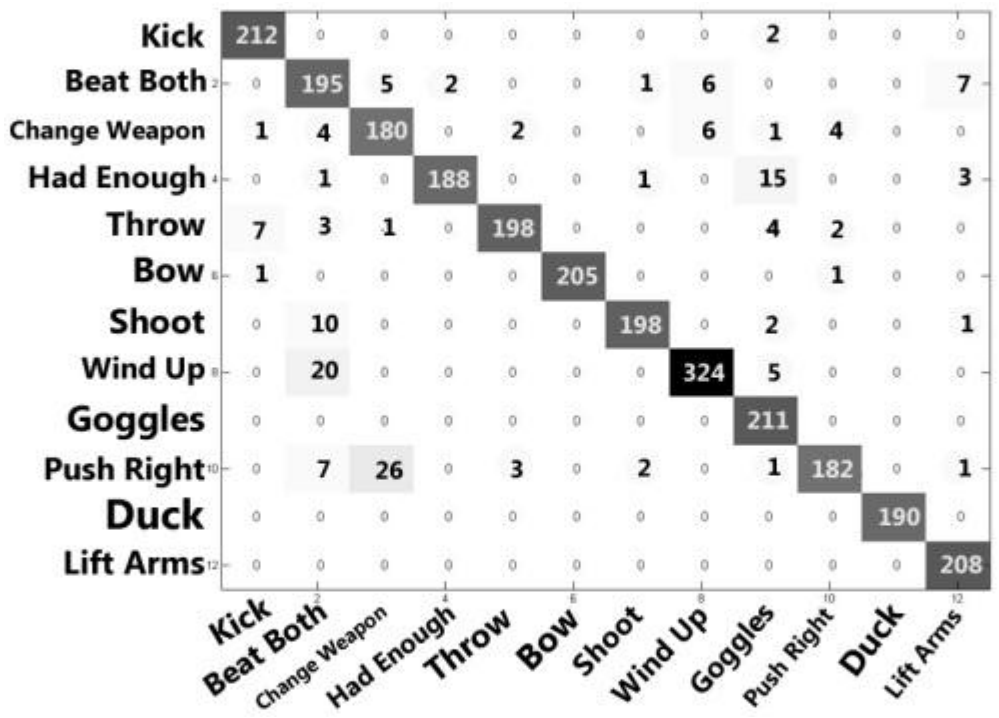


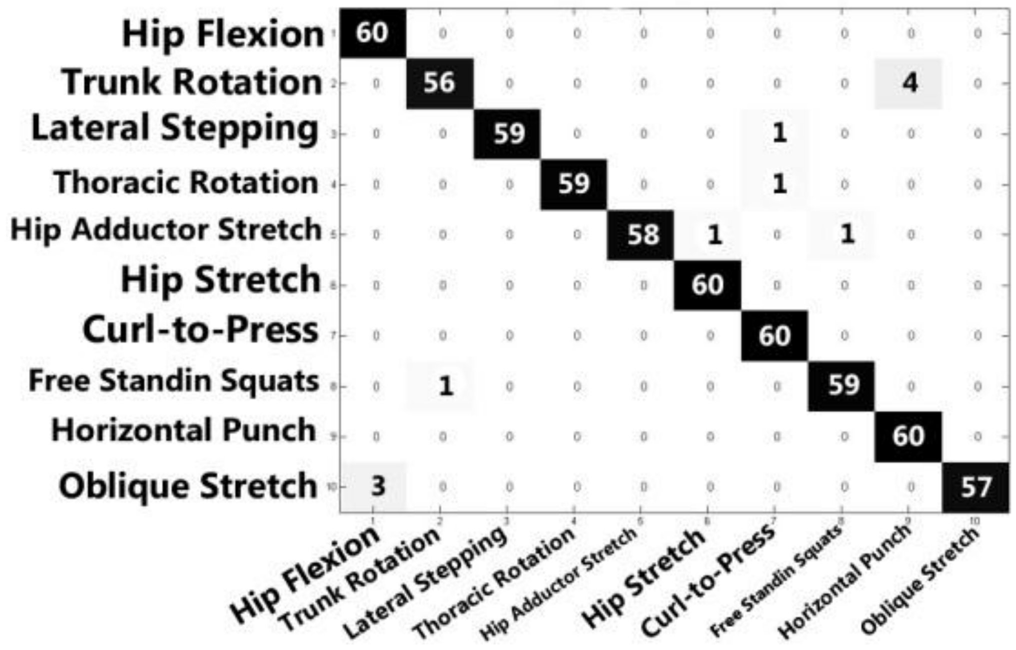
Figure 4.11: Selection of the best forest configuration using the accuracy vs. feature reduction efficiency measure on MSRC-12

Table 4.7: Classification Performance of the Linear SVM with RDF-selected features

MSCR-12		WorkoutSU-10	
Class	Accuracy	Class	Accuracy
Kick	99	Hip Flexion	100
Beat Both	90.2	Trunk Rotation	93.3
Change Weapon	90.9	Lateral Stepping	98.3
Had Enough	90.3	Thoracic Rotation	98.3
Throw	92	Hip Adductor Stretch	96.6
Bow	99	Hip Stretch	100
Shoot	93.8	Curl-To-Press	100
Wind Up	92.8	Free Standing Squats	98.3
Goggles	100	Horizontal Punch	100
Push Right	81.9	Oblique Stretch	95
Duck	100		
Lift Arms	100		
Average	94.03	Average	98
standart deviation	5.62		2.34



	Kick	Beat Both	Change Weapon	Had Enough	Throw	Bow	Shoot	Wind Up	Goggles	Push Right	Duck	Lift Arms
Kick	212	0	0	0	0	0	0	0	2	0	0	0
Beat Both	0	195	5	2	0	0	1	6	0	0	0	7
Change Weapon	1	4	180	0	2	0	0	6	1	4	0	0
Had Enough	0	1	0	188	0	0	1	0	15	0	0	3
Throw	7	3	1	0	198	0	0	0	4	2	0	0
Bow	1	0	0	0	0	205	0	0	0	1	0	0
Shoot	0	10	0	0	0	0	198	0	2	0	0	1
Wind Up	0	20	0	0	0	0	0	324	5	0	0	0
Goggles	0	0	0	0	0	0	0	0	211	0	0	0
Push Right	0	7	26	0	3	0	2	0	1	182	0	1
Duck	0	0	0	0	0	0	0	0	0	0	190	0
Lift Arms	0	0	0	0	0	0	0	0	0	0	0	208

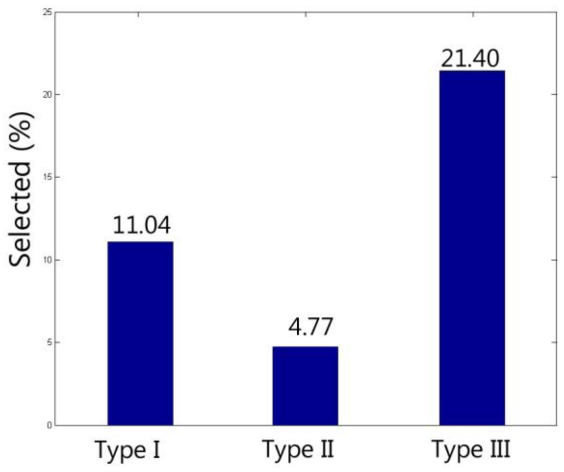


	Hip Flexion	Trunk Rotation	Lateral Stepping	Thoracic Rotation	Hip Adductor Stretch	Hip Stretch	Curl-to-Press	Free Standin Squats	Horizontal Punch	Oblique Stretch
Hip Flexion	60	0	0	0	0	0	0	0	0	0
Trunk Rotation	0	56	0	0	0	0	0	0	4	0
Lateral Stepping	0	0	59	0	0	0	1	0	0	0
Thoracic Rotation	0	0	0	59	0	0	1	0	0	0
Hip Adductor Stretch	0	0	0	0	58	1	0	1	0	0
Hip Stretch	0	0	0	0	0	60	0	0	0	0
Curl-to-Press	0	0	0	0	0	0	60	0	0	0
Free Standin Squats	0	1	0	0	0	0	0	59	0	0
Horizontal Punch	0	0	0	0	0	0	0	0	60	0
Oblique Stretch	3	0	0	0	0	0	0	0	0	57

