COMPUTER VISION-BASED HUMAN ACTION RECOGNITION VIA KEYPOINT TRACKING

by

Yunus Emre Kara

B.S., in Mathematics, Boğaziçi University, 2008

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

Graduate Program in Computer Engineering

Boğaziçi University

2011

COMPUTER VISION-BASED HUMAN ACTION RECOGNITION VIA KEYPOINT TRACKING

APPROVED BY:

Prof. Lale Akarun .....
(Thesis Supervisor)

Prof. Cem Ersoy .....

Dr. Hülya Yalçın .....

DATE OF APPROVAL: 17.01.2011

ACKNOWLEDGEMENTS

First and foremost, I would like to offer my deepest gratitude to my thesis advisor Prof. Lale Akarun. Without her guidance and patience, this thesis would not be complete.

I also would like to thank my friends and colleagues in MediaLab and PILAB, namely, Alp Kındıroğlu, Cem Keskin, Furkan Kıraç, Hamdi Dibeklioğlu, Heysem Kaya, İsmail Arı, Koray Balcı, Neşe Alyüz, Onur Dikmen, Pınar Sağlam, Pınar Santemiz, and Umut Konur for providing a fun working environment and keeping up with me.

I thank all my friends in the computer engineering department that have helped me raise my spirits in times of bad. My thanks specifically go to Birkan Yılmaz, Can Komar, Çetin "the Dayı" Meriçli, Itır Karaç, İlker Ulutaş, Ozan Özen "the popstar", Remzi Yavuz, Salim Eryiğit, Serhan Daniş, Şükrü Kuran, and Tekin Meriçli.

I would also like to thank my friends Erinç Dikici, Mehmet Ekinci, Uluç Pamuk, and Yusuf Gören for their valuable friendship and continuous support.

I would like to offer my deepest gratitude to my parents Nevin Kara and İsmail Hakkı Kara, and my sister Efser Kara. Without the support and encouragement of my family throughout my educational life I would not have been the person that I am today.

Lastly, I would like to express my deepest gratitude to Gaye Genç who has always been there in times of need. Without her support, motivation, and patience, I could not bear the burden.

This research is supported by the Scientific and Technical Research Council of Turkey (TÜBİTAK) under grant numbers 108E161 and 108E207.

ABSTRACT

COMPUTER VISION-BASED HUMAN ACTION RECOGNITION VIA KEYPOINT TRACKING

Computer vision-based human action recognition is a highly active research area which has many application areas including security, surveillance, assisted living, and entertainment. In this thesis, a new system for computer vision-based recognition of human actions is presented. The proposed system uses videos as input. The approach is invariant of the location of the action and zoom levels, the appearance of the person, partial occlusions including self-occlusions and some viewpoint changes. It is robust against temporal length variations. Keypoints are tracked through time and the trajectories of tracked keypoints are used for interpreting the human action in the video. Then, features from videos are extracted. A group of features for describing a trajectory are proposed. Trajectories are clustered using these trajectory features. The clustered trajectories are used for describing an image sequence. Image sequence descriptors are the normalized histograms of the clusters of trajectories. At the final stage, the proposed system uses the descriptors of the image sequences in a supervised learning approach. An application based on the proposed method has been developed and applied to various datasets. A new multi modal dataset, called WeCare, which is focused on elderly care systems is introduced. The main objective of the dataset is to detect falls of humans. For attaining this goal, some other actions that can be confused with the falling action are included in the dataset. The evaluation of the proposed approach is done using two datasets: KTH Human Action Dataset and URADL Dataset. The proposed technique performs comparable to the methods in the literature. It has 87.25 per cent accuracy on the KTH dataset, 88 per cent accuracy on the URADL dataset. It has an accuracy of 98.75 per cent on the WeCare dataset.

ÖZET

ANAHTAR NOKTA TAKİBİ İLE BİLGİSAYARLA GÖRME TEMELLİ İNSAN HAREKETİ TANIMA

Bilgisayarla görme temelli insan hareketi tanıma, güvenlik, gözetim, destekli yaşam ve eğlence gibi bir çok alanda uygulaması olan çok aktif bir araştırma konusudur. Bu tezde, bilgisayarlı görme temelli insan hareketi tanıma için yeni bir sistem sunulmaktadır. Önerilen sistem girdi olarak videoları kullanmaktadır. Yaklaşım, hareketin konumuna, ölçek seviyelerine, kişinin görünümüne, kendini örtmeler de dahil olmak üzere kısmi örtmelere ve bir takım görüş açısı değişikliklerine karşı değişimsizdir. Zamansal uzunluk değişimlerine karşı gürbüzdür. Anahtar noktalar zaman boyunca takip edilmektedir ve takip edilen anahtar noktaların gezingeleri videodaki insan hareketini yorumlamak için kullanılmaktadır. Ardından, videolardan öznitelikler çıkarılmaktadır. Gezingeyi tanımlamak için bir grup öznitelik önerilmektedir. Gezingeler, bu gezinge öznitelikleri kullanılarak öbeklenmektedir. Öbeklenen gezingeler bir imge dizisini tanımlamak için kullanılmaktadır. Imge dizisi tanımlayıcıları, gezinge öbeklerinin düzgelenmiş histogramlarıdır. Son aşamada, önerilen sistem, imge dizilerinin tanımlayıcılarını bir güdümlü öğrenme yönteminde kullanır. Önerilen yönteme dayalı bir uygulama geliştirilmiştir ve çeşitli veri kümelerine uygulanmıştır. WeCare adını verdiğimiz yaşlı bakım sistemleri odaklı yeni bir çok kipli veri kümesi sunulmuştur. Veri kümesinin asıl amacı insan düşmelerinin saptanmasıdır. Bu amaca ulaşmak için düşme hareketiyle karıştırılabilecek bazı başka hareketler de veri kümesine dahil edilmiştir. Önerilen yöntem KTH İnsan Hareketi Veri Kümesi ve URADL Veri Kümesi adlı iki ilave veri kümesi ile de değerlendirilmiştir. Önerilen yöntem yazındaki yöntemlerle kıyaslanabilir başarımdadır. KTH veri kümesinde yüzde 87,25 ve URADL veri kümesinde yüzde 88 hatasızlık başarımına sahiptir. WeCare veri kümesinde hatasızlığı yüzde 98,75'tir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . xi
LIST OF SYMBOLS/ABBREVIATIONS . . . xiii
1. INTRODUCTION . . . 1
1.1. Literature Review . . . 3
1.1.1. Global representations . . . 3
1.1.2. Local representations . . . 5
1.1.3. Model Fitting approaches . . . 7
1.1.3.1. Human detection . . . 7
1.1.3.2. Body tracking . . . 7
1.1.3.3. Model fitting . . . 8
1.2. Outline of the Thesis . . . 10
2. TECHNICAL BACKGROUND . . . 12
2.1. Keypoint Detection and Matching . . . 12
2.1.1. Feature Detection . . . 12
2.1.2. Descriptor Extraction . . . 12
2.1.3. Keypoint Matching . . . 13
2.1.4. Speeded-up Robust Features (SURF) . . . 13
2.1.4.1. SURF interest point detection . . . 13
2.1.4.2. SURF interest point description . . . 17
2.2. Clustering . . . 20
2.2.1. K-Means . . . 20
2.2.2. Bag-of-Words . . . 21
2.3. Supervised Learning . . . 22
2.3.1. K-Nearest Neighbor (k-NN) . . . 22
2.3.2. Support Vector Machines (SVM) . . . 22

2.4. KLT Feature Tracker 26
2.5. Space-Time Interest Points (STIP) 30
3. ACTION RECOGNITION VIA KEYPOINT TRACKING 33
3.1. Outline of the System 33
3.2. Generic Keypoint Tracker 34
3.2.1. Outline of the Generic Keypoint Tracker Algorithm 35
3.2.2. Keypoint-Trajectory Matching 37
3.2.3. Trajectory Updating 42
3.2.4. Elimination and Storing 43
3.3. Feature Extraction 46
3.3.1. Extracting Features from Video Sequences 46
3.3.2. Normalizing Against Time 47
3.3.3. Normalizing Against Spatial Position 47
3.3.4. Extracting Sub-trajectories 49
3.3.5. Trajectory Feature Extraction 50
3.3.6. Bag-of-Trajectories 54
3.3.7. Classification 55
4. EXPERIMENTS 56
4.1. Datasets 56
4.1.1. KTH Human Action Dataset 56
4.1.2. University of Rochester Activities of Daily Living Dataset . . . 58
4.1.3. WeCare Dataset 60
4.2. Experiment Setup 63
4.2.1. Training, Validation and Test sets 64
4.2.2. Parameters 64
4.3. Validation and Parameter Selection 65
4.3.1. Effects of the Keypoint Tracker Parameters 67
4.3.2. Effects of the Learning Parameters 69
4.4. Recognition results on the test sets 75
5. CONCLUSIONS 79
APPENDIX A: TABLES OF THE VALIDATION RESULTS 82
REFERENCES 90

LIST OF FIGURES

Figure 2.1. Calculating the sum of the values in a rectangular region using the integral image [1] 14

Figure 2.2. Filters used in SURF 16

Figure 2.3. Filters in x and xy directions for the scale levels 9 x 9 and 15 x 15 [1] 17

Figure 2.4. Haar wavelet filters in x and y directions [1] 18

Figure 2.5. A sliding orientation window of size $\frac{\pi}{3}$ [1] 18

Figure 2.6. SURF descriptor calculation 19

Figure 2.7. Change in descriptors 20

Figure 3.1. Block diagram of the system 33

Figure 3.2. Outline of the Generic Keypoint Tracker 36

Figure 3.3. Falling action trajectory samples from the WeCare dataset 38

Figure 3.4. Trajectory Bounding Box 44

Figure 3.5. Flow of Feature Extraction Processes 46

Figure 3.6. Coordinate System for Trajectory Normalization 49

Figure 3.7. Sub-trajectories 50

Figure 3.8. L_i, L_N, E_i with trajectory keypoints 51

Figure 4.1. Sample frames from the KTH dataset [2] 57

Figure 4.2. Sample frames from the URADL dataset [3] 59

Figure 4.3. Sample frames from the WeCare dataset [4] 62

Figure 4.4. The effect of ν and σ to accuracy 68

Figure 4.5. The effect of σ and ν to accuracy 68

Figure 4.6. The effect of cluster count K and σ to accuracy 69

Figure 4.7. The effect of ρ to accuracy 69

Figure 4.8. The effect of cluster count K and k-NN k value to accuracy 70

Figure 4.9. The effect of cluster count K and sub-trajectory length ω to accuracy 71

Figure 4.10. The effect of cluster count K and normalization to accuracy . . . 71

Figure 4.11. The effect of sub-trajectory length ω and normalization to accuracy 72

Figure 4.12. The effect of cluster count K and SVM cost C to accuracy 72

Figure 4.13. The effect of k-NN k value and sub-trajectory length ω to accuracy 73

Figure 4.14. The effect of k-NN k value and normalization to accuracy 73

Figure 4.15. The effect of SVM cost C and sub-trajectory length ω to accuracy 74

Figure 4.16. The effect of SVM cost C and normalization to accuracy 74

LIST OF TABLES

Table 4.1. Confusion matrix 63

Table 4.2. Validation parameters 66

Table 4.3. Confusion matrix of the results on the test set of the KTH dataset 75

Table 4.4. Confusion matrix of the results on the URADL dataset 77

Table 4.5. Confusion matrix of the results on the test set of the WeCare dataset 78

Table A.1. Change of validation results relative to ν and σ 82

Table A.2. Change of validation results relative to σ and ν 82

Table A.3. Change of validation results relative to K and σ 83

Table A.4. Change of validation results relative to ρ 83

Table A.5. Change of validation results relative to K and k 83

Table A.6. Change of validation results relative to K and ω 84

Table A.7. Change of validation results relative to K and normalization . . . 84

Table A.8. Change of validation results relative to k and ω 85

Table A.9. Change of validation results relative to K and C 86

Table A.10. Change of validation results relative to k and normalization 87

Table A.11. Change of validation results relative to ω and normalization 88

Table A.12. Change of validation results relative to C and ω 88

Table A.13. Change of validation results relative to C and normalization 89

LIST OF SYMBOLS/ABBREVIATIONS

c	Counter in K-means++ method
C	Cost value for SVM
$d^d(\xi, \tau)$	Descriptor distance between a keypoint tuple ξ and a trajectory τ
$d^s(\xi, \tau)$	Spatial distance between a keypoint tuple ξ and a trajectory τ
$\mathcal{D}(\mathcal{N}_{\varepsilon,t^*}(\xi_i))$	Set of descriptor distances for each trajectory in the neighborhood of ξ_i
$\mathcal{D}(\mathcal{N}_{\varepsilon,t^*}(\tau_i))$	Set of descriptor distances for each keypoint tuple in the neighborhood of τ_i
D_xy	Box filter approximation of the second order derivative of Gaussian in xy direction
E_i	Distance of the ith keypoint to L_N
$\mathbf{g}$	Trajectory descriptor
$\mathcal{K}_t$	The set of keypoint tuples at time t
H_f	Hessian of a function f
I	Image
I_σ	Integral image
L	The line segment between the last keypoint of the trajectory and the new keypoint to be added
L_i	The line segment between the ith and first keypoints of the trajectory
L_xy	The second order derivative of Gaussian in xy direction
N	Number of keypoint tuple in a trajectory
N^*	Number of keypoints in the video sequence
$\mathcal{N}_{\varepsilon,t^*}(\xi)$	The neighborhood of the keypoint tuple $\xi = (\kappa, \mathbf{s}, t^*)$ at time t^*
$\mathcal{N}_{\varepsilon,t^*}(\tau)$	The neighborhood of the trajectory τ at time t^*
p_i	Projection of L_i in the direction of L_N
$\mathrm{pl}_\tau(t)$	Path length until time t over a trajectory τ

r^t	Label of the sample data x^t
R	Length of the line segment between the last keypoint of the trajectory and the new keypoint to be added
$\mathbf{s}$	Keypoint descriptor
t	Time
$\mathcal{T}_t$	The set of trajectories at time t
w	Window size for trajectory calculation
x^t	Sample data
β	The minimum required spread
$\delta(\cdot)$	Displacement function
ε	The maximum allowed spatial position change of a keypoint between consecutive frames
κ	Keypoint
λ	Threshold for matching ratio
μ_x	x coordinate of the mean of the keypoints belonging to the video sequence
μ_y	y coordinate of the mean of the keypoints belonging to the video sequence
ν	The maximum allowed time since last observation
ξ	Keypoint tuple
ω	The maximum number of keypoints for a sub-trajectory
ρ	The minimum age to be eliminated
σ	The minimum required sample count
σ_x	x coordinate of the standard deviation of the keypoints belonging to the video sequence
σ_y	y coordinate of the standard deviation of the keypoints belonging to the video sequence
τ	Trajectory
KLT	Kanade-Lucas-Tomasi feature tracker
k-NN	K-Nearest Neighborhood
KTH	Kungliga Tekniska Högskolan

MEI	Motion Energy Image
MHI	Motion History Image
SVM	Support Vector Machine
SIFT	Scale Invariant Feature Transform
STIP	Space Time Interest Points
SURF	Speeded Up Robust Features
URADL	University of Rochester Activities of Daily Living Dataset
WeCare	Wireless Sensor Network Enabled Care

1. INTRODUCTION

Vision based human action recognition is the process of assigning action labels to videos. In action recognition, videos are readily segmented in time. A video is expected to have only one action. Additionally, an activity is defined to be an ordered set of actions. For example, cooking is an activity whereas stirring is an action. Human action understanding has many application areas concerning security, surveillance, assisted living, and even entertainment. Access control, person identification, anomaly detection, and human-computer interaction are some of the areas that can benefit from human motion analysis.