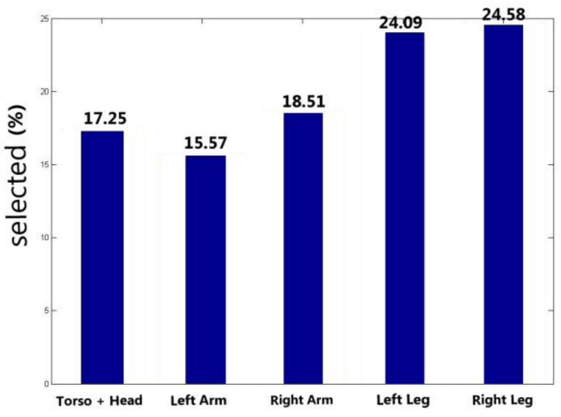
Figure 4.12: Confusion matrices for MSRC-12 (top) and WorkoutSU-10 (down) using linear SVM with RDF-selected features

4.3.3 Performance Assessment

By data mining the results, we can investigate the feature selection process in order to find out which feature type has the most effect on the performance of the classifier. In Figure. 4.13(a), the ratio of selected features in each type of features with respect to the total number of features in that type is plotted. Since we have a large initial feature set, there is considerable reduction in number of features used in each type. Feature type III (velocity of the joints) is the most selected type in tree nodes, suggesting they are the most influential features for the classifier performance. We have also looked at the ratio of the selected features in skeleton by group of joints each one belongs to in Figure. 4.13(b). It can be observed that features from leg joints have more impact the remaining joints, which have more or less the same influence.



(a)



(b)

Figure 4.13: (a) Ratio of selected features of each feature type on MSRC-12, (b) Ratio of selected features with respect to joint categories on MSRC-12

4.3.4 Fall Detection Performance Assessments

We run tests on new dataset to find baseline performance of the dataset using the correlation score metric. The results of baseline come in table 4.8:

Class	k-NN			Average	MAX	Product	k-first product			k-first Average			Borda Cou
	k = 3	k = 5	k = 7				k = 3	k = 5	k = 7	k = 3	k = 5	k = 7	
walking	95	98.33	96.66	86.66	95	90	90	83.33	73.33	98.33	98.33	98.33	81.66
Falling	86.66	83.33	78.33	80	90	86.66	80	68.33	58.33	96.66	95	90	80
Sitting	95	95	93.33	93.33	96.66	96.66	95	93.33	90	98.33	96.66	95	93.33
Average Accuracy	92.73	92.73	89.94	87.15	94.41	91.62	88.82	82.12	74.3	98.32	97.2	94.97	85.47

Table 4.8: Baseline performance results for fall dataset

As mentioned before despite the popularity of decision forests in classification tasks we use this method to find the most discriminative features in our feature set. We do the same procedure we have done for the other datasets where the whole feature set stream into the forest (instead of bagging) and the best parameters of the weak learners extracted through random node optimization process. We observe that even with a small subset of whole feature set we can classify the test instances with a remarkable accuracy. With fine tuning of the forest's parameters the optimal subset of features found and at the final step employed to train a SVM machine.

Again In order to find the best forest configuration we trained 27 forests with different tree sizes (10,50 and 200 tree), tree depths (4,6 and 8), number of selected features at each node (10,100 and 1000) and number of threshold tests (10). In this step we used cross-subject cross-validation on dataset. We split the datasets into two group A $(training)$ and B $(testing)$. For fall dataset we split equally that there are 90 instances in both sets A and B. As shown in Figure 4.14 the best configuration expected to take place in top right area of the figure. We selected the configuration with highest feature reduction efficiency (0.976) and the best performance result (97.77%), that is the forest with tree number = 50, maximum tree depth = 4 and number of selected features at each node = 100. The results of performances for each forest come below:

Num ber of Trees	**De pth of Tree**	**Num ber of Splits**	**Num-ber of Unique Features**	**Reduc-tion Effi-ciency**	**Accu-racy**
200	8	1000	9620	0,299	92.222
200	8	100	10259	0,253	94.333
200	8	10	10009	0,271	97.777
200	6	1000	2710	0,803	93.111
200	6	100	3520	0,744	95,112
200	6	10	4728	0,656	96.669
200	4	1000	560	0,960	94.336
200	4	100	1112	0,936	96.665

200	4	10	1271	0,908	97.777
50	8	1000	4223	0,697	93.435
50	8	100	3748	0,677	94.444
50	8	10	3880	0.718	93.333
50	6	1000	1352	0.902	95.555
50	6	100	1384	0.900	96.667
50	6	10	1401	0.901	94.444
50	4	1000	289	0.980	92.223
50	4	100	330	0.976	97.777
50	4	10	339	0.976	96.666
10	8	1000	832	0.940	91.111
10	8	100	891	0.936	91.116
10	8	10	863	0.938	91.111
10	6	1000	289	0.979	92.220
10	6	100	295	0.979	93.331
10	6	10	292	0.979	87.777
10	4	1000	68	0.996	90.00
10	4	100	69	0.995	94.44
10	4	10	70	0.995	95.55

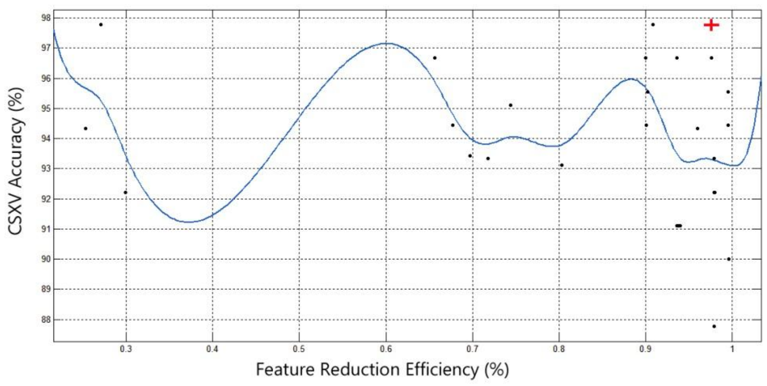
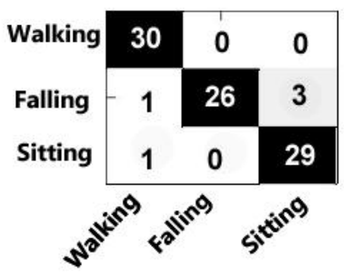


Figure 4.14: Feature reduction process

Then we used the parameters of the selected forest to train a SVM classifier. The SVM classifier trained with instances in set *A* with 330 unique features selected by forest. We applied a 5-fold cross-validation to find the best parameters of the SVM classifier. The results of assigning of the optimized classifier on set *B* of fall datasets are shown in Figure 4.15 (a) and confusion matrix is shown in Figure 4.15 (b).

Results of SVM and confusion matrix:

Fall Dataset	
Class	Accuracy
Walking	100
Falling	86.66
Sitting	96.4
Average	94.44
standart deviation	2.24



	Walking	Falling	Sitting
Walking	30	0	0
Falling	1	26	3
Sitting	1	0	29

Figure 4.15: (a) classification results of SVM classifier (b) confusion matrix

Like before by data mining the results, we can investigate the feature selection process in order to find out which feature type has the most effect on the performance of the classifier. In bar graph shown below, the ratio of selected features in each type of features with respect to the total number of features in that type is plotted. Since we have a large initial feature set, there is considerable reduction in number of features used in each type. Feature type II (Euclidian distance of the joints) is the most selected type in tree nodes, suggesting they are the most influential features for the classifier performance:

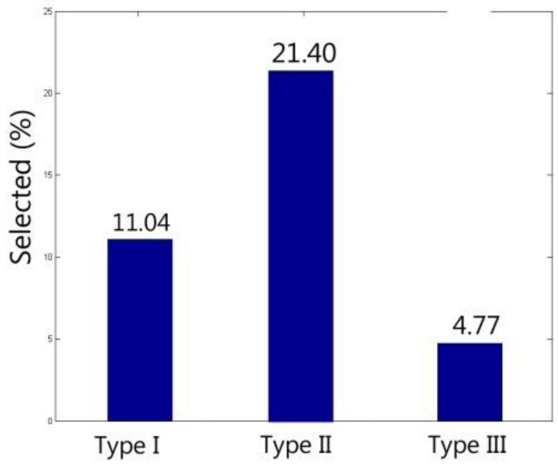


Figure 4.16: Ratio of selected features of each feature type on fall dataset

Chapter 5

Therapy Applications

5.1 Interactive Therapy Application (KinematEval)

We develop the KinematEval application that provides a tool which allows the users to record and recognize different human body motions and customized gestures using depth images as provided by Microsoft Kinect sensor. The underlying algorithm comprehends the interactions of the users by processing of the body joint's information of skeleton. Some notable properties of the software is that

- **Easy Evaluation process**

  The user can easily record new action and add it to the list of actions to be evaluated by the software
- **Gestures are almost person independent**

  Actions previously recorded by one person can be used by another person
- **Orientation and scale invariance**

  In order to be orientation and scale invariant, some precautions made provide in feature extraction phase
- **Speed independence**

  Speed invariance property of action enable the system to recognize actions with faster or slower speed thanks to dynamic time warping technique
- **Adjustable setting modes**

  User can control the flow of the application with different tools provided

The application is developed along with a research project at Sabanci University in order to test the proposed research methods in course of the studies. A variety of tools and libraries are used in development process such as OpenGL library for rendering the graphics and Microsoft MFC library for user interface part. The software has written in C++ and is based on Kinect SDK library.

The aim of KinematEval is twofold; first we want to recognize gestures and in the meantime analyze them from different aspects and second detect falling of a subject user. This is suitable for situation a patience collapse when an instant incident happens.

5.2 Installation and Requirements

To make the software run on your system, first of all you need to install Microsoft Kinect software development kit (SDK)[4]. Current version that we are developing upon is SDK version 1.7. You will also need to have OpenGL[5] on your system. In order to run the graphics properly your system's graphic hardware should be capable of running OpenGL 3 or later. You should have OpenGL's dynamic link library file "OpenGL32.dll" in your system folder or in the running path of the application.

You can find a detailed step by step installation instruction for OpenGL on your system in following link:

http://www.opengl.org/wiki/Getting_Started

And also you can find installation tips and download links of Microsoft Kinect SDK:

http://www.microsoft.com/en-us/kinectforwindows/develop/overview.aspx

5.3 Features and How to Use

This section gives you a brief overview on how to use the KinematEval and what are its features.

Used Feature: We extract and use 4 kinds of features for each joint in each frame to evaluate the actions. These features are angle features (Figure 5.1a), normal vector features (Fig.5.1b), velocity features (Figure 5.1c) and Euclidian distance features (Fig 5.1d) as you can see in the table below:

[4] http://www.microsoft.com/en-us/kinectforwindows/

[5] *www.**opengl**.org/*

θ	Type I features (Angle)
φ	
V_x	
V_y	
V_z	Type II features (Velocity)
d1	
d2	
d3	
.	Type III features (Distance)
.	
.	
d19	

Table 5.1: Features of joints

So for each joint we have 24 potential features (for some joints we don't have some of the features. For example for joints of hands we don't calculate angular features).

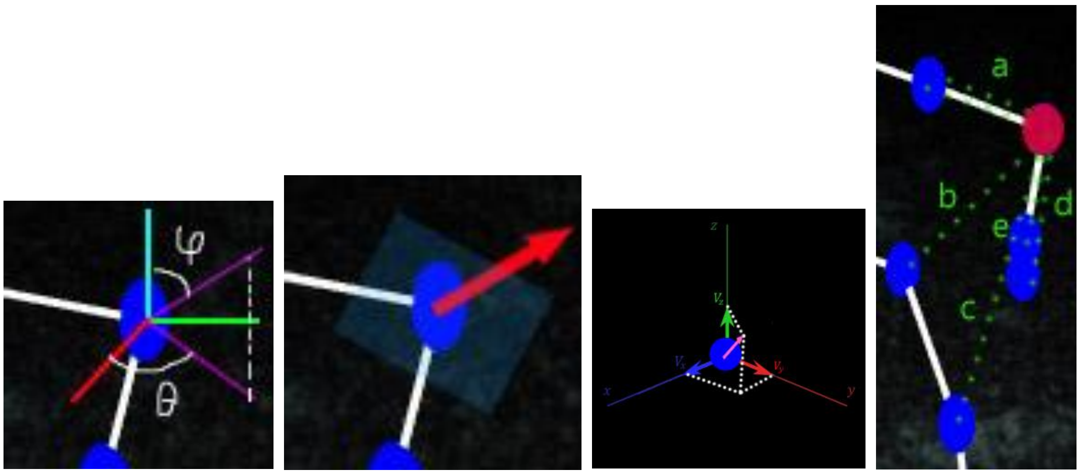


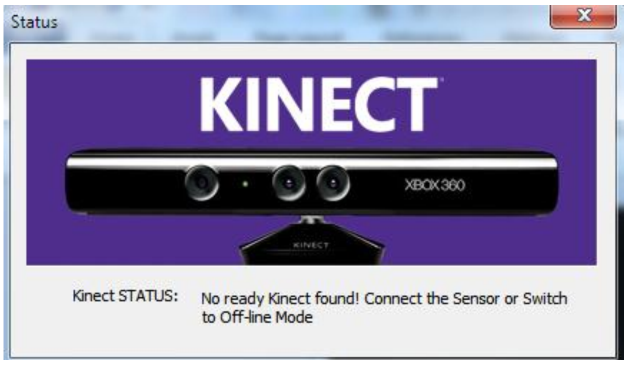
Figure 5.1: (a) Angle features (b) Normal features (c) Velocity features (d) Euclidian distance features

Evaluation of Actions: Now that you installed the application and all its requirements you are ready to run. After running the application first you have to be recognized by the system. KinematEval has two running mode; the **off-line** and the **on-line** mode. Therefore after running of the application there are two possibilities:

- If the Kinect sensor is not connected to the system to the system the application automatically switches into off-line mode
- If you connect the Kinect sensor to the system, the application switches back to on-line mode.

In off-line mode you can still evaluate and compare the default actions available in the application database with each other. In on-line mode a recorded or system default action get evaluate or compare to real-time user data streams directly from the sensor. The positive side of having this mode selection property is that when user or developer decides to test some feature in application, he or she is not obliged to stand in front of the sensor and fetch the data stream to use in application environment, instead s/he can use prerecorded ones. There is also a notification mechanism in the application to alert the user or developer about what status that the application or connected devices is.

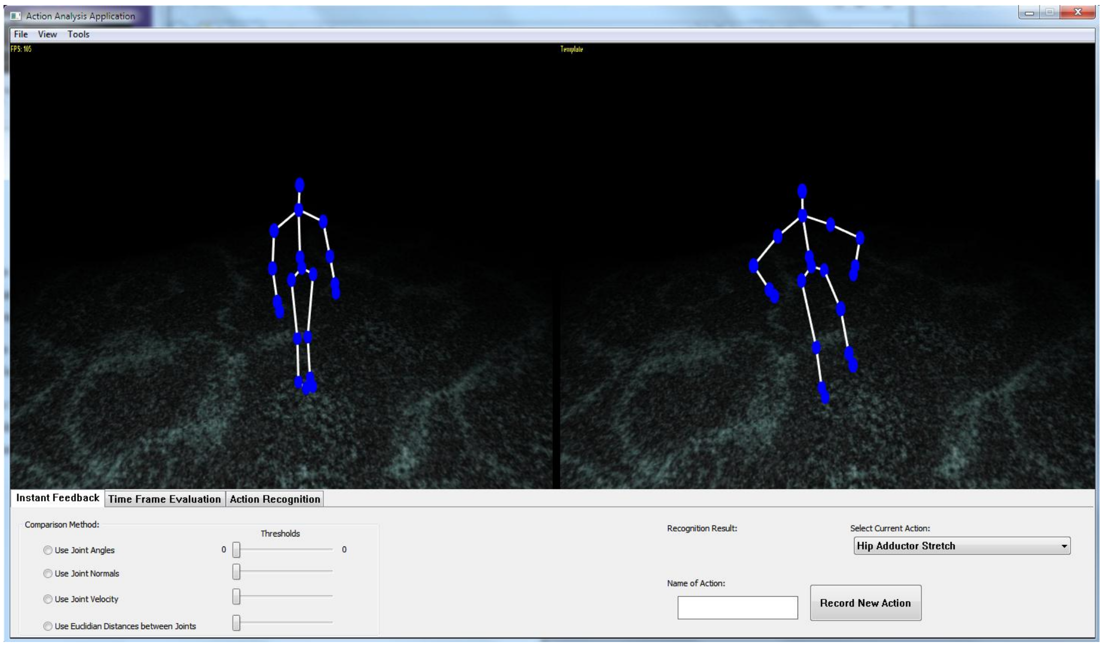
All these system notifications are notified to the user through a pop up window beside the main application window:



Kinect STATUS: No ready Kinect found! Connect the Sensor or Switch to Off-line Mode

Figure 5.2: Notification window

To start evaluation of the actions, first the user should go in front of the system so that the depth sensor could register the skeleton. Next, on-line or off-line evaluation begins.