The increasing use of security cameras requires a great amount of human labor for monitoring purposes. The use of human motion tracking systems result in diverting this human power to more obligatory areas. In addition, the human error caused by fatigue and lack of caution can be eliminated. Furthermore, complementing video cameras with additional sensors can result in a more enhanced sensing mechanism than that of the human eye.

Another application may be crowd flux statistics and congestion analysis at densely populated public areas [5]. Moreover, life quality improvement can be achieved through using these systems; some examples can be tracking the elderly or children. The use of these systems can be extended to the area of entertainment, such as controlling video games [6] or using for motion capture in cinema. All these application areas make the problem of human motion tracking attractive for academic research.

The performances of actions have large variations among different people. For instance, walking speed and stride length may differ in different performances. Let alone the fact that walking speed and stride length of the same person may vary from time to time, they would drastically yield dissimilarities among different people. Moreover, anthropometric differences of individuals hardens the problem. Clothing changes can sometimes make the problem harder. There may be some combined actions. The pri-

mary action may be combined with some other action which can be regarded as noise, such as avoiding obstacles while walking. These inter-class and intra-class variations must be considered in a human action recognition system.

Illumination variances, shadows, and occlusions are challenging situations in human tracking. Cluttered and dynamic environments make the problem of person localization harder. Illumination variances and shadows make it hard to acquire the data and upgrade the background. Handling occlusions is especially a hard problem when working with a single camera. Occlusions can be handled more effectively using multiple cameras.

Temporal variations present another challenging situation. Segmentation of actions in time can prove difficult. Rate of performance is an issue that should be taken into consideration.

Obtaining and labeling training data is another difficulty. When obtaining the dataset, previously mentioned variances should be taken into account. Variations of an action must be accommodated. Also, including different illuminations and environments makes the data more realistic. Moreover, it is very difficult to determine the beginning and end of an action.

In addition, different zoom levels, and the location of the action in the scene present challenges. Furthermore, observations of the same actions from different viewpoints can be very different. A human action recognition system should work under different scales, should be invariant of the location of the action and the viewpoint.

For overcoming some of the mentioned problems, we propose a local representation based approach. Local representations are somewhat invariant to viewpoint changes, the appearance of the person, and partial occlusions [7]. Accurate localization of people, and background subtraction in an environment (such as simple, cluttered, or dynamic backgrounds) are not required. We track local keypoints, and extract information form the trajectories of these tracked keypoints. This makes our approach

local and invariant against the previously mentioned problems. In addition, we normalize the trajectories against time for overcoming the problem of the temporal length variations. Moreover, we normalize the trajectories against spatial position for dealing with different zoom levels and making it invariant of the location of the action.

Furthermore, we introduce a new dataset with a main concentration on fall detection. Fall detection is a major part of elderly care systems.

1.1. Literature Review

There are numerous surveys in the literature addressing the problems of human motion tracking, action recognition and their challenges [6, 8, 7]. Moeslund et al. addresses the advances in human motion capture and analysis [6]. Poppe also addresses the advances in vision-based human motion analysis [8]. In [7], Poppe addresses the advances in vision-based human action recognition.

We divide the feature extraction methods from the images of a video sequence into two categories: global and local representations. In global representations, the person is localized first, then the region of interest is encoded as a whole. On the other hand, in local representations, some patches in the images are encoded separately. These patches are usually extracted from the neighborhoods of interest points. The representation is found by combining the information of these patches.

In addition, some approaches fit human body models onto video sequences and recognize the actions using these models.

1.1.1. Global representations

Bobick and Davis [9] construct a binary motion energy image which they call MEI. In MEI, the pixels where a motion occurred since the start frame are shown in white, and the other pixels are shown in black. They also construct a motion history image(MHI), where pixel intensity is a function of the temporal history of the motion

at that point. In MHI, more recently moving pixels are brighter [9]. They use these two images together, and extract seven Hu moments to the describe the motion of a sequence.

Weinland and Boyer [10] find edges of the frames of a sequence and match these with previously labeled silhouette templates using Chamfer distance. Then, they use the vector of minimum distances between silhouettes and frames to describe the sequence.

Efros et al. [11] calculate optical flow measurements in a spatio-temporal volume for figure centered images. They use sports footage in which the people in the image are very small, thus noisy. Instead of using the measurements as they are, they treat optical flow measurements as a spatial pattern of noisy measurements and blur them by smoothing. They divide both horizontal and vertical components of optical flow into positively and negatively directed parts, yielding 4 distinct channels. They do the smoothing for each channel. At the end, they do a nearest neighbor search using these values in a database of preclassified actions to find labels of actions, joint locations, and appearance informations.

Tran and Sorokin [12] compute optical flow using Lucas-Kanade algorithm [13] and smooth using Efros' method [11]. They use background subtraction to find the silhouette and a bounding box. They use two components of optical flow together with the silhouette. They rescale the part in the bounding box and split the rescaled part into 2 x 2 sub-windows. Then each sub window is divided into 18 pie slices covering 20 degrees each. They calculate the sum of the values in the regions to form a 216 dimensional frame descriptor. They stack together the descriptors of five consecutive frames to form a descriptor of the moment. They use one nearest neighbor to classify actions.

Gorelick et al. [14] stack silhouettes of the consecutive frames to form space-time volumes. They regard actions as three dimensional shapes using these space-time volumes. They solve the Poisson equation to extract space-time features such as

local space-time saliency, action dynamics, shape structure and orientation. They use weighted moments over these features to find global features.

Batra et al. [15] find silhouettes in the frames and extract three dimensional sub-patches of the stacked silhouettes which they call Space-Time Shapelets. They propose a dictionary of these three dimensional patches and represent an action using a bag of words approach over these space-time patterns.

Ogata et al. [16] construct space-time volumes of optical flow for event detection and use Efros' method [11] for describing motions. They combine these two methods with boosting.

1.1.2. Local representations

Laptev [17] proposed space-time interest points by extending the Harris interest point detector to 3D. Space-time interest points are the points where local neighborhood has a significant change in both the temporal and the spatial domain. We give the details of the work of Laptev in Section 2.5. Schuldt et al. [2] used space-time interest points for recognition. They use a bag of words approach on space-time interest point and classification is done using support vector machines.

Oikonomopoulos et al. [18] use three dimensional spatio-temporal salient point detection. They measure the variations in the information content of pixel neighborhoods for detecting the points. The centers of the spatio-temporal cuboids with local maximum energy are selected as salient points. They introduce a distance metric based on the Chamfer distance to measure the distance between collections of salient points.

Dollar et al. [19] propose a method for detecting spatio-temporal interest points with the claim that 2D interest point detectors' 3D extensions are not adequate enough for detecting spatio-temporal feature points. They use Gabor filters both on spatial and temporal dimensions individually. By changing the neighborhood's spatial and temporal sizes, the number of interest points are adjusted.

Willems et al. [20] use integral videos to find salient points. By the use of integral videos, they easily filter the spatio-temporal space for finding both spatially and temporally scale-invariant cuboids. They use box filters and use determinant of a 3D Hessian matrix as saliency measure.

Wang et al. [21] evaluate and compare different space-time feature methods in a common experimental setup. They use bag of features approach and use support vector machines for classification. They show that the tested space-time interest point detectors are outperformed by regular sampling of space-time features for human actions in realistic settings.

Sun et al. [22] find scale invariant feature transform (SIFT) [23] descriptors and track interest points using these descriptors. They use Markov chains to represent trajectories as dynamic systems. Then, they use bag of words approach on Markov chain representations and classify actions using nonlinear support vector machines.

Messing et al. [3] use the Kanade-Lucas-Tomasi (KLT) tracker to track features and extract trajectories. They find each trajectory's quantized velocity over time. They quantize velocity histories uniformly in log-polar coordinates and use eight bins for direction, and five bins for magnitude. They use a bag of words approach on these velocity features to describe the video and use naive Bayes for modeling. Additionally, they augment the codeword of the position of the face detected by Viola-Jones algorithm [24] and the codewords of the initial and the final positions of the trajectories to their features. When adding these features, the codewords are independently generated by clustering the set of related positions in the training set separately. This leaves the codewords highly dependent on the position of the action on the frame. The details of the KLT tracker are given in Section 2.4.

Niebles et al. [25] extract space-time interest points and represent a video as a collection of these points using a bag of words approach. Then, a generative model is constructed for each class using probabilistic latent semantic analysis.

1.1.3. Model Fitting approaches

In the literature, human detection, and body tracking are usually done before model fitting. Recognition is done using the models after the model fitting stage.

1.1.3.1. Human detection. Human body detection starts with finding the moving regions in an image, then classifying these regions into human and non-human objects. Background subtraction, temporal differencing and optical flow are some of the commonly used motion detection techniques [5].

For stationary backgrounds, background subtraction can be used for motion segmentation. However, this method is very sensitive to the changes in the lighting conditions of the scene. Therefore, a good background model should be used [5, 26, 27, 28]. Temporal differencing compares the pixels of two or three consecutive frames for extracting moving regions. In [29], temporal differencing is used for detecting moving targets in real-time videos. After motion detection, some filtering techniques, morphological operators and connected component analysis may be used for improving results.

Object classification can be categorized into two approaches, which are shape-based classification and motion-based classification. In shape-based classification, objects are classified according to their modal features such as points, boxes, silhouettes and blobs [5]. Lipton et al. [29], classified moving objects as humans, vehicles and clutter. In motion-based classification, objects are classified according to their movement characteristics. Since human motion has a periodic property, this has been used as a classification feature [5].

1.1.3.2. Body tracking. Tracking stage deals with matching a human body blob with its temporal correspondent and tracking it through time. There may be more than one person in the scene and also there may be some occlusions during the observation. Human body tracking also tries to deal with these types of situations. Generally, dif-

ferentiation of tracking from detection and model fitting stages are not very clear, since numerous tracking algorithms incorporate detection or model fitting in the pipeline. Tracking can be categorized into region-based tracking, active-contour-based tracking, feature-based tracking and model-based tracking [5].

In region-based tracking methods, objects are tracked according to variations of image regions. Region-based tracking algorithms usually detect motion regions by background subtraction [5]. McKenna et al. [27], combine color and gradient information for dealing with shadows and unreliable color cues, which is used in an adaptive background subtraction method.

As the name implies, active-contour-based tracking algorithms track objects by representing their boundaries as contours which are updated dynamically. Paragios et al. [30], use geodesic active contours for tracking multiple moving objects.

Feature-based tracking algorithms extract features of objects for each frame and track objects by matching these features. Centroids, perimeters, areas, colors, line segments, curve segments, and corner vertices are some of the features that are used in the literature [5].

In model-based tracking algorithms, objects are tracked by matching the projections of object models, which are produced with prior knowledge, to the image. Model-based tracking has significant overlap with model fitting and will be discussed in the next section.

1.1.3.3. Model fitting. Model fitting is done for finding the pose of the tracked person, which can also be used for behavior analysis. The main objective is minimizing the error between the observation and the human body model. Pose estimation can be done in either 2D or 3D, using either monocular or multi-view images. A variety of models can be used; such as stick figures, 2D contours, volumetric models, and hierarchical models [5].

Some notable works in the literature have been on using multi-view images for obtaining 3D models. Deutscher et al. [31] develop a modified particle filter, termed as annealed particle filtering, for searching in high dimensional configuration spaces. The method combines annealing with stochastic sampling. Kehl et al. [32] use stochastic sampling and stochastic meta descent optimization for full body pose tracking with 24 degrees of freedom from multiple views. In [33], partitioned sampling is used on articulated objects such as the hand for estimating the pose. In [34], Plankers et al. tracked arm movements using stereo and silhouette cues.

There are also monocular approaches, since for many applications only a single camera is available [8]. The major drawbacks of using a monocular approach arise from the difficulties in solving depth and occlusion problems. Eliminating these drawbacks requires using constraints on kinematics and movement [6]. 3D pose estimation using monocular images is much more difficult than using multiple cameras.

There are two main approaches for estimating the pose parameters of the tracked human body. They are classified as top-down and bottom-up approaches. In addition, some researchers use a combination of both approaches [8] for fitting the human body model onto the tracked human body.

Top-down approaches deal with searching the appropriate model pose that matches the observation with a minimum error. This is also called as analysis-by-synthesis. In [35], a local search is performed around the initial estimate. In [36], physical forces are used for minimization of the differences between the pose of the model and the observation. Since search is costly, they use the estimate of the previous frame as an initial estimate for the current frame to reduce the search space. However, manual initialization is needed for the first frame, since it does not have a previous frame. Rendering the model and calculating the error is also costly. Erroneous estimation of head or torso parts affects other body parts in the lower kinematic chain. These drawbacks are reported in [8].

Bottom-up approaches deal with finding the individual body parts and assembling

them. During the assembly, some physical constraints are considered, such as body part proximity. Appearance models are widely used for modeling body parts. In [37], the appearances of individual body parts are modeled by a 2D approach. In [38], the images first pass through a segmentation phase and then body part locators are used on the segmented image. The outputs of the body part locators are combined into a human model. The templates may produce false positives misguided by the limb-like regions in the image. Also each template often needs its own part detector [8].

Combination of both the top-down and bottom-up approaches can help eliminate the drawbacks mentioned above. The initialization problem of the top-down approach can be solved with the bottom-up approach using a decent part detection framework. In [39], search space decomposition is used. Part detectors for the lower kinematic chain are used on the image region defined by the parent in the kinematic chain. This approach is computationally cheaper yet the individual part detectors need to have a good performance. Bottom-up information can also be used in a statistical framework, such as in [40]. Part of the pose space can be estimated by using part detectors and inverse kinematics [41]. This approach solves the problems of a pure top-down approach by using the bottom-up information only when available. This way there is no need for a part detector for each body part [8]. The combined approach has also been utilized in some recent work for the recovery of poses in cluttered scenes, as reported by [8].

1.2. Outline of the Thesis

Chapter 2 describes the mathematical details of the methods used throughout the thesis. In addition, it covers some other notable methods in the literature which are used for human action recognition tasks.

In Chapter 3, the outline of the system is explained. The Generic Keypoint Tracker method is proposed and elaborated. Also in Chapter 3, the details of feature extraction from trajectories and video sequences are given. Moreover, the classification method is explained.

In Chapter 4, a new dataset is introduced. In addition, the datasets that are used in the evaluation of the thesis are described. Chapter 4 also describes the experiment setup. Analysis of validation results and parameter selection is explained in Chapter 4. Finally, test results are presented.

Chapter 5 concludes this thesis with a summary of the obtained results, followed by possible future work.

2. TECHNICAL BACKGROUND

This chapter provides the details of the methods that are used throughout this thesis and some other notable methods that are used in the literature for human action recognition tasks.

2.1. Keypoint Detection and Matching

Keypoint detection and matching is generally used for object detection, image registration, and camera calibration. There are three main steps in the task of finding point correspondences between two images:

- Feature Detection
- Descriptor Extraction
- Keypoint Matching

2.1.1. Feature Detection

Feature detection is the task of finding important points in the images that we try to match. We call these points keypoints or interest points. In a feature detector, repeatability is a very important concern. Here, repeatability denotes the ability of the feature detector to find the same physical point under different conditions of illumination and position.

2.1.2. Descriptor Extraction

Descriptor extraction is the process of extracting meaningful descriptors about the neighborhoods of interest points. A descriptor should make a distinction between unrelated keypoints. In addition, it should be similar between related keypoints that are having different illumination conditions or geometric transformations.

2.1.3. Keypoint Matching

A descriptor matcher matches keypoints between two images using the explained descriptor extractor. The keypoints are matched using the distance between their descriptors. The distance measure can differ among different descriptor matchers such as Euclidean distance or Mahalanobis distance. Matching can be done using nearest neighbors, thresholding, or any other method.

2.1.4. Speeded-up Robust Features (SURF)

Speeded-up Robust Features (SURF) provides solutions to both keypoint detection and descriptor extraction problems. It is first proposed in 2006 by Bay et al. [42], and later elaborated in 2008 [1]. The algorithm is faster than Scale-invariant feature transform (SIFT) which was introduced by Lowe [43], and detailed in [23]. In addition, Bay et al. show that SURF is more robust under some image transformations. SURF algorithm has two main parts:

- Interest point detection
- Interest point description

2.1.4.1. SURF interest point detection. There are three subparts in SURF interest point detection; integral image calculation, scale space representation, and interest point localization.

Integral images: In 1984, Crow et al. introduced the concept of summed area tables to the computer graphics world [44]. Later in 2001, Viola and Jones used the concept for object detection [24] and named it integral images. It is an algorithm for efficiently calculating the sum of the values of any rectangular subpart of an image. The value of a pixel of an integral image is the sum of the values of the rectangular region of the original image formed by the origin and that pixel.

The integral image I_Σ of an image I is defined as

$$I_\Sigma(x,y) = \sum_{i=0}^{i\leq x}\sum_{j=0}^{j\leq y} I(x,y). \tag{2.1}$$

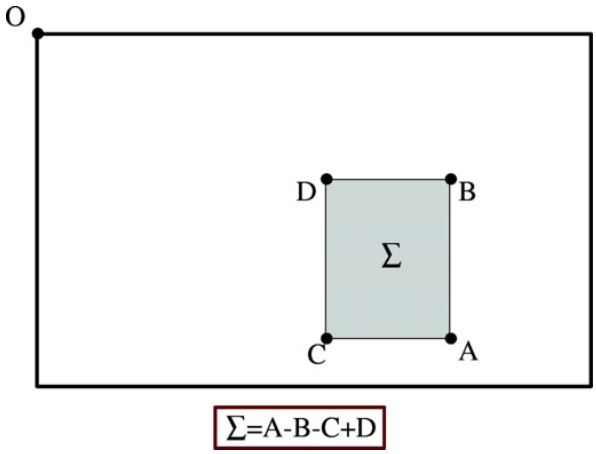

Figure 2.1. Calculating the sum of the values in a rectangular region using the integral image [1]

This operation can be easily done by cumulatively summing all the pixels. After calculating the integral image, only three additions are needed to calculate the sum of the values in a rectangular region. Figure 2.1 shows this operation.

Hessian matrix-based interest points: We know that using the second partial derivative test we can classify the extrema of a function. We can do the test using the determinant of the Hessian of a function f. Hessian of f is

$$H_f(x,y) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}. \tag{2.2}$$

And the determinant of the Hessian of f is

$$\det H_f(x,y) = \frac{\partial^2 f}{\partial x^2}\frac{\partial^2 f}{\partial y^2} - \left(\frac{\partial^2 f}{\partial x \partial y}\right)^2. \tag{2.3}$$

If $\det H(x,y) < 0$, the point (x,y) is not a local extremum. If $\det H(x,y) > 0$, the point

(x, y) is a local extremum. If $\det H(x, y) = 0$, the point (x, y) the test is inconclusive.

For using this method on images, we need to calculate the second order partial derivatives of an image. The derivatives are found using Laplacian filters. Derivative filters are very sensitive to noise. Thus, the image is smoothed before applying the Laplacian. This is equivalent to convolving the smoothing filter with the Laplacian. Gaussians are used in SURF.

$$\det H_I(x, y, \sigma) = L_{xx}(x, y, \sigma)L_{yy}(x, y, \sigma) - (L_{xy}(x, y, \sigma))^2, \tag{2.4}$$

where

$$L_{xy}(x, y, \sigma) = \frac{\partial^2 g(\sigma)}{\partial x \partial y} \circ I(x, y), \tag{2.5}$$

is the second order derivative of $g(\sigma)$ in xy direction, and $g(\sigma)$ is the Gaussian with scale σ.

Lowe approximated Laplacian of Gaussian using the difference of two nearby scales separated by a constant multiplicative factor [23]. Bay et al. pushed the approximation further using box filters. This way integral images can be used efficiently and computational cost decreases significantly. This makes calculation time independent of the filter size. Bay et al. found that the multiplicative factor of 0.9 is suitable for their approximations.

$$\det H_{\text{approx}} = D_{xx}D_{yy} - (0.9D_{xy})^2, \tag{2.6}$$

where D_{xy} is the box filter approximation of L_{xy}. Figure 2.2 shows discretized and cropped Gaussian filters and box filter approximations.

Scale space representation: Interest points may appear in different scales. Thus, they are needed to be searched in different scales. In computer vision, image pyramids are commonly used to construct the scale-space.

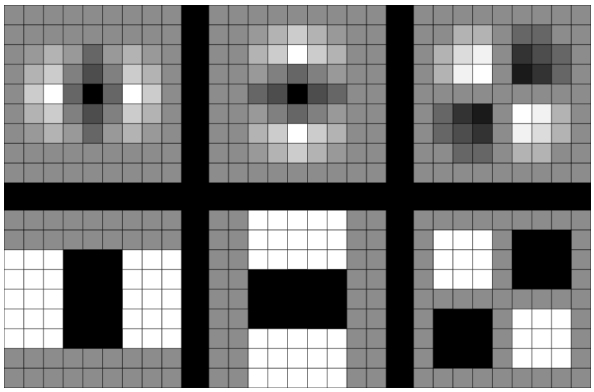

Figure 2.2. Filters. Top: The discretized and cropped Gaussian second order partial derivatives in the x, y and xy-directions. Bottom: Weighted box filter approximations in the x, y and xy-directions. [45]

Previously explained integral image and box filter approaches used together speed up this search. Increase in the filter size corresponds to decrease in the image scale. Initial scale layer is the output of the 9 x 9 box filter, referred as scale s=1.2, since it approximates Gaussian filters with $\sigma = 1.2$. Scales are calculated with the formula

$$\sigma_f = 1.2\frac{f}{9}, \tag{2.7}$$

where f is the filter size. Figure 2.3 shows the change in filter size when the scale changes. The scale space is divided into octaves. An octave is a series of response maps obtained by filtering the same input image with a filter of increasing size. Each octave is divided into a number of scale levels. Difference between scale levels depend on the length of the lobes of the partial second order derivative. In a 9 x 9 filter, this number is 3. Since we require a central pixel in each lobe, we need to enlarge these lobes from each side. Thus, we need to add an even integer, a minimum number of two. Since there are three lobes in D_{xx} and D_{yy}, we need to increase the filter size by six. The filter sizes for the first octave are 9 x 9, 15 x 15, 21 x 21, and 27 x 27. The filter size increase is doubled on each new octave. At the second octave it is increased from six to 12, at the third octave it is increased from 12 to 24, so on and so forth.

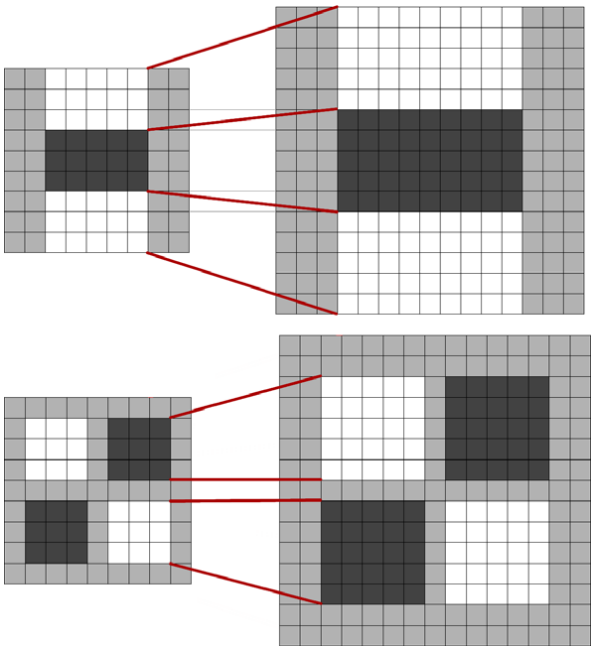

Figure 2.3. Filters in x and xy directions for the scale levels 9 x 9 and 15 x 15 [1]

The formula for filter size is

$$\text{FilterSize} = 3 * (2^{\text{octave}} * \text{layer} + 1) \tag{2.8}$$

as shown by Evans [45]. For each octave and layer couple, the response of the pixels are calculated.

Interest point localization: Non-maxima suppression is applied in a 3 x 3 x 3 neighborhood for localizing the interest points. The first two dimensions of the neighborhood represent the spatial position and the third dimension represents scale. Every response is compared with its neighboring responses. If a response is greater than all its neighboring responses, then it is classified as an interest point. Then, the method proposed by Brown and Lowe [46] is used for interpolating the responses to find the sub-pixel accurate location in both space and scale.

2.1.4.2. SURF interest point description. There are two subparts in SURF interest point description; interest point orientation assignment, and descriptor calculation.

Orientation assignment: For achieving invariance to image rotation, the orientation of the detected interest point is found. First, for each interest point, Haar wavelet responses of size $4s$ within a circular neighborhood of radius $6s$ are calculated, where s is the scale at which the interest point was detected. The sampling step is chosen to be s. Figure 2.4 shows Haar wavelet filters. Then, they are weighted with a Gaus-

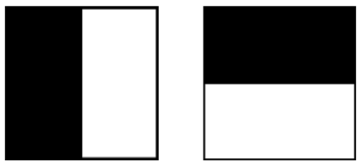

Figure 2.4. Haar wavelet filters in x and y directions [1]

sian having $\sigma = 2s$, centered at the interest point. After weighting, the responses are mapped to a new space where the response in x direction is the abscissa and the response in y direction is the ordinate. The dominant orientation is found by rotating a sliding orientation window of size $\frac{\pi}{3}$ and calculating the sum of all responses within the window to form a new two dimensional vector. Then, the orientation having the longest of such vectors defines the orientation of the interest point. Figure 2.5 shows the orientation assignment.

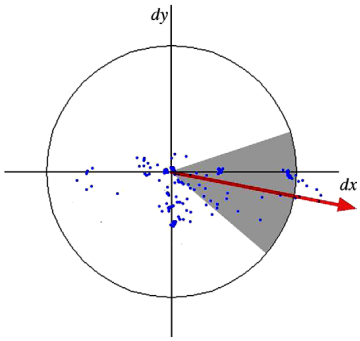

Figure 2.5. A sliding orientation window of size $\frac{\pi}{3}$ [1]

Descriptor based on sum of Haar wavelet responses: For the descriptor extraction stage, a 4 x 4 grid of regions having a size of 20s x 20s and oriented along the previously found orientation is placed around the interest point. For each of these regions,

Haar wavelet responses are computed at 5 x 5 regularly spaced sample points and the responses are weighted with a Gaussian filter having $\sigma = 3.3s$, centered at the interest point. Figure 2.6 shows the oriented grid and sample points. Calling these weighted responses d_x and d_y, four dimensional descriptor vectors are extracted from each of the 16 regions which are of the form

$$\left(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\right). \tag{2.9}$$

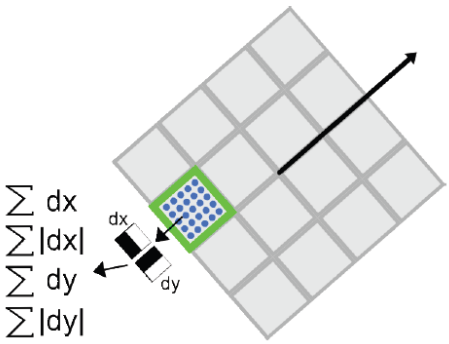

Figure 2.6. The green square shows a region of the 4 x 4 grid and blue circles represent the sample points at which the wavelet responses are computed. Haar wavelet responses are calculated relative to the dominant orientation. [45]

Then, these vectors are concatenated to form the 64 dimensional descriptor vector of the interest point. The 64 dimensional vector is normalized to form a unit vector for bringing invariance to contrast. Absolute valued sums give information about the polarity of the intensity changes. Figure 2.7 shows the difference in descriptor vectors in three different situations.

Fast indexing for matching: For fast indexing during the matching stage, the sign of the Laplacian is used. It needs no extra computation since it was already computed during the detection stage. The sign of the Laplacian distinguishes the dark blobs on bright backgrounds from the bright blobs on dark backgrounds. Thus, before calculating Euclidean distance between two descriptors it is advantageous to check whether the signs match.

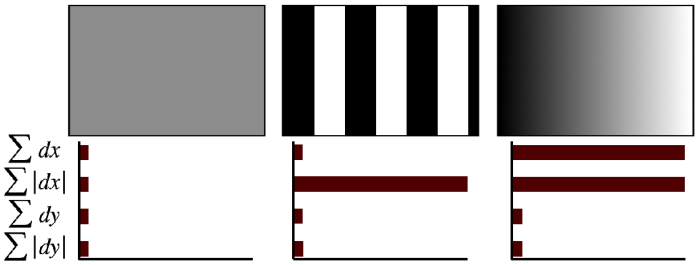

Figure 2.7. Change in descriptors. Left: In a homogeneous region, all values are low. Middle: In the presence of frequencies in x direction, the value of $|dx|$ is high, but others remain low. Right: If the intensity is gradually increasing in x direction, both values dx and $|dx|$ are high. [42]

2.2. Clustering

2.2.1. K-Means

K-means is a method for clustering. It was proposed by MacQueen in 1967 [47]. The purpose of the k-means method is to partition data having N observations into K clusters.