Select Current Action: Hip Adductor Stretch

Figure 5.3: Initialized application window with registered skeleton

To evaluate the actions we provided three tabs for application's user interface. We want to analyze each action with three different methods. **Instant feedback**, **time frame evaluation** and **action recognition** are application's tabs:

Figure 5.4: Evaluation tabs

In the first method (instant feedback) we want to obtain instant status of each individual skeleton joint. We want to show (using color tunes) how much this individual joint of test skeleton is similar in action to a template action. That is, we calculate the similarity score of that particular joint with its corresponding joint in template action and render the joint in a particular hue (green shows highest similarity and red indicate least similar situations):

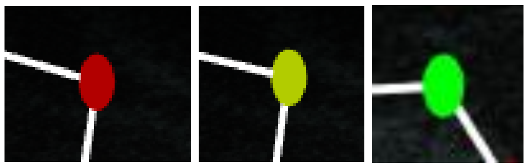


Figure 5.5: Similarity score representation through color saturations

You have several options here; you can change between the features described in section 3.1 to take it into account in evaluations. You can also set the threshold value for each of the feature options. This means for example in angle features, if the angular difference between template and real-time action is less than this threshold we consider this as almost similar joint status and show with green spectrum. In Figure 5.6 you can see this menu and its proposed options.

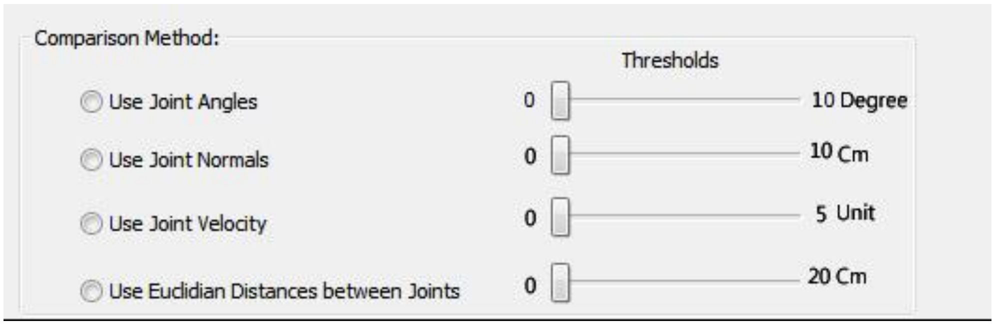
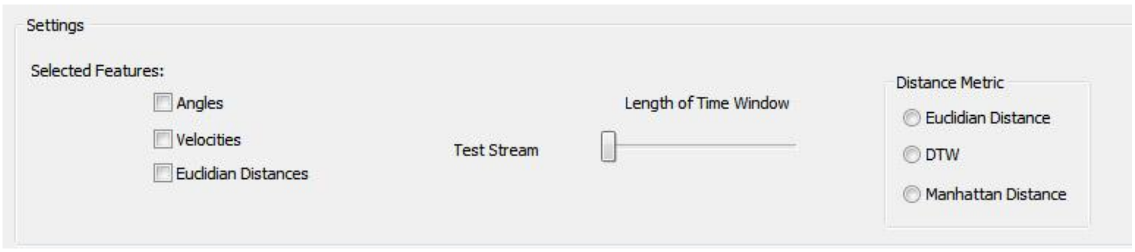


Figure 5.6: Instant feedback tab

Second tab is time frame evaluation of the actions, which is in each action user selects the window frame size in template and test action to be taken into account in evaluations. In first method we just used a single frame to compare the joints but here we consider a sequence of frames which its size is set by the user. First the used feature is set and next the length of the time window is adjusted. To compare the two frame sequences three distance metrics is provided as shown below:


Settings

Selected Features:

- [ ] Angles
- [ ] Velocities
- [ ] Euclidian Distances

Test Stream

Length of Time Window

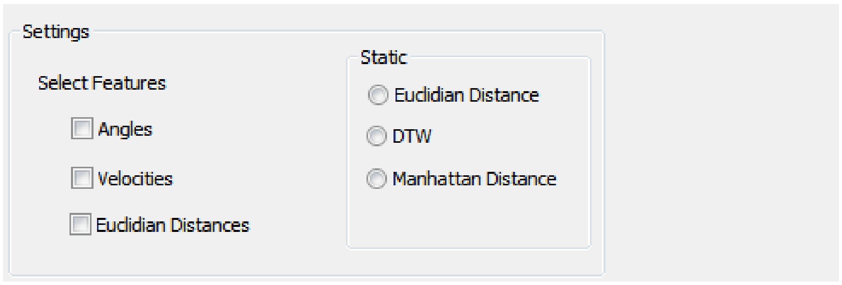
Distance Metric

- ○ Euclidian Distance
- ○ DTW
- ○ Manhattan Distance

Figure 5.7: Time frame evaluation tab

the user is free to choose between Euclidian distance, dynamic time warping and Manhattan distance as a metric to evaluate the two time-series.

The final tab is the action recognition tab which performs a complete recognition process using proposed algorithms. Again like the previous tab first the user chooses features to be used in evaluations and next s/he selects the recognition algorithm.


Settings

Select Features

- [ ] Angles
- [ ] Velocities
- [ ] Euclidian Distances

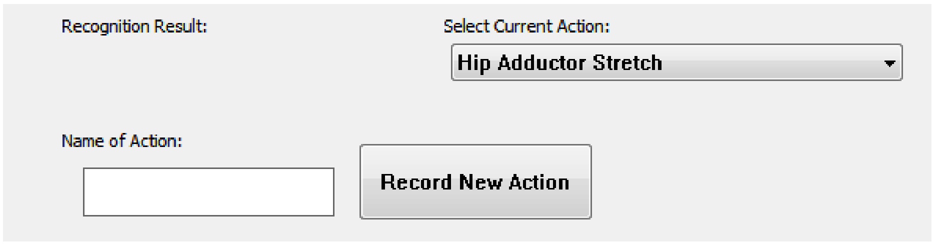
Static

- ○ Euclidian Distance
- ○ DTW
- ○ Manhattan Distance

Figure 5.8: Action recognition tab

In this tab user is free to choose between Euclidian distance, dynamic time warping and Manhattan distance as recognition algorithm. The recognized action will be shown to the user consequently.

Adding New Actions: In addition the software provides recording of new actions for the user. User by pushing the record button records a new action to evaluate. The recorded action instantly gets added to the action list and is ready to use.


Recognition Result:

Select Current Action: Hip Adductor Stretch

Name of Action:

Figure 5.9: Adding new actions

User must enter a name for the new action. If in offline mode, this option is disabled.

5.4 Fall Detection Mode

Furthermore we are interested to detect when a person fall on the ground. This fall detection is got serious especially while we are dealing with patients, which is very important to detect if s/he is fainted or not. To this aim, KinematEval is equipped with a fall specific mode in order to extract its particular features and distinguish its happening through skeleton data stream.

Chapter 6

Conclusions and Future Work

In this thesis we analyzed the human motion from different aspects. First, we tested and analyzed variety of kinematic features that are extracted from depth sensors. Using template matching approaches we obtained a baseline performance for our study. In the next level of the work we tried to improve upon this performance both on classification accuracy and feature selection efficiency.

We proposed a discriminative RDF-based feature selection framework capable of reaching impressive action recognition performance when combined with a linear SVM classifier. Our results showed state-of-the-art performances in action classification, beating for instance the 88.7% performance of Ellis et al.'s work [76] on MSRC-12. We obtained 94% average classification accuracy by using our synergetic classifier method on MSRC-12 and 98% average classification accuracy on WorkoutSU-10 dataset. The large, but possibly redundant set of invariant spatiotemporal features extracted from the skeleton in motion have been data-mined thanks to discriminative capabilities of RDF in order to reach comparable or even better performance with a significantly reduced number of features (one tenth of the original set).

Furthermore, we have introduced a novel therapeutic action recognition dataset to be prospectively used by the action recognition community. As an expansion to this dataset, we recorded a fall dataset to analyze fall detection. In order to do this we performed our same template matching and synergetic methods on this dataset and we obtained high average accuracy on this expansion of the WorkoutSU-10 dataset.