Let x^t be an observation. First, we start with some cluster centers m_i, initialized randomly or via another method, where m_i and x^t has the same number of dimensions and $i \in 1, \ldots, K$. We define b_i^t as

$$b_i^t = \begin{cases} 1 & \text{if } ||x^t - m_i|| = \min\limits_j ||x^t - m_j|| \\ 0 & \text{otherwise} \end{cases}. \tag{2.10}$$

Then, for each observation, we assign x^t to the cluster i if $b_i^t = 1$. After assigning each observation, we update our cluster centers [48] using

$$m_i = \frac{\sum\limits_{t=1}^{N} b_i^t x^t}{\sum\limits_{t=1}^{N} b_i^t}. \tag{2.11}$$

We repeat these two steps until cluster assignments do not change.

K-means++: K-means++ is a method for choosing the initial centers for k-means. It is proposed by Artur and Vassilvitskii in 2007 [49]. In this algorithm, initially, a center is chosen uniformly at random from the data points and a counter c is initialized to one. Then, for each data point x^t, $D(x^t) = \min_j ||x^t - m_j||$ is computed, where $j \in 1, \ldots, c$. Using a weighted probability distribution where a data point x^t is chosen with probability proportional to $D(x^t)^2$, another data point is chosen, then assigned as a new cluster center, and c is incremented by one. These steps are repeated until $c = K$ centers are chosen.

2.2.2. Bag-of-Words

Bag of words method has been proposed for document classification. In this method, d words are chosen that are believed to give information regarding the class. The representation of the text is a d dimensional vector where the value of a component of the vector is one if the corresponding word exists in the text, zero otherwise [48].

Alternatively, the histogram of the text can be found by counting the repetitions of these words. Therefore, given any text, d dimensional histogram vector of the text can be found which simplifies the comparison process. In addition, the histograms are normalized to have one as the sum of the elements of the vector. This normalization process makes the representation invariant of the text length.

The application of bag of words to computer vision problems is simple. We count the number of objects or some other features instead of words.

2.3. Supervised Learning

2.3.1. K-Nearest Neighbor (k-NN)

K-Nearest Neighbor is a simple classification method, sometimes referred to as k-NN. In K-Nearest Neighbor, given any vector, the distance of this vector to every vector in the training set is computed. Then, K training vectors which give the minimum distances are selected as the nearest neighbors. The class having the most examples in these neighbors is assigned as the class of the sample vector.

2.3.2. Support Vector Machines (SVM)

Support vector machines (SVM) are proposed by Vapnik [50]. In 1995, Cortes and Vapnik suggested the usage of soft margin for allowing nonseperable instances [51].

SVM is a discriminant based method. It tries to separate the data into two classes using a hyperplane. In addition, a margin is used for better generalization [48].

Let $\mathbf{x^t}$ be a sample, and r^t be the label of $\mathbf{x^t}$. r^t can be $+1$ or -1 depending on the class. We would like to find $\mathbf{w}$ and w_0 such that $r^t(\mathbf{w}^T\mathbf{x^t} + w_0) \geq 1$.

$\frac{r^t(\mathbf{w}^T\mathbf{x^t}+w_0)}{||w||}$ gives the distance of $\mathbf{x^t}$ to the discriminant. We would like this distance to be greater than or equal to some value ρ:

$$\frac{r^t(\mathbf{w}^T\mathbf{x^t} + w_0)}{||w||} \geq \rho, \forall t. \tag{2.12}$$

For maximizing the margin, we minimize $||w||$. The problem can be defined as

$$\min \frac{1}{2}||w||^2 \text{ subject to } r^t(\mathbf{w}^T\mathbf{x^t} + w_0) \geq 1, \forall t. \tag{2.13}$$

The problem can be redefined using Lagrange multipliers α^t [48]:

$$L_p = \frac{1}{2}||w||^2 - \sum_t \alpha^t r^t(\mathbf{w}^T\mathbf{x^t} + w_0) + \sum_t \alpha^t. \tag{2.14}$$

We would like to minimize L_p with respect to $\mathbf{w}$ and w_0.

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \implies \mathbf{w} = \sum_t \alpha^t r^t \mathbf{x^t} \tag{2.15}$$

$$\frac{\partial L_p}{\partial w_0} = 0 \implies \sum_t \alpha^t r^t = 0 \tag{2.16}$$

Dual of the problem can be defined by plugging Equations 2.15 and 2.16 into L_p [48]:

$$L_d = -\frac{1}{2}\sum_t - \sum_s \alpha^t \alpha^s r^t r^s (\mathbf{x^t})^T \mathbf{x^s} + \sum_t \alpha^t, \tag{2.17}$$

which must be maximized with respect to α^t, subject to the constraints $\sum_t \alpha^t r^t = 0$, and $\alpha^t \geq 0, \forall t$.

After solving Equation 2.17 using quadratic optimization methods [48], most of α^t will be 0. The set of $\mathbf{x^t}$ having $\alpha^t > 0$ are support vectors. $\mathbf{w}$ can be computed by Equation 2.15, and w_0 can be computed by $w_0 = r^t - \mathbf{w}^T\mathbf{x^t}$. This should be done for all support vectors and the average is to be taken for numerical stability.

While using a soft margin, we relax the requirement as

$$r^t(\mathbf{w}^T\mathbf{x^t} + w_0) \geq 1 - \xi^t, \tag{2.18}$$

where $\xi^t \geq 0$ are slack variables. If $0 < \xi^t < 1$, x^t is in the margin and correctly classified. If $\xi^t \geq 1$, x^t is misclassified. Soft error is the sum of the slack variables. It is multiplied by a penalty factor C and added to the problem. Therefore, the problem

becomes minimizing

$$\min \frac{1}{2}||w||^2 + C\sum_t \xi^t \text{ subject to } r^t(\mathbf{w}^T\mathbf{x^t} + w_0) \geq 1 - \xi^t, \forall t. \tag{2.19}$$

New Lagrange parameters are added to the problem to guarantee the positivity of ξ^t. The derivatives with respect to $\mathbf{w}$, w_0, and ξ^t are taken and set to 0 for minimization. Then, the results are plugged similarly. The dual is:

$$L_d = -\frac{1}{2}\sum_t - \sum_s \alpha^t\alpha^s r^t r^s (\mathbf{x^t})^T\mathbf{x^s} + \sum_t \alpha^t, \tag{2.20}$$

which must be maximized with respect to α^t, subject to the constraints $\sum_t \alpha^t r^t = 0$, and $0 \leq \alpha^t \leq C, \forall t$.

The set of $\mathbf{x^t}$ having $\alpha^t > 0$ are support vectors; they define $\mathbf{w}$. The $\mathbf{x^t}$ having $\alpha^t < C$ are on the margin; they have $\xi^t = 0$ and define w_0. $\mathbf{x^t}$ having $\alpha^t = C$ are in the margin or misclassified.

For dealing with nonlinear problems, we can map the instances to a new space, then use a linear model in the new space. This linear model corresponds to a nonlinear model in the original space [48]. In the SVM case, we map x as

$$\mathbf{z} = \phi(\mathbf{x}), \tag{2.21}$$

where $z_j = \phi_j(\mathbf{x})$, $j = 1, \ldots, k$, $\mathbf{x}$ is d dimensional, $\mathbf{z}$ is k dimensional, and $z_1 = \phi_1(\mathbf{x}) = 1$. Then, we define the discriminant function:

$$g(\mathbf{x}) = \sum_{j=1}^{k} w_j\phi_j(\mathbf{x}). \tag{2.22}$$

Similar operation as the soft margin case gives the dual:

$$L_d = -\frac{1}{2}\sum_t - \sum_s \alpha^t \alpha^s r^t r^s \phi(\mathbf{x^t})^T \phi(\mathbf{x^s}) + \sum_t \alpha^t, \tag{2.23}$$

subject to the same constraints as Equation 2.20.

Then, we define a kernel function $K(\mathbf{x^t}, \mathbf{x^s}) = \phi(\mathbf{x^t})^T \phi(\mathbf{x^s})$ and plug it into the Equation 2.23:

$$L_d = -\frac{1}{2}\sum_t - \sum_s \alpha^t \alpha^s r^t r^s K(\mathbf{x^t}, \mathbf{x^s}) + \sum_t \alpha^t. \tag{2.24}$$

Then, the discriminant becomes:

$$g(\mathbf{x}) = \sum_t \alpha^t r^t K(\mathbf{x^t}, \mathbf{x}). \tag{2.25}$$

It is much simpler to define and use a kernel function instead of defining $\phi(\mathbf{x})$ and calculating $\phi(\mathbf{x^t})^T \phi(\mathbf{x^s})$.

In multiclass classification, there are two widely used approaches. Let K be the number of classes. In the one-vs-all approach, K two class problems are defined, each one separating one class from all others. In this approach, the test data is compared with all discriminants and the maximum one is selected as the class. In the one-vs-one approach, $K(K-1)/2$ classifiers are defined, each one separating a pair of classes. In the classification phase of this approach, the test data is compared with all discriminants, each resulting in a candidate class, and a voting is done among these candidate classes. The class receiving the most votes is assigned as the class of the test data.

$\chi 2$ *Kernel:* Schiele and Crowley showed that $\chi 2$ distance is suitable for comparing histograms [52]. Chapelle et al. used χ^2 distance in SVM kernels [53]. We define χ^2

kernel as

$$K_{\chi^2}(\mathbf{x^t}, \mathbf{x^s}) = 1 - \sum_{i=1}^{n} \frac{(x_i^t - x_i^s)^2}{\frac{1}{2}(x_i^t + x_i^s)}. \tag{2.26}$$

2.4. KLT Feature Tracker

The name KLT is derived from the initials of the last names of the authors Kanade, Lucas, and Tomasi. KLT is a feature tracking method. It is based on the work of Lucas and Kanade [13]. Its development is completed by Tomasi and Kanade [54], and it was clarified by Shi and Tomasi [55]. The algorithm is widely used in the literature for tracking and action recognition purposes. In this section, we are going to give a brief explanation about some details of the algorithm.

As described in [13], the algorithm does not track single pixels, but windows of pixels, and it requires windows that contain sufficient texture. It models the motion of the window using an affine motion field [55]. We define the displacement function $\delta(x)$ as

$$\delta(x) = Dx + d, \tag{2.27}$$

where $D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix}$ is a deformation matrix, and d is the translation of the window.

Assuming I to be the first, and J to be the second image:

$$J(Ax + d) = I(x), \tag{2.28}$$

where $A = 1_{2x2} + D$, and 1_{2x2} is the 2 x 2 identity matrix. Tracking means finding the six parameters of D and d.

We try to minimize the dissimilarity between two images for a given feature

window. We define dissimilarity as

$$\epsilon = \int\int_W [J(Ax+d) - I(x)]^2 w(x) dx, \tag{2.29}$$

where W defines the feature window, and $w(x)$ is a weighting function.

We differentiate it with respect to the unknowns D and d.

$$\frac{1}{2}\frac{\partial \epsilon}{\partial D} = \int\int_W [J(Ax+d) - I(x)] g x^T w dx \tag{2.30}$$

$$\frac{1}{2}\frac{\partial \epsilon}{\partial d} = \int\int_W [J(Ax+d) - I(x)] g w dx, \tag{2.31}$$

where $g = \left(\frac{\partial J}{\partial x}, \frac{\partial J}{\partial y}\right)^T$ is the spatial gradient of the image intensity.

We approximate $J(Ax+d)$ by its Taylor series expansion to the linear term.

$$J(Ax+d) = J(x) + g^T(u), \tag{2.32}$$

where $u = Dx + d$. Then, using Equations 2.30, 2.31, and 2.32, we have

$$\int\int_W g x^T (g^T u) w dx = \int\int_W [I(x) - J(x)] g x^T w dx, \text{ and} \tag{2.33}$$

$$\int\int_W g (g^T u) w dx = \int\int_W [I(x) - J(x)] g w dx. \tag{2.34}$$

In adjacent frames, the affine deformation D is expected to be very small, for tracking to work at all. Thus, it is safer to set D to the zero matrix, initially. Then, the goal is to determine d. It is shown in [55] that this system is equivalent to

$$Tz = a, \tag{2.35}$$

where

$$T = \int\int_W \begin{bmatrix} U & V \\ V^T & Z \end{bmatrix} w dx, \tag{2.36}$$

$$z = (d_{xx}, d_{yx}, d_{xy}, d_{yy}, d_x, d_y)^T, \tag{2.37}$$

$$a = \int\int_W [I(x) - J(x)] \begin{bmatrix} xg_x \\ xg_y \\ yg_x \\ yg_y \\ g_x \\ g_y \end{bmatrix} w dx, \tag{2.38}$$

and U, V, and Z are matrices of the variables x, y, g_x, and g_y.

$$U = \begin{bmatrix} x^2g_x^2 & x^2g_xg_y & xyg_x^2 & xyg_xg_y \\ x^2g_xg_y & x^2g_y^2 & xyg_xg_y & xyg_y^2 \\ xyg_x^2 & xyg_xg_y & y^2g_x^2 & y^2g_xg_y \\ xyg_xg_y & xyg_y^2 & y^2g_xg_y & y^2g_y^2 \end{bmatrix}. \tag{2.39}$$

$$Z = \begin{bmatrix} g_x^2 & g_xg_y \\ g_xg_y & g_y^2 \end{bmatrix}. \tag{2.40}$$

$$V = \begin{bmatrix} xg_x^2 & xg_xg_y \\ xg_xg_y & xg_y^2 \\ yg_x^2 & yg_xg_y \\ yg_xg_y & yg_y^2 \end{bmatrix}. \tag{2.41}$$

Since we assume pure translation, we need to solve a smaller system

$$Zd = e, \tag{2.42}$$

where e collects the last two entries of the vector a. As time passes, the dissimilarity between the first and the current frame will increase. Therefore, it may be essential to solve the entire system.

The symmetric 2 x 2 matrix Z must be above the image noise level and well-conditioned. These requirements imply that the eigenvalues of Z must be large and cannot differ by several orders of magnitude [55]. Salt and pepper textures, or any other pattern that can be tracked results in two large eigenvalues. Let λ_1 and λ_2 be the two eigenvalues of Z. Then, we accept a window if

$$\min(\lambda_1, \lambda_2) > \lambda, \tag{2.43}$$

where λ is a predefined threshold. For determining this threshold, two measurements are taken with the camera to be used during the tracking. As a first measurement, the eigenvalues of a region of approximately uniform brightness are measured. Then, the eigenvalues of a highly textured region are measured. The halfway in-between these two eigenvalues is chosen to be the threshold.

Bouguet proposed a pyramidal implementation of the algorithm [56]. Bouguet defines pyramid levels, starts from the deepest level and iterates the solution through levels. The aim of this method is to track large windows along with small windows. The pyramidal tracking algorithm proceeds as follows. First, the optical flow is computed at the deepest pyramid level. The result of the computation is propagated to the upper level as an initial guess for the pixel displacement. The refined optical flow is computed at the upper level using the previously computed initial guess. The same procedure is repeated until reaching the level of the original image.

2.5. Space-Time Interest Points (STIP)

Space-Time Interest Points (STIP) is an interest point detection and descriptor extraction method in which the notion of interest points is extended into the spatio-temporal domain. In this method, the behavior of interest points are investigated in spatio-temporal scale space and both spatial and the temporal scales of the detected features are adapted [17].

Interest Points in the Spatial Domain: The detection is based on Harris and Förstner interest point operators. Let I be an image in spatial domain and g^{sp} be the Gaussian with scale σ, then

$$L^{sp} = g^{sp} \circ I. \tag{2.44}$$

The idea of the Harris detector is to find the locations where I has significant changes in both directions [17]. Such points can be found by:

$$\mu^{sp} = g \circ \begin{bmatrix} (L_x^{sp})^2 & L_x^{sp} L_y^{sp} \\ L_x^{sp} L_y^{sp} & (L_y^{sp})^2 \end{bmatrix}, \tag{2.45}$$

where L_x^{sp} and L_y^{sp} are Gaussian derivatives of L^{sp}. For detecting points

$$H^{sp} = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2, \tag{2.46}$$

is used, where λ_1 and λ_2 are eigenvalues of μ^{sp}. By setting k value and finding positive local maxima of H^{sp} the interest points can be detected. $k = 0.04$ is a commonly used value in the literature [17].

Interest Points in the Spatio-Temporal Domain: Laptev extended this notion into the spatio-temporal domain. Let f be a spatio-temporal image sequence and g be the

Gaussian with scale σ in the spatio-temporal domain, then

$$L = g \circ f, \tag{2.47}$$

where $\circ$ denotes the convolution operator.

Then, μ becomes

$$\mu = g \circ \begin{bmatrix} L_x^2 & L_x L_y & L_x L_t \\ L_x L_y & L_y^2 & L_y L_t \\ L_x L_t & L_y L_t & L_t^2 \end{bmatrix} \tag{2.48}$$

in spatio-temporal domain, where L_x, L_y and L_t are Gaussian derivatives of L. Hence, H is

$$H = \lambda_1 \lambda_2 \lambda_3 - k(\lambda_1 + \lambda_2 + \lambda_3)^3, \tag{2.49}$$

where λ_1, λ_2 and λ_3 are eigenvalues of μ. For sufficiently large values of k, positive local maxima of H correspond to interest points in spatio-temporal domain. Laptev showed $k = 0.005$ is a good value for k.

Scale Selection in Space-Time: Let us assume that the blob that defines the interest point is a spatio-temporal Gaussian blob. Then, convolving a Gaussian with another Gaussian results in a Gaussian with scale equal to the sum of the scales of the operands. Thus, L becomes a Gaussian. We define a normalized spatio-temporal Laplacian operator

$$\nabla^2_{norm} L = \sigma^2 \tau^{1/2} (L_{xx} + L_{yy}) + \sigma \tau^{3/2} L_{tt}, \tag{2.50}$$

where σ is the spatial, and τ is the temporal scale of the Gaussian,L_{xx}, L_{yy}, and L_{tt} are the second order partial derivatives of L. Finding the extrema of $\nabla^2_{norm} L$ over spatial and temporal scales, helps to estimate the spatio-temporal Gaussian extent of

the Gaussian.

Scale Adapted Space-Time Interest Points: We want to detect interest points that are both the maxima of the spatio-temporal corner function H and the extrema of the normalized spatio-temporal Laplace operator $\nabla^2_{norm} L$. Since it is very costly to compute space-time maxima of H for each spatio-temporal scale level and then selecting the ones that maximize $\nabla^2_{norm} L$, we use an alternative setup for sparsely distributed scale values.

First, interest points are detected as the maxima of H (Equation 2.49). Then, we select the neighboring spatio-temporal scale that maximizes $(\nabla^2_{norm} L)^2$. After that, we re-detect the space-time location of the interest point at the new scale.

3. ACTION RECOGNITION VIA KEYPOINT TRACKING

3.1. Outline of the System

Our system uses the trajectories of keypoints for recognizing human actions. Figure 3.1 shows the outline of the action recognition system. In our system, we

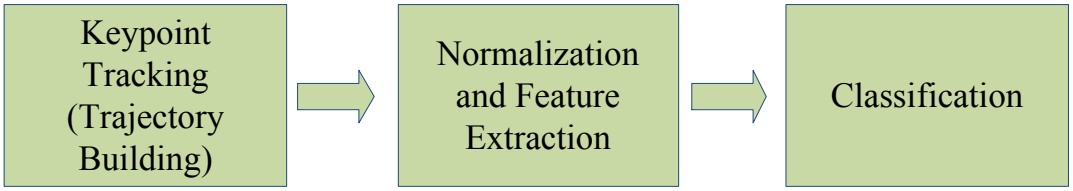

Figure 3.1. Block diagram of the system

first find keypoints in each frame and track their trajectories throughout the video. When the recognition of the action is to be performed, we describe the state using the trajectories at a given time instant t_i. For this purpose, we extract descriptors not only from the trajectories at that time but also from a timespan $[t_{i-n}, t_i]$.

For making our decisions invariant of both the timespan and the spatial position, we do some normalizations before extracting features from the trajectories. In addition, it is not guaranteed that the number of the keypoints belonging to any two different trajectories are equal. Thus, the dimension of our trajectory descriptor must be independent from the number of the keypoints of the trajectory.

Moreover, the equality of the number of the trajectories among different videos is also not guaranteed. In dealing with this problem, we use a bag-of-words approach. We quantize trajectories, and use the histogram of the quantized trajectories in a timespan to describe that timespan.

At the end, we use supervised learning for recognizing the human actions. The

next section explains the details of our keypoint tracking algorithms.

3.2. Generic Keypoint Tracker

Keypoint tracking defined as the task of following a keypoint throughout the frames of a video. A trajectory is the motion history of a keypoint throughout its life time. Some critical points should be considered.

One is to avoid losing track in the existence of short occlusions. A keypoint can disappear in tracking of some frames. This can be the result of self-occlusion or occlusion caused by any other object. In both cases, the keypoint can reappear a few frames later. In a keypoint tracking system, it is important not to lose the track of such keypoints prematurely. In addition, there should be an interpolation system for predicting the missed positions of keypoints.

Another essential feature is the ability of the system to deal with noisy matches. There can be incorrect matches of keypoints. These incorrect matches can cause jumps of keypoints between very distant parts of the video frame. This can be dealt with by disallowing matches having a spatial distance higher than a certain threshold at the time of matching, or by removing such trajectories later.

Another matter is stationary points. Since not every part of a frame of the video is expected to move, tracking of still keypoints can occur. In some systems, stationary points can be used, and in some, they can be discarded for narrowing down the search space.

In each frame, a large number of keypoints are introduced to the system. Some of these keypoints can be continuations of previous trajectories, and some can be the beginnings of new trajectories. As the time progresses, the search space for matching new keypoints can become huge. It is important to take into account that this problem can slow down the system as the time passes. Consequently, it is crucial to eliminate some of the trajectories for narrowing down the search space.

We propose an algorithm called the Generic Keypoint Tracker to deal with the keypoint tracking problem. There are five main modules of the Generic Keypoint Tracker algorithm:

- Keypoint Detection
- Descriptor Extraction
- Keypoint-Trajectory Matching
- Trajectory Updating
- Elimination and Storing

The Generic Keypoint Tracker tracks keypoints in a sequence of images. Even though it has a built-in "Keypoint Matching" module, it is designed for allowing the use of different algorithms in "Feature Detection" and "Descriptor Extraction" modules.

In the next subsections, we first give an outline of the keypoint tracker and then describe it in detail.

3.2.1. Outline of the Generic Keypoint Tracker Algorithm

At the start of each frame, the frame is fed into the Generic Keypoint Tracker which detects keypoints and extracts their descriptors. It then matches newly found keypoints to the trajectories: when a match is detected, the trajectory is updated. Otherwise, a new trajectory is started. At the end, it checks whether a trajectory is mature enough to be used or erroneous to be removed. Figure 3.2 shows the outline of the Generic Keypoint Tracker algorithm. Each newly detected keypoint either becomes a part of a trajectory or starts a new trajectory itself.

If there does not exist any trajectory, a new trajectory is initiated for each of the detected keypoints. The absence of trajectories in a frame may occur in two ways. It may be the first frame. Since, there are no trajectories in the first frame, a new trajectory is initiated for each keypoint of the first frame. Moreover, the absence of trajectories may occur in an intermediary frame, due to the fact that all trajectories

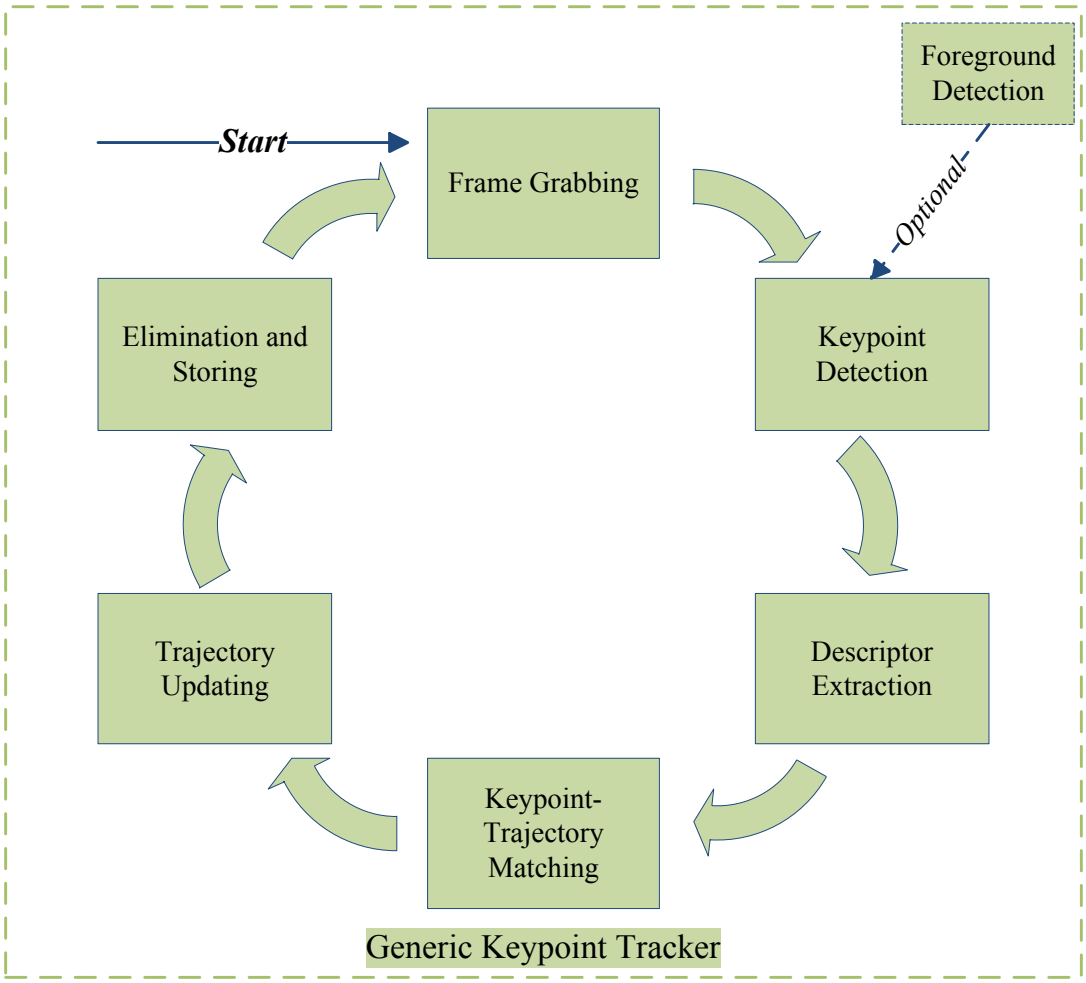

Figure 3.2. Outline of the Generic Keypoint Tracker

may be eliminated in the elimination phase.

Elimination phase follows these operations. Elimination phase is for narrowing down the search space. In our system, stationary points are regarded as insignificant trajectories and can be removed from the search space. Also, there may be some significant trajectories which have not been seen for some time. We want to remove those insignificant trajectories and store the significant but not lately modified trajectories. The remaining trajectories are left in the search space to be evaluated at the next frame.

Optionally, a background subtraction method can be used for decreasing the number of the detected keypoints at the first phase.

Figure 3.2.1 shows some sample frames with corresponding trajectories tracked with the Generic Keypoint Tracker.

3.2.2. Keypoint-Trajectory Matching

Let us start by giving the definitions of a keypoint and a trajectory. We define a keypoint κ as

$$\kappa := (x, y)^T \tag{3.1}$$

where x is the abscissa and y is the ordinate of the keypoint.

The keypoint descriptor vector is defined as

$$\mathbf{s} := (s_1, s_2, \ldots, s_d)^T, \tag{3.2}$$

where d is the dimension of the vector.

Let us define a three tuple, namely keypoint tuple, as

$$\xi := (\kappa, \mathbf{s}, t), \tag{3.3}$$

where $\mathbf{s}$ is the descriptor of the keypoint κ, and t is the time when κ is seen.

Then, we define a trajectory τ as the ordered set of such tuples, and represent it as

$$\tau := \{\xi_i = (\kappa_i, \mathbf{s_i}, t_i) : t_i < t_{i+1}, i \in 1, \ldots, N\} \tag{3.4}$$



Figure 3.3. Falling action trajectory samples from the WeCare dataset

where ξ_i is a keypoint tuple defined in Equation 3.3, and N is the number of keypoint samples forming the trajectory. Note that the value of N can differ between different trajectories.

Trajectory descriptor $\mathbf{g}$ of a trajectory τ is defined to be the average of the descriptors of the last w keypoints of τ, and can be represented as

$$\mathbf{g} := \frac{1}{w} \sum_{i=N-w+1}^{N} \mathbf{s_i}, \tag{3.5}$$

where $1 \leq w \leq N$, and $N = |\tau|$.

First, we calculate "spatial" distances between keypoints and trajectories. Here, we emphasize the word "spatial" due to the fact that we also have "descriptor" distances between keypoints and trajectories. Deducing from the definitions of the trajectory (Equation 3.4), the keypoint tuple (Equation 3.3), and the keypoint (Equation 3.1), $\kappa_N = (x_N, y_N)$ defines the spatial position of the lastly added keypoint to the trajectory τ. Then, we can define the "spatial distance" between the keypoint tuple $\xi = (\kappa, \mathbf{s}, t)$ and the trajectory τ as

$$d^s(\xi, \tau) := \sqrt{(x_N - x)^2 + (y_N - y)^2}, \tag{3.6}$$

where $\kappa = (x, y)$.