In addition, we developed an action evaluation and recognition application. The purpose here is to test our proposed methods in real-world problems. Here we should deal with real-time scenarios. The efficiency of the implemented algorithms is of utmost importance which we should cope with. It should be stressed that while our designed framework is planned to be used as part of a therapy application for treatment purposes, it has enough potentials to be applicable in other action recognition tasks such as assisted living, intelligent surveillance and games.

Here we discuss future direction of our work:

- Part of workload will be on improvement of the application we described in chapter 5. New challenges show up when one is trying to develop a real-time action recognition framework. Handling and alignment of time-series and also efficiency of the algorithms in real-time is the major part.
- We used DTW just as a similarity measure in baseline tests and action recognition in application. Different forms and features of DTW can be investigated and incorporated to produce more precise and efficient algorithms.
- Baseline tests were evaluated by using average score between two action's feature sets. This means, after comparison of time-series of two actions by DTW or NCC, the overall similarity score is computed by averaging between the all scores. Since averaging method is not a reliable method to estimate the overall score we can practice other methods to deal with this type of inaccuracy. As you see in Figure 6.1 one potential solution can rely on vote of individual time-series. Each time-series in each time point can vote that the test action is belongs to which class. Then, a two-step majority vote can be used for decide the final label of the test action (Figure 6.1 Method 1). The accuracy and validity of this idea can be tested and discussed.
- In addition, for baseline performance, clustering and partitioning methods like k-means, k-centroids or microclustering algorithms can be useful.
- Other principal focus of future work will be on fall detection problem. So far, we applied the same method we used for analyzing the normal actions of human body. But as we mentioned in chapter 2, fall detection needs special regard which we should put emphasis on. In recognition part we used kinematic features which in this particular case can be useful to some extent. We need to keep track of the subject after falling on the ground to assure the action has happened. But as you see in Figure 6.2 it is not possible to extract kinematic features from such representation of the skeleton. In this point we need to consider other types of features like image-based features or fall-specific features. Also we can use other source of data coming from sensors like depth

information of the scene or RGB images which we also have recorded in our dataset.

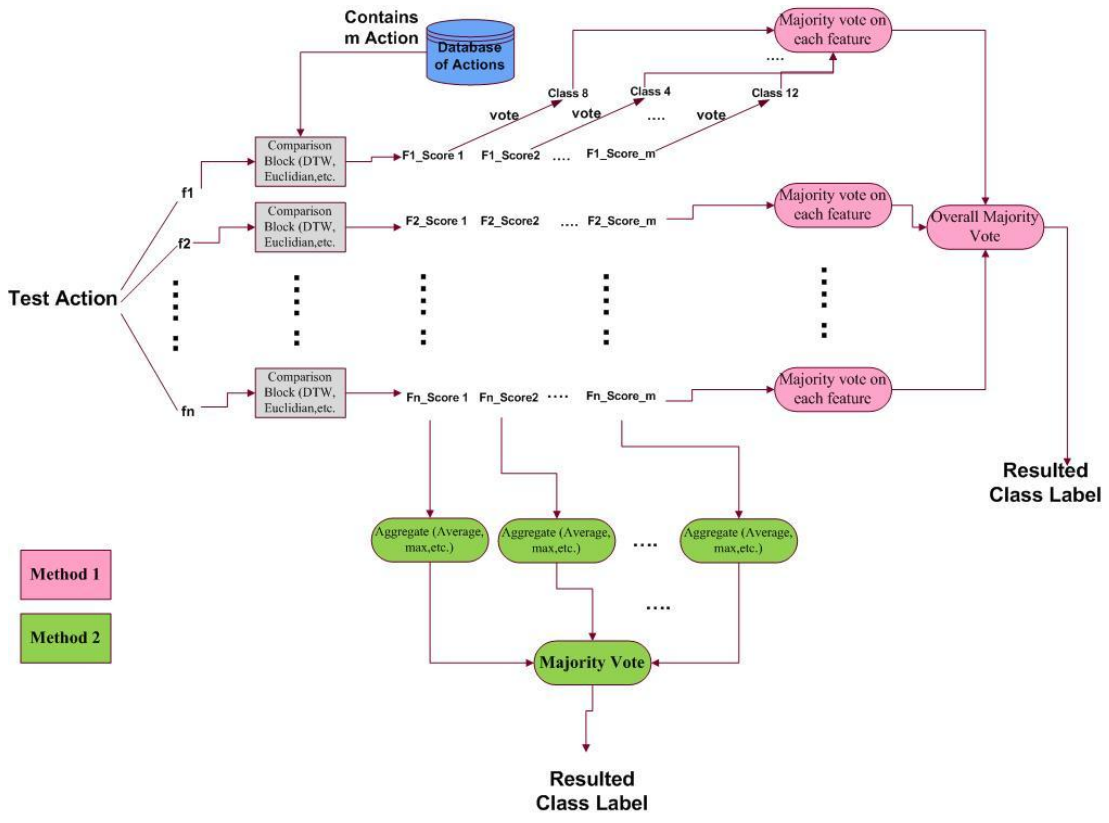


Figure 6.1: Block diagram of the testing baseline performance using new method

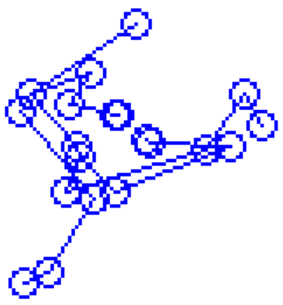


Figure 6.2: Inaccurate skeletal joints representation when the subject is falling

Appendix 1

Microsoft Kinect

In this work we used Microsoft Kinect [2], a depth sensor has been developed by Microsoft to extract depth and skeleton information from human body. In this section we present a brief overview about Kinect sensor and its underlying algorithm.

Microsoft Kinect sensor is an end-to-end solution which allows a computer to catch sight of the 3D world and to convert this grasped information into a depth image. This is a process analogous to what humans do.

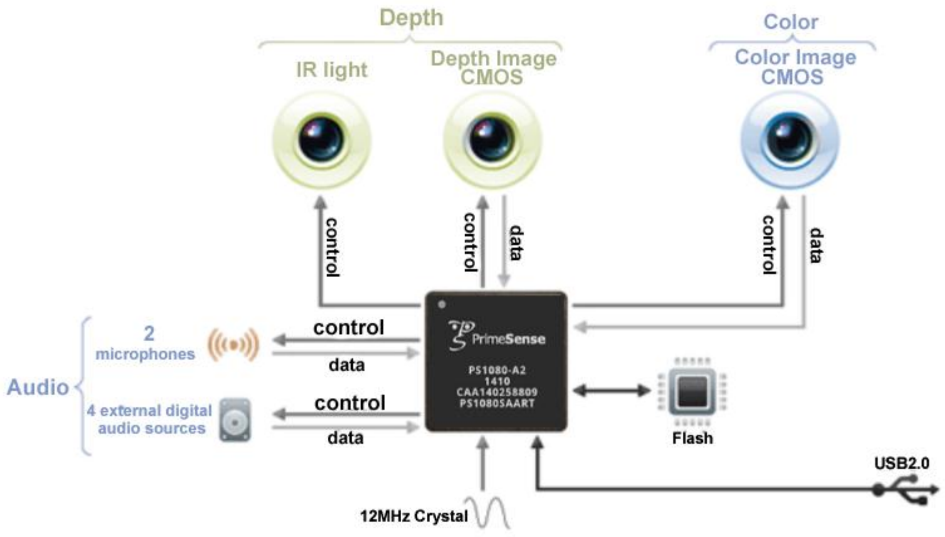


Figure a.1: The kinect sensor components.

A sensor component observes the scene (contains users and surroundings), and an underlying algorithm comprehends the interactions of the users. Kinect sensor for acquiring the depth image uses light coding. Light coding works by coding the scene volume with invisible near infra-red (IR) light. Kinect make use of a standard CMOS image sensor in order to read the coded light which reflected back from facing scene. The System on Chip (SoC) chip is connected to the CMOS image sensor, run an algorithm to decode the received light coding and extract a depth image of the scene. The sensor is secure against ambient light. The device also has two optional sensory input capabilities: One color (RGB) image and an audio input. A registration process performs in order to get more accurate information. That is every pixel in image is aligned with pixels in depth image. All of the extracted image, depth and audio information are transferred to the host via a USB2.0 interface in a synchronous fashion Figure a.1.