We define "descriptor distance" between the keypoint tuple $\xi = (\kappa, \mathbf{s}, t)$ and the trajectory τ as

$$d^d(\xi, \tau) := ||\mathbf{s} - \mathbf{g}||, \tag{3.7}$$

where $\mathbf{g}$ is the descriptor of the trajectory τ.

Let us define the set of keypoint tuples at time t as

$$\mathcal{K}_t := \{\xi_i = (\kappa_i, \mathbf{s_i}, t_i) : t_i = t\}, \tag{3.8}$$

where ξ_i is a keypoint tuple.

We define the set of trajectories at time t as

$$\mathcal{T}_t := \{\tau_i : \tau_i \text{ is a trajectory at time } t\}. \tag{3.9}$$

We define the neighborhood of the trajectory τ at time t^* as

$$\mathcal{N}_{\varepsilon,t^*}(\tau) := \{\xi : \xi \in \mathcal{K}_{t^*}, d^s(\xi, \tau) \leq \varepsilon\}, \tag{3.10}$$

where $\tau \in \mathcal{T}_{t^*}$, and ε is a scalar defining the neighborhood diameter.

We define the neighborhood of the newly detected keypoint tuple $\xi = (\kappa, \mathbf{s}, t^*)$ at time t^* as

$$\mathcal{N}_{\varepsilon,t^*}(\xi) := \{\tau : \tau \in \mathcal{T}_{t^*}, d^s(\xi, \tau) \leq \varepsilon\} \tag{3.11}$$

where ε is a scalar defining the neighborhood diameter.

In Equation 3.10 and Equation 3.11, ε represents the maximum allowed spatial position change of a keypoint between consecutive frames.

In matching, we use a modified version of the method proposed by Lowe [23]. Lowe's method is for matching a keypoint in the first image to one of the keypoints in the second image. First, the "descriptor distances" of the keypoint in the first image to every keypoint in the second image are computed. Then, the ratio of the minimum distance divided by the second minimum distance is calculated. If this ratio is less

than a certain threshold, the matching is done between the keypoint in the first image and the keypoint that gives the minimum distance in the second image.

In our modified version, we first select a newly detected keypoint. Then, we find its matching trajectory. If the matching trajectory passes the threshold condition of Lowe's method, we find that trajectories matching keypoint, again using Lowe's method. If its matching keypoint satisfied the threshold condition, we check whether this keypoint is the same with our initial keypoint. If two keypoints are same, we do the matching.

Let us elaborate the details of the method. First, we select a keypoint tuple, say ξ_i, from $\mathcal{K}_{t^*}$. Then, for each trajectory in the set $\mathcal{N}_{\varepsilon,t^*}(\xi_i)$, we calculate "descriptor distances". Let us define the set of these distances as

$$\mathcal{D}(\mathcal{N}_{\varepsilon,t^*}(\xi_i)) := \{d^d(\xi_i, \tau) : \tau \in \mathcal{N}_{\varepsilon,t^*}(\xi_i)\}. \tag{3.12}$$

Then, the trajectory that gives the minimum distance is denoted

$$\tau_m = \underset{\tau}{\operatorname{argmin}}(\mathcal{D}(\mathcal{N}_{\varepsilon,t^*}(\xi_i))). \tag{3.13}$$

If the ratio of the minimum distance divided by the second minimum distance is less than a threshold λ, then we mark τ_m as a candidate to be matched to ξ_i.

$$\frac{\min(\mathcal{D}(\mathcal{N}_{\varepsilon,t^*}(\xi_i)))}{\min(\mathcal{D}(\mathcal{N}_{\varepsilon,t^*}(\xi_i) \setminus \tau_m))} \leq \lambda \implies \text{Mark } \tau_m, \tag{3.14}$$

where $0 \leq \lambda \leq 1$.

If τ_m is not marked, then continue with the next keypoint tuple in $\mathcal{K}_{t^*}$. Otherwise, let us build another set consisting of "descriptor distances" between τ_m and the

keypoints in the set $\mathcal{N}_{\varepsilon,t^*}(\tau_m)$ as shown in Equation 3.15.

$$\mathcal{D}(\mathcal{N}_{\varepsilon,t^*}(\tau_m)) := \{d^d(\xi, \tau_m) : \xi \in \mathcal{N}_{\varepsilon,t^*}(\tau_m)\}. \tag{3.15}$$

The keypoint tuple that gives the minimum distance is denoted

$$\xi_m = \underset{\xi}{\operatorname{argmin}}(\mathcal{D}(\mathcal{N}_{\varepsilon,t^*}(\tau_m))). \tag{3.16}$$

Thus,

$$\frac{\min(\mathcal{D}(\mathcal{N}_{\varepsilon,t^*}(\tau_m)))}{\min(\mathcal{D}(\mathcal{N}_{\varepsilon,t^*}(\tau_m) \setminus \xi_m))} \leq \lambda \implies \text{Mark } \xi_m, \tag{3.17}$$

where $0 \leq \lambda \leq 1$.

If at the end, ξ_m is marked and is the same with ξ_i , we match $\xi_i = \xi_m$ with τ_m.

3.2.3. Trajectory Updating

After matching keypoints with trajectories, we either add the keypoint to a trajectory or initiate a new trajectory with the keypoint. First, every keypoint that is matched to a trajectory is added at the end of its matching trajectory, causing it to grow. Note that using the previous matching strategy, a keypoint found in a frame can only be added to at most one trajectory. Additionally, this also guarantees that at most one keypoint can be added to a trajectory at a time.

Since we allow trajectories to be occluded for a few frames, when a new keypoint is added to a trajectory, some gaps may occur. We fill these gaps by interpolation.

Let trajectory τ have N elements. Then, the last element is $\xi_N = (\kappa_N, \mathbf{s_N}, t_N)$ according to the definition of the trajectory (Equation 3.4). Let $\xi = (\kappa, \mathbf{s}, t)$ be the

candidate keypoint tuple, where $\kappa = (x, y)$. We need to add $t - t_N - 1$ many interpolated elements to τ before adding the candidate keypoint tuple. We prefer linear interpolation for its simplicity. Thus, we find the line passing through the points $\kappa = (x, y)$ and $\kappa_N = (x_N, y_N)$. The line segment between these points and its length are

$$L = \kappa - \kappa_N = (x - x_N, y - y_N), \tag{3.18}$$

$$R = ||L|| = ||\kappa - \kappa_N|| = \sqrt{(x - x_N)^2 + (y - y_N)^2}, \tag{3.19}$$

respectively.

We divide L into $t - t_N$ many equal length pieces having lengths

$$\Delta = \frac{R}{t - t_N}. \tag{3.20}$$

Then, for each interpolated keypoint i, we find its coordinates as

$$\kappa_{N+i} = (x_i, y_i) = p_2 + \Delta i, \tag{3.21}$$

where $i \in 1, \ldots, t - t_N - 1$ and add $(\kappa_{N+i}, \emptyset, t_{N+i})$ keypoint tuple to the trajectory. Finally, after adding each interpolated element, we add the candidate keypoint tuple ξ to the trajectory.

At the end, for each unmatched keypoint, we initialize a new trajectory, having that keypoint as the starting keypoint.

3.2.4. Elimination and Storing

At the end of each frame iteration, there is an elimination and storing phase. The first task in this phase is checking every trajectory whether they need to be eliminated or not.

At the first step, we mark the trajectories that satisfy at least one of the elimination conditions as a candidate for elimination. These conditions are:

- Bounding box diagonal length of the trajectory(τ) is less than a certain threshold

$$\sqrt{(max(x_i) - min(x_i))^2 + (max(y_i) - min(y_i))^2} \leq \beta, \tag{3.22}$$

where $\kappa_i = (x_i, y_i)$ represents a keypoint of τ, and β is a scalar, namely, the minimum required spread. This condition helps us to eliminate the stationary points and the points that are moving very little. Figure 3.4 shows the bounding box.

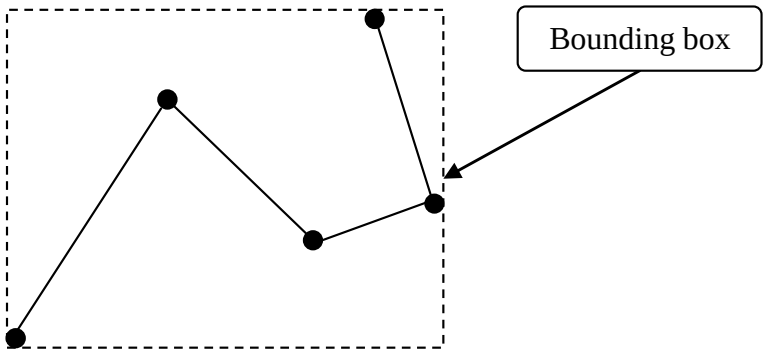

Figure 3.4. Trajectory Bounding Box

- Not enough samples are added to the trajectory since its creation

$$N \leq \sigma, \tag{3.23}$$

where N is the total number of keypoints of τ, and σ is an integer, namely, the minimum required sample count.

We mark these trajectories as candidates for elimination and proceed to the second step.

At the second step, we check unmarked trajectories to see whether they are modified lately or not. If they are not modified for a long time, we store them for using later. Conditions for storing a trajectory τ are

- Not marked as a candidate for elimination, and
- Not modified for a long time

$$t - t_N \geq \nu, \tag{3.24}$$

where t_N is the time stamp of the last keypoint of τ, and ν is an integer, namely, the maximum allowed time since the last observation. ν also defines the maximum length for gaps between consecutive keypoints of trajectories on occlusions.

Trajectories are stored immediately if the conditions are satisfied.

At the last step, we check marked trajectories to see whether they have lived for some time, or not. We do not want to remove a trajectory if it is not mature enough. Conditions for eliminating a trajectory are

- Marked as a candidate for elimination, and
- Has lived for some time

$$t_N - t_1 \geq \rho, \tag{3.25}$$

where t_1 and t_N are the time stamps of the first and the last keypoints of τ, respectively, and ρ is an integer, namely, the minimum age to be eliminated.

We remove stored trajectories from the search space along with the eliminated ones.

3.3. Feature Extraction

In this section, we explain how we extract features for describing videos. Figure 3.5 shows the flow of feature extraction processes.

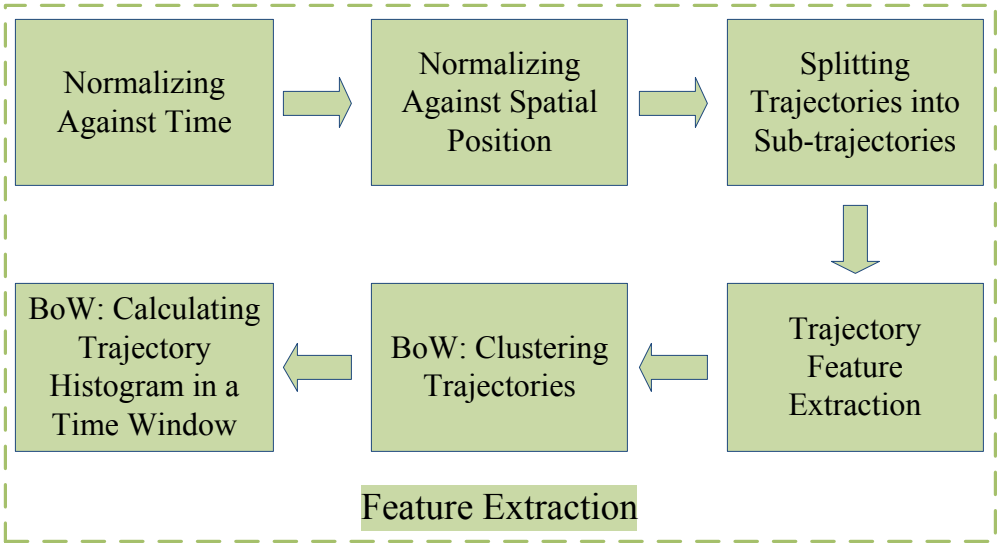

Figure 3.5. Flow of Feature Extraction Processes

3.3.1. Extracting Features from Video Sequences

We define a video sequence as a sequence of images having only one person and one action. We assume that the region of interest is cropped to include only one person and the sequence contains only one action of this person. Cropping the region of interest and segmenting the actions are not parts of our work.

Since we want to recognize the action on a video sequence, a comparison metric between two different video sequences is required. To compare two video sequences, we need to solve three problems:

(i) The first problem is that the lengths of the video sequences may differ. Since we need to compare video sequences, we want to extract the same dimension

descriptors from each video sequence.

(ii) The second problem is that the height and width of the regions of interests of different video sequences may differ. We need to extract trajectories considering this problem and extract descriptors that are invariant to the height and width of the region of interest.

(iii) Another problem occurs when the video sequences are filmed using different zoom levels. An action has different appearance under different zoom levels. We should normalize the scale.

3.3.2. Normalizing Against Time

As the first step in video sequence feature extraction, we normalize against time. Let t_f be the time stamp of the first frame of the video sequence, and t_l be the time stamp of the last frame of the video sequence. Furthermore, let $t_{j,i}$ be the time stamp of the ith element of the trajectory τ_j. We update the time stamp of every element of each trajectory in a video sequence with

$$t_{j,i}^{\text{updated}} = \frac{t_{j,i} - t_f}{t_l - t_f}, \tag{3.26}$$

resulting in $0 \leq t_{j,i}^{\text{updated}} \leq 1$.

3.3.3. Normalizing Against Spatial Position

As the second step, we normalize the positions of the keypoints belonging to the trajectories of the video sequence. Let T be the total number of trajectories in a video sequence, and N_j be the number of elements of the trajectory τ_j where $j \in 1, \ldots, T$. Then, there are a total of

$$N^* = \sum_{j=1}^{T} N_j \tag{3.27}$$

keypoints in the video sequence.

We calculate the mean of the keypoints belonging to the video sequence separately for each dimension as

$$\mu_x = \frac{1}{N^*} \sum_{j=1}^{T} \sum_{i=1}^{N_j} x_{i,j}, \tag{3.28}$$

$$\mu_y = \frac{1}{N^*} \sum_{j=1}^{T} \sum_{i=1}^{N_j} y_{i,j}, \tag{3.29}$$

respectively, where $\kappa_{i,j} = (x_{i,j}, y_{i,j})$ is the ith keypoint of the trajectory τ_j.

Furthermore, we calculate the standard deviation of the keypoints belonging to the video sequence separately for each dimension as

$$\sigma_x = \sqrt{\frac{1}{N^* - 1} \sum_{j=1}^{G} \sum_{i=1}^{N_j} (x_{i,j} - \mu_x)^2}, \tag{3.30}$$

$$\sigma_y = \sqrt{\frac{1}{N^* - 1} \sum_{j=1}^{G} \sum_{i=1}^{N_j} (y_{i,j} - \mu_y)^2}, \tag{3.31}$$

respectively, where $\kappa_{i,j} = (x_{i,j}, y_{i,j})$ is the ith keypoint of the trajectory τ_j.

Then, we update each keypoint $\kappa_{i,j} = (x_{i,j}, y_{i,j})$ by doing Student's t-statistic normalization with

$$\kappa_{i,j}^{\text{updated}} = \left(\frac{x_{i,j} - \mu_x}{\sigma_x}, \frac{y_{i,j} - \mu_y}{\sigma_y} \right). \tag{3.32}$$

In Figure 3.6, asterisks are keypoints of the trajectories, the red circle represents the mean point (μ_x, μ_y), and the grids are aligned to integer multiples of corresponding standard deviations (σ_x and σ_y).

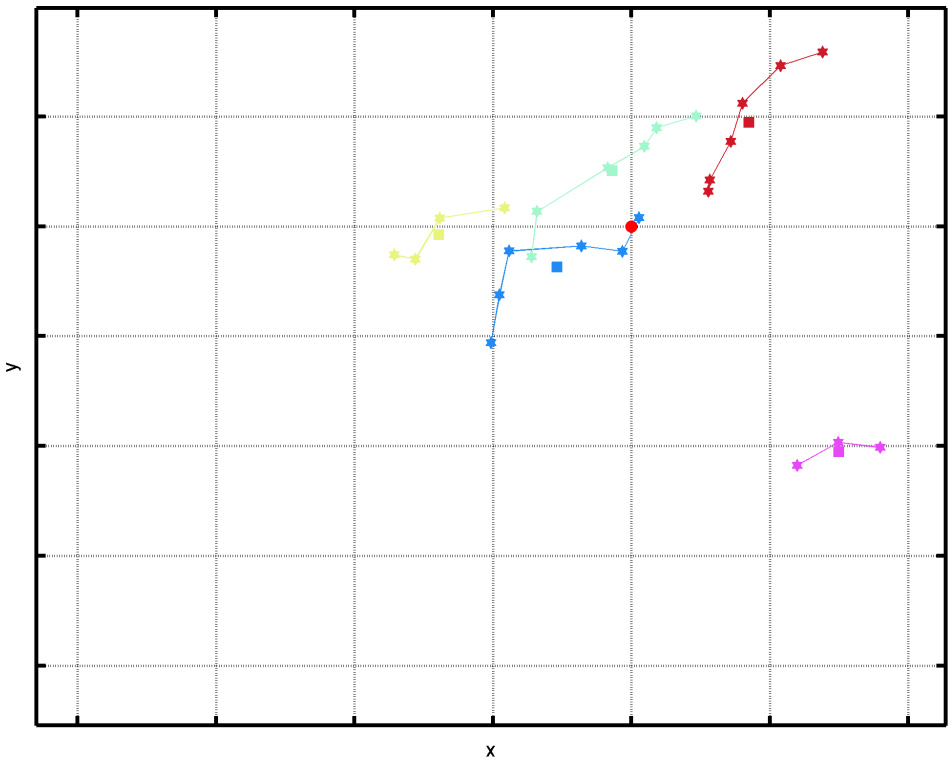

Figure 3.6. Coordinate System for Trajectory Normalization

Since the mean of the keypoints are now translated to the origin, the second problem has been overcome. In addition, we deal with the third problem by scaling the coordinates by the related standard deviations. From now on, we are going to use the updated coordinates.

3.3.4. Extracting Sub-trajectories

After normalizing the time stamps and keypoint coordinates, we extract sub-trajectories from the trajectories. We decide on a length ω and split the trajectories in our sequence of interest into sub-trajectories having at most ω many points. Let τ_j be

a trajectory having N_j elements, then we define sub-trajectories of length ω of τ_j as

$$\begin{aligned}
\tau_{j,1} &:= \{\xi_{i,j} = (\kappa_{i,j}, \mathbf{s_{i,j}}, t_{i,j}) : t_{i,j} < t_{i+1,j}, i \in 1, \ldots, \omega\} \\
\tau_{j,2} &:= \{\xi_{i,j} = (\kappa_{i,j}, \mathbf{s_{i,j}}, t_{i,j}) : t_{i,j} < t_{i+1,j}, i \in 2, \ldots, \omega + 1\} \\
&\vdots \\
\tau_{j,N-\omega+1} &:= \{\xi_{i,j} = (\kappa_{i,j}, \mathbf{s_{i,j}}, t_{i,j}) : t_{i,j} < t_{i+1,j}, i \in N_j - \omega + 1, \ldots, N_j\} \qquad (3.33)
\end{aligned}$$

which are also trajectories, where $\xi_{i,j} \in \tau_j, \forall i \in 1, \ldots, N_j$. Figure 3.7 illustrates some trajectories and their corresponding sub-trajectories.

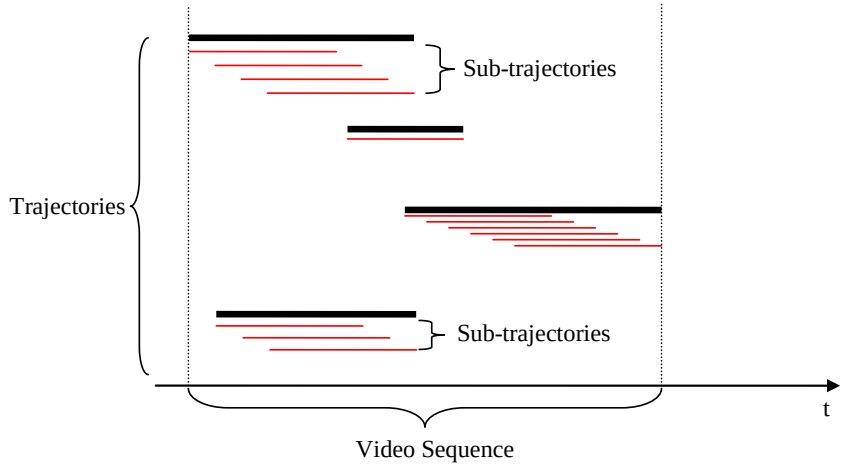

Figure 3.7. Sub-trajectories

3.3.5. Trajectory Feature Extraction

Different trajectories may have different number of elements. However, we want to compare trajectories with each other. Thus, we need to describe trajectories on a common basis. For fulfilling this necessity, we extract some features from the trajectories. Note that we use normalized time stamps and keypoint positions in the calculation of trajectory features. Moreover, we would like to emphasize that we extract features not from the whole trajectories but only from the sub-trajectories. This is possible since the definition of the sub-trajectories in Equation 3.33 states that the sub-trajectories are also trajectories themselves.

Let $\kappa_i = (x_i, y_i)$ be the coordinates of the ith keypoint of a trajectory τ. We first define the vector from κ_1 to κ_i as

$$L_i = \kappa_i - \kappa_1 = (x_i - x_1, y_i - y_1), \forall i \in 1, \ldots, N \tag{3.34}$$

where N is the total number of keypoints in τ.

Our first and second trajectory features are the first and second coordinates of L_N, respectively.

$$F_1 = x_N - x. \tag{3.35}$$

$$F_2 = y_N - y. \tag{3.36}$$

Next, we find the distances of keypoints to L_N. Then, pick the keypoint giving the maximum distance. As our third feature, we multiply this distance value by the sign of the side of L_N that this keypoint resides. The following are the detailed calculations for this feature.

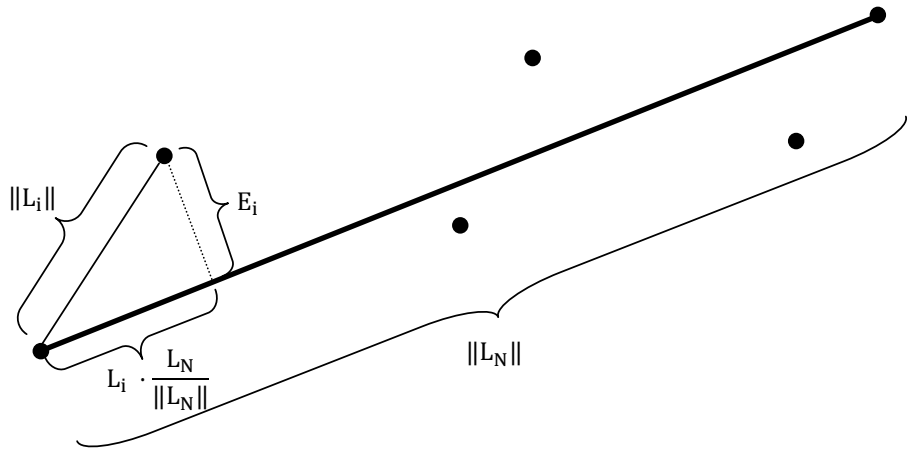

Figure 3.8. L_i, L_N, E_i with trajectory keypoints

The magnitude of the projection of L_i in the direction of L_N is calculated by

$$\begin{aligned} ||p_i|| &= L_i \cdot \frac{L_N}{||L_N||} \\ &= \frac{L_i \cdot L_N}{||L_N||} \\ &= \frac{(x_N - x_1)(x_i - x_1) + (y_N - y_1)(y_i - y_1)}{\sqrt{(x_N - x_1)^2 + (y_N - y_1)^2}} \end{aligned} \tag{3.37}$$

By Pythagorean theorem, we calculate the distance of κ_i to L_N as

$$\begin{aligned} E_i &= \sqrt{||L_i||^2 - (||p_i||)^2} \\ &= \sqrt{(x_i - x_1)^2 + (y_i - y_1)^2 - \left(\frac{(x_N - x_1)(x_i - x_1) + (y_N - y_1)(y_i - y_1)}{\sqrt{(x_N - x_1)^2 + (y_N - y_1)^2}} \right)^2} \\ &= \sqrt{(x_i - x_1)^2 + (y_i - y_1)^2 - \frac{((x_N - x_1)(x_i - x_1) + (y_N - y_1)(y_i - y_1))^2}{(x_N - x_1)^2 + (y_N - y_1)^2}}. \end{aligned} \tag{3.38}$$

We find the keypoint having the maximum value of E_i. The index of that keypoint is

$$j = \underset{i \in 1,\ldots,N}{\operatorname{argmax}} E_i. \tag{3.39}$$

We use the sign of the third component of $L_N \times L_j$ for determining the side of the line that κ_j resides and use the multiplication of this sign with the corresponding error E_j as our third feature.

For calculating the cross product between L_N and L_j, we need to extend our L_i vectors which reside in $2D$ into $3D$. We extend them by adding a third component

with value 0.

$$L_j^{\text{extended}} = (L_{x,j}, L_{y,j}, 0). \tag{3.40}$$

Then, the calculation of the cross product between L_N^{extended} and L_j^{extended} is done as

$$\begin{aligned} L_N^{\text{extended}} \times L_j^{\text{extended}} &= (x_N - x_1, y_N - y_1, 0) \times (x_j - x_1, y_j - y_1, 0) \\ &= (0, 0, L_{N,x}L_{j,y} - L_{j,x}L_{N,y}) \\ &= (0, 0, (x_N - x_1)(y_j - y_1) - (x_j - x_1)(y_N - y_1)). \end{aligned} \tag{3.41}$$

Thus, our third feature becomes

$$F_3 = E_j \operatorname{sgn}\left((x_N - x_1) * (y_j - y_1) - (x_j - x_1) * (y_N - y_1)\right). \tag{3.42}$$

We use the ratio between the magnitude of the projection of L_j in the direction of L_N and the magnitude of L_N as our fourth feature.

$$\begin{aligned} F_4 &= \frac{||p_i||}{||L_N||} \\ &= \frac{(x_N - x_1)(x_i - x_1) + (y_N - y_1)(y_i - y_1)}{(x_N - x_1)^2 + (y_N - y_1)^2}. \end{aligned} \tag{3.43}$$

As fifth and sixth features, we use both x and y coordinates of the mean of the keypoints belonging to the trajectory. We calculate them as

$$F_5 = \frac{1}{N}\sum_{i=1}^{N} x_i, \tag{3.44}$$

$$F_6 = \frac{1}{N}\sum_{i=1}^{N} y_i, \tag{3.45}$$

respectively, where N is the total number of the keypoints belonging to the trajectory.

Color coded filled squares in Figure 3.6 represents the means of the keypoints belonging to the corresponding trajectories.

Our seventh and eighth features are the time codes of the first and last keypoints of the trajectory, respectively.

$$F_7 = t_1. \tag{3.46}$$

$$F_8 = t_N. \tag{3.47}$$

We define a path length function over a trajectory τ as

$$\begin{aligned} \mathrm{pl}_\tau(t) := & \sum_{\substack{t_i \leq t \\ i \in 2,\ldots,N}} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \\ & + \frac{t - t_j}{t_{j+1} - t_j} \sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2}, \end{aligned} \tag{3.48}$$

where $j = \underset{\substack{t < t_i \\ i \in 2,\ldots,N}}{\operatorname{argmin}}(t_i - t)$.

Then, we set our ninth and tenth features as the path length from start of the trajectory to the half-time $\left(\frac{t_N - t_1}{2}\right)$, and the remaining path length, respectively.

$$F_9 = \mathrm{pl}_\tau \left(\frac{t_N - t_1}{2}\right). \tag{3.49}$$

$$F_{10} = \mathrm{pl}_\tau(t_N) - \mathrm{pl}_\tau \left(\frac{t_N - t_1}{2}\right). \tag{3.50}$$

3.3.6. Bag-of-Trajectories

We calculate features for each sub-trajectory of the trajectories of the video sequence. Our problem was comparing two video sequences. It is possible to have different number of sub-trajectories for different video sequences. We use the bag-of-words

method for getting the same dimension descriptors for different video sequences.

First, we normalize the features of each trajectory on our training data. Then, we find cluster centers using K-Means clustering method on these data. Later, we assign each sub-trajectory of the video sequence to these cluster centers. Then, we count the number of sub-trajectories assigned to each cluster. We use this histogram to describe the video sequence.

3.3.7. Classification

After extracting bag-of-trajectories histograms, we use the descriptors of the training data to train a supervised classifier. We use k-NN and SVM as classifiers. When a new video sequence is to be classified, first the bag-of-trajectories histogram of the sequence is found using the previously calculated cluster centers. Then, we find the label of this histogram using the trained classifier.

In the next section, we give information about the datasets used, the experiment setups, and the results of the experiments.

4. EXPERIMENTS

4.1. Datasets

In this study, we used three different datasets to measure the performance of our method under different conditions. The details of these datasets are explained in this section.

4.1.1. KTH Human Action Dataset

KTH Human Action Dataset was presented in 2004 by Schuldt et al. [2]. There are six actions in this dataset:

- Walking
- Jogging
- Running
- Boxing
- Hand waving
- Hand clapping

All actions are performed 12-16 times by 25 subjects. The resolution of the sequences is 160 x 120 pixels having a length of four seconds on average. They are filmed at 25 frames per second over homogeneous backgrounds with a static camera.