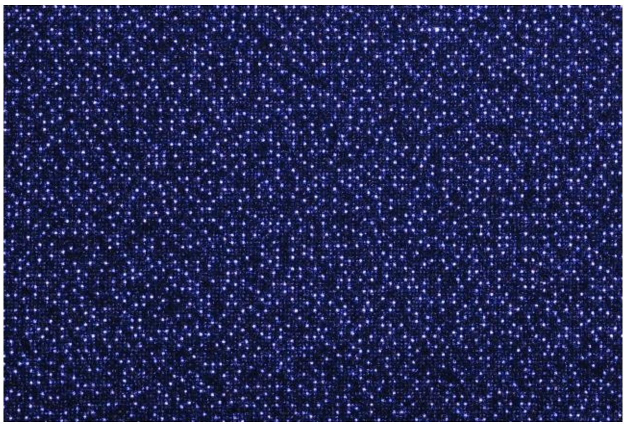


Figure a.2: IR pattern emitted by Kinect

The Kinect uses a method of mapping by projecting a speckle pattern of dots from an IR projector into a target region and detecting of reflected pattern by an IR camera [94] Figure a.2. A large number of reference images of the speckle pattern are hard coded into the Kinect at manufacture. Unique combination of dot patterns cause each infrared dot get identified when it projected onto the scene. Underlying calculation to find depth information in the scene is basically a depth calculation from stereo images **(**Figure a.3**)**.

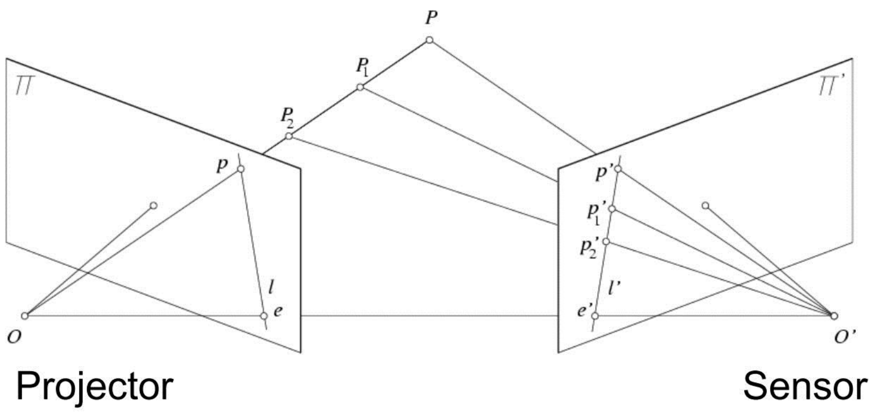


Figure a.3: Finding depth in kinect is similar to calculate depth from stereo images with the difference that in here there is one image and a reference pattern.

Each time the algorithm single out a specific dot in the reference pattern and looks through the observed for that dot. It also looks at each dot's eight surrounding dots because as we said there is a unique combination for each dot. When the dot found the disparity can be obtained:

$$disparity = x - x' = \frac{B.f}{z}$$

Considering the focal length of the IR camera and the baseline between the projector and the camera, depth of that particular dot can be calculated. Then we can repeat this process for each of the points in the reference image and acquire the complete depth image of the target scene (Figure a.4).

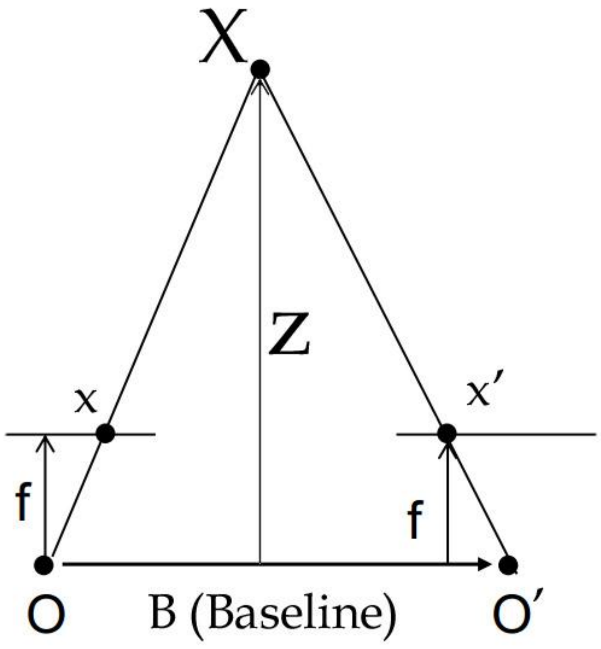


Figure a.4: Disparity is inversely proportional with depth value

For three different depth ranges IR projector casts speckles at different sizes. That is why Kinect can function in ranges approximately between 0.8 to 3.5 meters [95]. Kinect algorithm first identify where initially the object lie and then it just cast speckles which is optimized for extracting depth for that particular range. This feature of Kinect to project speckles with different sizes is a substantial benefit which enables Kinect to perform on a wide range just with a single scan at a time. Another important design feature of the Kinect is that other than calculation of pixel shifts in order to identify depth, it also compare the size of dots in the target scene with the size of dots in the reference pattern. If any change catch in size and shape, it reflects in depth estimation calculations. Kinect does all of these calculations in real-time on SOC and the result is a depth image of 640×480 with 30fps frame rate.

Body tracking in Microsoft Kinect

So far we gave a description of random decision forests and also the way Kinect depth sensing device operate. Now we describe the application of random decision forest for real-time tracking of human body used in Microsoft Kinect.

What we want to achieve here is with make use of a given depth image to ascertain each of the pixels belong to which part of the human body. This kind of classification is mainstream for a classification forests. This solution proposed in [1]. There are 31 different body part classes:

$c \in \{head, right\ hand, left\ hand, right\ shoulder, left\ shoulder, right\ elbow, \dots\}$. The computation unit in the 2D depth image is a pixel $p \in R^2$. A feature vector $v(p)$ is extracted for every single pixel. In test phase, given a pixel of an unseen image we want to estimate the posterior probability $p(c|v)$ of that pixel. Features of a pixel are simply depth difference of couples of neighbor pixels. That is, for a single pixel p each of the elements of the feature vector $v = (x_1, x_2, \dots, x_i, \dots, x_d) \in R^d$ is defined as:

$$x_i = J(p) - J(p + \frac{r_i}{J(p)})$$

Where function Jsignify depth of a pixel which is its distance from camera plane in millimeters. Vector r_i represents a displacement from the reference pixel p. $d = \infty$ since there are infinite number of displacements around p we can look for.

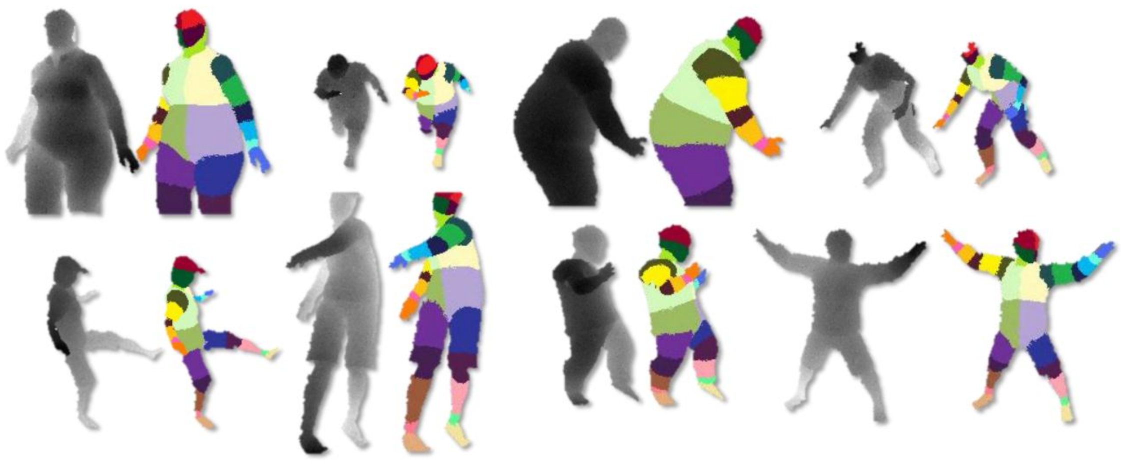


Figure a.5: Ground-truth labeling of body parts

During training a very large pixelwised labeled images as in Figure a.5 used to train the system. In training the algorithm tries to maximize the information gain where for a split node j parameters of the objective function are:

$$\theta_j = (r_j, \tau_j)$$

Where r_j is a randomly chosen displacement and τ_j is a threshold.

For weak learner, an axis-aligned weak learner is used. The split function in each node is like follow:

$$h(v, \theta_j) = [\phi(v, r_j) > \tau_j].$$

The selection vector $\boldsymbol{\phi}$ takes the entire feature vector and returns a feature in association with the selected displacement vector r_j. In reality when $d = \infty$ the possible set of split parameters will be also infinite. But in practice when we want to train a node, we randomly generate a parameter set for that particular node and using exhaustive search try to maximize the information gain for that node and we don't need to compute entire infinite set [17]. An example result of applying this algorithm is shown in Figure a.6.

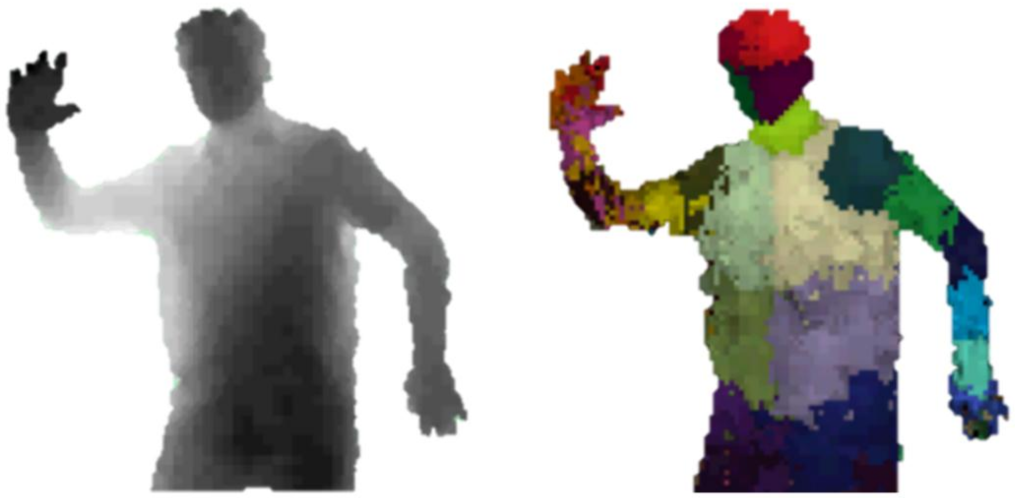


Figure a.6: Left) depth image with removed background Right) the posterior classification of 31 body parts, each color correspond to different body parts [1]

Bibliography

1. Shotton, J., et al. *Real-time human pose recognition in parts from single depth images*. in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. 2011. IEEE.
2. Microsoft, *Microsoft Corp. Redmond WA. Kinect for Xbox 360.*
3. Asus, *http://www.asus.com/Multimedia/Xtion_PRO_LIVE/.*
4. Arkin, S.M., *Student-led exercise sessions yield significant fitness gains for Alzheimer's patients.* American journal of Alzheimer's disease and other dementias, 2003. **18**(3): p. 159-170.
5. Shaughnessy, M., B.M. Resnick, and R.F. Macko, *Testing a Model of Post-Stroke Exercise Behavior.* Rehabilitation Nursing, 2006. **31**(1): p. 15-21.
6. Rougier, C. and J. Meunier, *Demo: Fall detection using 3D head trajectory extracted from a single camera video sequence.* Journal of Telemedicine and Telecare, 2005. **11**(4).
7. Sivalingam, R., et al. *A multi-sensor visual tracking system for behavior monitoring of at-risk children*. in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. 2012. IEEE.
8. Cadoz, C., *Les réalités virtuelles: un exposé pour comprendre, un essai pour réfléchir*. 1994: Flammarion.
9. Space, K., *http://code.google.com/p/kineticspace/.*
10. Hansard, M., et al., *Time-of-Flight Cameras*. 2013: Springer.
11. Peng, T. and S.K. Gupta, *Model and algorithms for point cloud construction using digital projection patterns.* Journal of Computing and Information Science in Engineering, 2007. **7**(4): p. 372.
12. Kinemote, *http://www.kinemote.net/.*
13. Raptis, M., D. Kirovski, and H. Hoppe. *Real-time classification of dance gestures from skeleton animation*. in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 2011. ACM.
14. nike, *http://www.nike.com/us/en_us/c/training/nike-plus-kinect-training.*
15. Kamel Boulos, M.N., *Xbox 360 Kinect Exergames for Health.* GAMES FOR HEALTH: Research, Development, and Clinical Applications, 2012.
16. Yao, A., et al. *Does human action recognition benefit from pose estimation?"*. in *Proceedings of the 22nd British machine vision conference-BMVC 2011*. 2011.
17. Criminisi, A., *Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning.* Foundations and Trends® in Computer Graphics and Vision, 2011. **7**(2-3): p. 81-227.
18. Amit, Y. and D. Geman, *Shape quantization and recognition with randomized trees.* Neural computation, 1997. **9**(7): p. 1545-1588.
19. Fothergill, S., et al. *Instructing people for training gestural interactive systems*. in *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*. 2012. ACM.
20. Erol, A., et al., *Vision-based hand pose estimation: A review.* Computer Vision and Image Understanding, 2007. **108**(1): p. 52-73.

21. Zhao, W., et al., *Face recognition: A literature survey.* Acm Computing Surveys (CSUR), 2003. **35**(4): p. 399-458.
22. Hu, W., et al., *A survey on visual surveillance of object motion and behaviors.* Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 2004. **34**(3): p. 334-352.
23. Weinland, D., R. Ronfard, and E. Boyer, *A survey of vision-based methods for action representation, segmentation and recognition.* Computer Vision and Image Understanding, 2011. **115**(2): p. 224-241.
24. Poppe, R., *Vision-based human motion analysis: An overview.* Computer Vision and Image Understanding, 2007. **108**(1): p. 4-18.
25. Masoud, O. and N. Papanikolopoulos, *A method for human action recognition.* Image and Vision Computing, 2003. **21**(8): p. 729-743.
26. Ji, X. and H. Liu, *Advances in view-invariant human motion analysis: A review.* Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 2010. **40**(1): p. 13-24.
27. Viola, P. and M.J. Jones, *Robust real-time face detection.* International journal of computer vision, 2004. **57**(2): p. 137-154.
28. Lo, B. and S. Velastin. *Automatic congestion detection system for underground platforms*. in *Intelligent Multimedia, Video and Speech Processing, 2001. Proceedings of 2001 International Symposium on*. 2001. IEEE.
29. Bradski, G.R. and J.W. Davis, *Motion segmentation and pose recognition with motion history gradients.* Machine Vision and Applications, 2002. **13**(3): p. 174-184.
30. Bogomolov, Y., et al. *Classification of moving targets based on motion and appearance*. in *British Machine Vision Conference*. 2003. Norwich, UK.
31. Min, S., et al. *Graph-Cut Based Background Subtraction Using Visual Hull in Multiveiw Images*. in *Computing: Techniques and Applications, 2008. DICTA'08. Digital Image*. 2008. IEEE.
32. Huang, Y. and T.S. Huang. *Model-based human body tracking*. in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*. 2002. IEEE.
33. Guo, Y., G. Xu, and S. Tsuji. *Understanding human motion patterns*. in *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*. 1994. IEEE.
34. Rohr, K., *Towards model-based recognition of human movements in image sequences.* CVGIP-Image Understanding, 1994. **59**(1): p. 94-115.
35. O'Rourke, J. and N.I. Badler, *Model-based image analysis of human motion using constraint propagation.* IEEE TRANS. PATTERN ANALY. AND MACH. INTELLIG., 1980. **2**(6): p. 522-536.
36. Carranza, J., et al., *Free-viewpoint video of human actors.* ACM Transactions on Graphics (TOG), 2003. **22**(3): p. 569-577.
37. Perales, F.J. and J. Torres. *A system for human motion matching between synthetic and real images based on a biomechanic graphical model*. in *Motion of Non-Rigid and Articulated Objects, 1994., Proceedings of the 1994 IEEE Workshop on*. 1994. IEEE.
38. Webb, J.A. and J. Aggarwal, *Visually interpreting the motion of objects in space*. 1981: Computer Science Department, University of Texas at Austin.
39. Kehl, R. and L.V. Gool, *Markerless tracking of complex human motions from multiple views.* Computer Vision and Image Understanding, 2006. **104**(2): p. 190-209.

40. Ramanan, D.K. and D. Forsyth, *Automatic annotation of everyday movements*. 2003: Computer Science Division, University of California.
41. Agarwal, A. and B. Triggs, *Recovering 3D human pose from monocular images.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2006. **28**(1): p. 44-58.
42. Bottino, A. and A. Laurentini, *A silhouette based technique for the reconstruction of human movement.* Computer Vision and Image Understanding, 2001. **83**(1): p. 79-95.
43. Sidenbladh, H. and M.J. Black, *Learning the statistics of people in images and video.* International Journal of Computer Vision, 2003. **54**(1): p. 183-209.
44. Cheung, K., S. Baker, and T. Kanade. *Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture*. in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. 2003. IEEE.
45. Yamato, J., J. Ohya, and K. Ishii. *Recognizing human action in time-sequential images using hidden Markov model*. in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*. 1992. IEEE.
46. Wang, L. and D. Suter. *Recognizing human activities from silhouettes: Motion subspace and factorial discriminative graphical model*. in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. 2007. IEEE.
47. Rittscher, J. and A. Blake. *Classification of human body motion*. in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. 1999. IEEE.
48. Laptev, I., *On space-time interest points.* International Journal of Computer Vision, 2005. **64**(2): p. 107-123.
49. Blank, M., et al. *Actions as space-time shapes*. in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. 2005. IEEE.
50. Rabiner, L.R., *A tutorial on hidden Markov models and selected applications in speech recognition.* Proceedings of the IEEE, 1989. **77**(2): p. 257-286.
51. Gavrila, D. and L. Davis. *Towards 3-d model-based tracking and recognition of human movement: a multi-view approach*. in *International workshop on automatic face-and gesture-recognition*. 1995. Citeseer.
52. Wang, Y., et al. *Unsupervised discovery of action classes*. in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. 2006. IEEE.
53. Kovar, L. and M. Gleicher. *Automated extraction and parameterization of motions in large data sets*. in *ACM Transactions on Graphics (TOG)*. 2004. ACM.
54. Hernández, J., et al., *Human activity recognition based on kinematic features.* Expert Systems, 2013.
55. Ali, S. and M. Shah, *Human action recognition in videos using kinematic features and multiple instance learning.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2010. **32**(2): p. 288-303.
56. Healey, F., et al., *Falls in English and Welsh hospitals: a national observational study based on retrospective analysis of 12 months of patient safety incident reports.* Quality and Safety in Health Care, 2008. **17**(6): p. 424-430.
57. Oldies, D.P.m.l.t.t.a.T., *Thaindian News, 18 June 2008.*

58. Fu, Z., et al. *Fall detection using an address-event temporal contrast vision sensor*. in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*. 2008. IEEE.
59. Foroughi, H., et al. *An eigenspace-based approach for human fall detection using integrated time motion image and neural network*. in *Signal Processing, 2008. ICSP 2008. 9th International Conference on*. 2008. IEEE.
60. Miaou, S.-G., P.-H. Sung, and C.-Y. Huang. *A customized human fall detection system using omni-camera images and personal information*. in *Distributed Diagnosis and Home Healthcare, 2006. D2H2. 1st Transdisciplinary Conference on*. 2006. IEEE.
61. Rougier, C., et al., *Robust video surveillance for fall detection based on human shape deformation.* Circuits and Systems for Video Technology, IEEE Transactions on, 2011. **21**(5): p. 611-622.
62. Cucchiara, R., et al., *Probabilistic posture classification for human-behavior analysis.* Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 2005. **35**(1): p. 42-54.
63. Thome, N. and S. Miguet. *A HHMM-based approach for robust fall detection*. in *Control, Automation, Robotics and Vision, 2006. ICARCV'06. 9th International Conference on*. 2006. IEEE.
64. Jansen, B. and R. Deklerck. *Context aware inactivity recognition for visual fall detection*. in *Pervasive Health Conference and Workshops, 2006*. 2006. IEEE.
65. Johansson, G., *Visual perception of biological motion and a model for its analysis.* Perception & psychophysics, 1973. **14**(2): p. 201-211.
66. Moeslund, T.B. and E. Granum, *A survey of computer vision-based human motion capture.* Computer Vision and Image Understanding, 2001. **81**(3): p. 231-268.
67. Moeslund, T.B., *Interacting with a virtual world through motion capture*, in *Virtual interaction: interaction in virtual inhabited 3D worlds*. 2001, Springer. p. 221-234.
68. Andersen, C., *A Survey of Gloves for Interaction with Virtual Worlds, Technicalreport.* LaboratoryofImageAnalysis,AalborgUniversity,
Denmark.
69. Essid, S., et al. *An advanced virtual dance performance evaluator*. in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. 2012. IEEE.
70. Alexiadis, D.S., et al. *Evaluating a dancer's performance using kinect-based skeleton tracking*. in *Proceedings of the 19th ACM international conference on Multimedia*. 2011. ACM.
71. Bianco, S. and F. Tisato. *Karate moves recognition from skeletal motion*. in *IS&T/SPIE Electronic Imaging*. 2013. International Society for Optics and Photonics.
72. Ju, S.X., M.J. Black, and Y. Yacoob. *Cardboard people: A parameterized model of articulated image motion*. in *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*. 1996. IEEE.
73. Kakadiaris, L. and D. Metaxas, *Model-based estimation of 3D human motion.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2000. **22**(12): p. 1453-1459.
74. Ramanan, D. and D.A. Forsyth. *Finding and tracking people from the bottom up*. in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. 2003. IEEE.

75. Grest, D., J. Woetzel, and R. Koch, *Nonlinear body pose estimation from depth images*. Pattern Recognition, 2005: p. 285-292.
76. Ellis, C., et al., *Exploring the Trade-off Between Accuracy and Observational Latency in Action Recognition.* International Journal of Computer Vision, 2012: p. 1-17.
77. Müller, M., T. Röder, and M. Clausen. *Efficient content-based retrieval of motion capture data*. in *ACM Transactions on Graphics (TOG)*. 2005. ACM.
78. Murray, C.D., et al., *The treatment of phantom limb pain using immersive virtual reality: three case studies.* Disability & Rehabilitation, 2007. **29**(18): p. 1465-1469.
79. Lange, B., et al. *Development and evaluation of low cost game-based balance rehabilitation tool using the Microsoft Kinect sensor*. in *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*. 2011. IEEE.
80. Chang, Y.-J., S.-F. Chen, and J.-D. Huang, *A Kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities.* Research in developmental disabilities, 2011. **32**(6): p. 2566-2570.
81. WAR, *http://www.wardellortho.com/.*
82. Pyle, D., *Data preparation for data mining*. Vol. 1. 1999: Morgan Kaufmann.
83. Breiman, L., *Random forests.* Machine learning, 2001. **45**(1): p. 5-32.
84. WorloutSU-10, *http://vpa2.sabanciuniv.edu/databases/WorkoutSU-10/.* 2013.
85. Microsoft-kinect-documentation-May-2012-SDK-Rlease, *http://msdn.microsoft.com/en-us/library/hh855347.asp.*
86. Pham, C., T. Plötz, and P. Olivier, *A dynamic time warping approach to real-time activity recognition for food preparation*, in *Ambient Intelligence*. 2010, Springer. p. 21-30.
87. Sakoe, H. and S. Chiba, *Dynamic programming algorithm optimization for spoken word recognition.* Acoustics, Speech and Signal Processing, IEEE Transactions on, 1978. **26**(1): p. 43-49.
88. Ho, T.K. *Random decision forests*. in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*. 1995. IEEE.
89. Ho, T.K., *The random subspace method for constructing decision forests.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1998. **20**(8): p. 832-844.
90. Sammut, C. and G.I. Webb, *Encyclopedia of machine learning*. 2011: Springer-Verlag New York Incorporated.
91. Breiman, L., *Bagging predictors.* Machine learning, 1996. **24**(2): p. 123-140.
92. Kullback, S. and R.A. Leibler, *On information and sufficiency.* The Annals of Mathematical Statistics, 1951. **22**(1): p. 79-86.
93. Gini, C., *Concentration and dependency ratios.* Rivista di Politica Economica, 1997. **87**: p. 769-792.
94. Garcia, J. and Z. Zalevsky, *Range mapping using speckle decorrelation*, 2008, Google Patents.
95. Andersen, M., et al., *Kinect depth sensor evaluation for computer vision applications*, 2012, Technical report ECETR-6, Department of Engineering, Aarhus University (Denmark).