The dataset consists of a total of 2391 video sequences in four different scenarios:

- Outdoors
- Outdoors with scale variation
- Outdoors with different clothes
- Indoor



Figure 4.1. Sample frames from the KTH dataset [2]

There are a total of 227637 frames in the dataset. The shortest sequence has 24 frames, and the longest sequence has 362 frames. The average number of frames among all sequences is approximately 95.

The authors divided the dataset into three subsets with respect to the subjects: a training set including the sequences of eight persons, a validation set also including the sequences of eight persons, and a test set including the sequences of nine different persons.

4.1.2. University of Rochester Activities of Daily Living Dataset

University of Rochester Activities of Daily Living Dataset (URADL) was introduced to the literature in 2009 by the work of Messing et al. [3].

The dataset includes 10 actions, which are

- Answering a phone
- Chopping a banana
- Dialing a phone
- Drinking water
- Eating banana
- Eating snack chips
- Looking up a phone number in a phone book
- Peeling a banana
- Using silverware
- Write a phone number on a white board

The activities are performed by five different people. The subjects are selected to have different shapes, sizes, genders, and ethnicities. Every performer repeats each activity three times. Hence, the dataset has a total of 150 videos.

Each video has a resolution of 1280 x 720 pixels. They are filmed at 30 frames



Figure 4.2. Sample frames from the URADL dataset [3]

per second. A tripod-mounted camera located approximately two meters away from the scene is used for shooting.

The dataset consists of 72729 frames, having approximately 485 frames in average per sequence. The shortest video sequence has 104 frames, and the longest one has 1506.

The authors did not separate a subset of the dataset as a test set, instead they used leave one person out method for each person, and reported the average accuracy.

4.1.3. WeCare Dataset

As a part of this work, we introduce a new dataset for the recognition of human actions, called WeCare. The dataset is collected as a part of a wireless sensor network project to implement an elderly care system. It is the abbreviation of Wireless Sensor Network Enabled Care [4]. WeCare is a multi-modal dataset, other than the video data it also includes accelerometer data. The primary objective of this dataset is to advance the fall detection module of the system. Hence, it is designed to include a set of actions that are expected to be confused with fall either by accelerometer data or video data.

Every video in the dataset is a series of actions performed continuously. There are 12 actions in the sequence. The ordered list of action sequences in a video is

(i) Walking from the outside to the front of the armchair
(ii) Jumping
(iii) Sitting on the armchair
(iv) Standing up from the armchair
(v) Walking to the front of the gym mat
(vi) Lying on the gym mat
(vii) Standing up from the gym mat
(viii) Walking to the front of the armchair

(ix) Falling onto the armchair
(x) Standing up from the armchair
(xi) Walking to the front of the gym mat
(xii) Falling onto the gym mat

Even though there are 12 action sequences in a video, eight distinct actions exist. The set of actions in the dataset with corresponding action sequences are

- Walking: action sequences i, v, viii, xi
- Jumping: action sequence ii
- Sitting on the armchair: action sequence iii
- Standing up from the armchair: action sequences iv, x
- Lying on the gym mat: action sequence vi
- Standing up from the gym mat: action sequence vii
- Falling onto the armchair: action sequence ix
- Falling onto the gym mat: action sequence xii

The data of the WeCare dataset was collected using a single 3-axis accelerometer located at the chest of the performer, and two cameras surveying the scene from two different viewing angles. Network cameras were used in collecting the dataset. For reducing the network load, the resolutions of the videos were set to 320 x 240 pixels. The frame rate is 20 frames per second.

There are six performers in the dataset. Every performer repeated the previously defined series of actions 10 times. Therefore, this resulted in 120 action sequences per performer, and 720 action sequences in total.

The sequences of two of the performers are left out to be used as a test set. Hence, there are 240 sequences for testing. The sequences of the other four performers are used for training and validation. The total number of sequences left for training and validation sets is 480.



Figure 4.3. Sample frames from the WeCare dataset [4]

4.2. Experiment Setup

In this section, we give information about some basic methods and tools that are used in the validation of our method. Then, we describe the training, validation and test sets of the datasets we use. At the end, we give information about the parameters and the tested parameter ranges for each of the dataset.

In *K-fold cross validation*, the dataset is divided into K equal parts. Then, we use one part for forming the validation set and combine the remaining $K-1$ parts for forming the training set. Doing this for every distinct part, we get K many different validation and training set pairs. We do the training on the formed training set and validate the method on the related validation set. Then, we get K many validation results which we use together for validating the method. We use stratification on forming the folds, for maintaining the class proportions on each fold.

Confusion matrix is a matrix which shows relations between the predictions and actual classes. An entry $a_{i,j}$ of a confusion matrix shows the number of samples which actually belong to class j but predicted as class i. It is easy to see if the system confuses two classes with each other using a confusion matrix. Table 4.1 shows the structure of a confusion matrix.

Table 4.1. Confusion matrix

		Actual Class			
		C_1	C_2	$\cdots$	C_n
Prediction	C_1	$a_{1,1}$	$a_{1,2}$	$\cdots$	$a_{1,n}$
	C_2	$a_{2,1}$	$a_{2,2}$	$\cdots$	$a_{2,n}$
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	C_n	$a_{n,1}$	$a_{n,2}$	$\cdots$	$a_{n,n}$

Accuracy is the ratio of correctly classified sample count to the total number of samples. It can be calculated from the confusion matrix by dividing the trace of the

confusion matrix to the sum of all values in it. It is equivalent to 1-error rate.

4.2.1. Training, Validation and Test sets

KTH dataset: Schuldt et al. [2] separated the KTH dataset into training, validation and test sets according to the persons. Their training set has eight persons which are 11, 12, 13, 14, 15, 16, 17, 18; validation set also has eight persons which are 19, 20, 21, 23, 24, 25, 01, 04; and the test set has nine persons which are 22, 02, 03, 05, 06, 07, 08, 09, 10. We combined the training and validation sets of Schuldt et al. and used 10-fold cross validation on this combined set. We leave the test set as is and used it on our tests which are reported in Section 4.4.

URADL dataset: Messing et al. [3] used leave-one-person out method in their tests on the URADL dataset and repeated this for each person. There are five persons in the dataset, so they have five folds. They did not separate a test set, since the amount of data is relatively small. We use the setup of Messing et al. as is in our tests.

WeCare dataset: We separate the six persons of the WeCare dataset into two groups: a combined training and validation set of four persons, and a test set of two persons. We use 10-fold cross validation on the combined training and validation set. We report our tests on the test set in Section 4.4.

4.2.2. Parameters

Recalling from Section 3.2, we have six parameters for the Generic Keypoint Tracker which are:

- w of Equation 3.5, window size for trajectory descriptor calculation
- β of Equation 3.22, the minimum required spread
- σ of Equation 3.23, the minimum required sample count
- ν of Equation 3.24, the maximum allowed time since last observation
- ρ of Equation 3.25, the minimum age to be eliminated

- ε of Equation 3.11, the maximum allowed spatial position change of a keypoint between consecutive frames
- λ of Equation 3.14, threshold for matching ratio

We also have parameters for feature extraction and learning:

- ω of Equation 3.33, the maximum number of keypoints for a sub-trajectory
- Normalization method before bag-of-trajectories
- Cluster count K for K-means of bag-of-trajectories
- SVM cost parameter C or k-NN k value depending on the method. Note that, all SVM tests in this work use χ^2 kernel.

For normalization before bag-of-trajectories, we tried two different normalization methods:

- Student's t-statistic normalization: First, mean and standard deviation values for each of the ten features of the trajectories in the training data are found. Then, from each trajectory feature the corresponding mean value is subtracted and the result is divided by the corresponding standard deviation.
- Min-max normalization: First, minimum and maximum values for each of the ten features of the trajectories in the training data are found. Then, from each trajectory feature the corresponding minimum value is subtracted and the result is divided by the difference of corresponding maximum and minimum values.

Table 4.2 presents the tried feature extraction, learning and the Generic Keypoint Tracker parameters for validation.

4.3. Validation and Parameter Selection

In this section, we discuss the effects of different pairs of parameters on the accuracy. In all our experiments, we use SURF with Hessian threshold 300, two octaves and two octave layers for keypoint detection and descriptor extraction. First, we inves-

Table 4.2. Validation parameters

Dataset	Parameter Name	Parameter Values
All datasets	Sub-trajectory length ω	2, 4, 5, 8, 10
	Normalization	t-statistic, min-max
	Cluster count K	1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000
	SVM cost C	0.1, 0.4, 1.6, 6.4, 25.6, 102.4, 409.6, 1638.4, 6553.6, 26214.4, 104857.6
	k-NN k	All integers between 1 and 32
	w	1
	λ	1
KTH	β	15
	σ	1
	ν	2
	ρ	3
	ε	45
URADL	β	40, 60
	σ	1, 5, 8, 10, 12, 15
	ν	0, 2, 4, 5
	ρ	3, 5, 8, 12, 15
	ε	40
WeCare	β	30
	σ	1
	ν	2
	ρ	3
	ε	60

tigate the effects of the keypoint tracker parameters. The experiments on the keypoint tracker parameters are done on the URADL dataset. We try different combinations of keypoint tracker parameters to see their cross effects on accuracy. The experience we gain at these experiments help us fixing the parameter sets for learning parameters.

In addition to the experiments on keypoint parameters, we also have experiments on learning parameters. All cross-combinations of ten different values of bag-of-trajectories word count, five different values of maximum sub-trajectory lengths, and two types of normalizations are experimented on all of the three datasets. For all parameter combinations, SVMs are trained for 11 values of the cost parameter and k-NNs are trained with 32 different k values. Thus, for a fixed set of keypoint tracker parameters, we have 4300 experiments on learning parameters for each dataset. We performed a total of 23994 experiments for investigating the effects of the parameters.

4.3.1. Effects of the Keypoint Tracker Parameters

In this section, we investigate the effects of the keypoint tracker parameters on accuracy. SVM with $\chi 2$ kernel is used for obtaining the results of this section. The cost parameter of SVM is fixed to $C = 1.6$. Student's t-statistic normalization is used in normalization step. The maximum number of keypoints for a sub-trajectory is fixed to $\omega = 4$. The minimum required spread β and the maximum allowed spatial position change of a keypoint between consecutive frames ε are fixed to 40. The bag-of-trajectories cluster count is fixed to $K = 1000$, the minimum age to be eliminated is fixed to $\rho = 15$ in the majority of the tests. For those tests that these last two parameters are not fixed, their values are additionally denoted in the related tests. Different values of the minimum required sample count σ, and the maximum allowed time since the last observation ν are used in the tests.

When we have smaller ν, using larger and smaller σ values perform similar to each other with a larger σ being slightly favorable. When we relax the constraint on ν by letting it obtain larger values, lower values of σ perform better than the higher values. For smaller values of ν, the accuracy increases very slowly when σ increases but

this increase results in a rapid decrease. For moderate values of ν, having too small or too large σ results in degraded accuracy. Figures 4.4 and 4.5 shows these phenomena. The related results can also be observed in Tables A.1 and A.2.

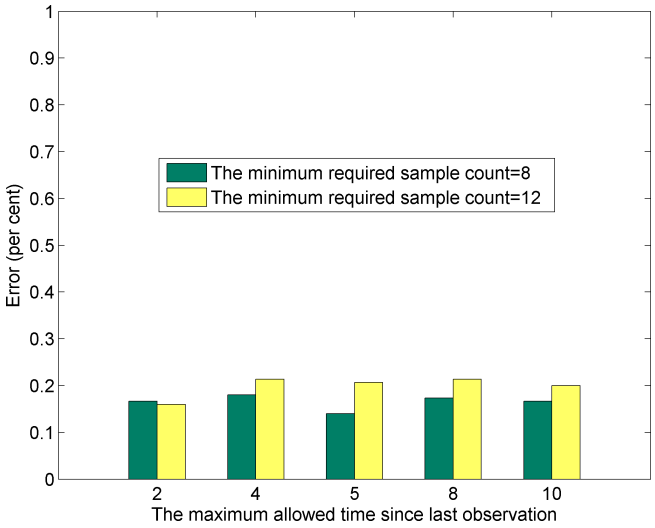

Figure 4.4. The effect of ν and σ to accuracy

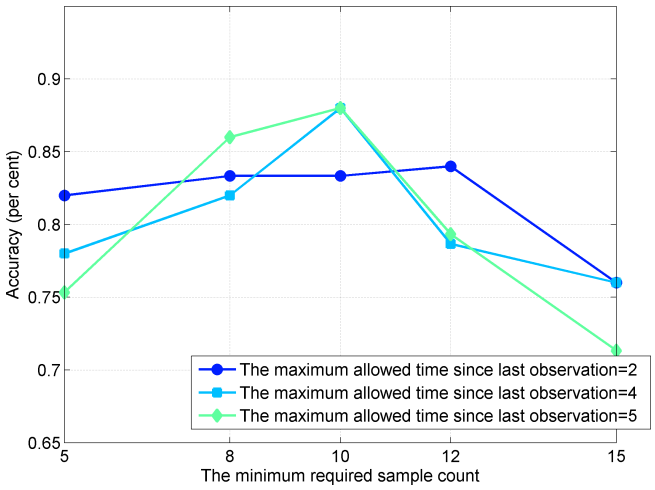

Figure 4.5. The effect of σ and ν to accuracy

For smaller values of σ, increasing bag-of-trajectories size is preferable because of the increase in accuracy. However, as σ gets larger increasing bag-of-trajectories size does not necessarily mean a higher accuracy; in fact the accuracy starts to decrease after some value of K. In Figure 4.6, we show a sample of this behavior. In addition,

Table A.3 shows the numerical values associated with Figure 4.6.

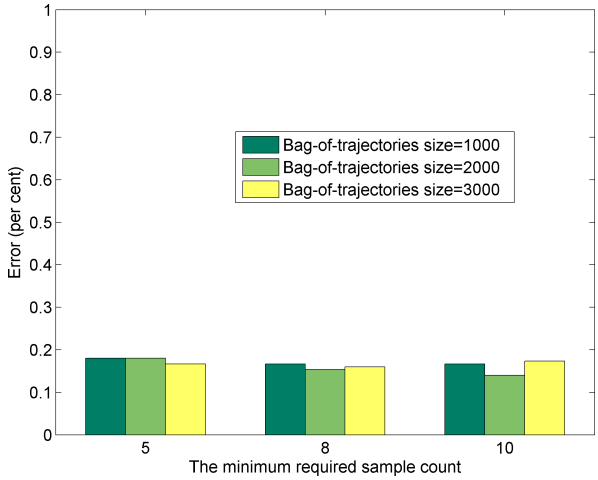

Figure 4.6. The effect of cluster count K and σ to accuracy

Figure 4.7 and Table A.4 display the effect of the change in ρ on accuracy. Manipulation of the parameter ρ induces a small but insignificant variance on accuracy.

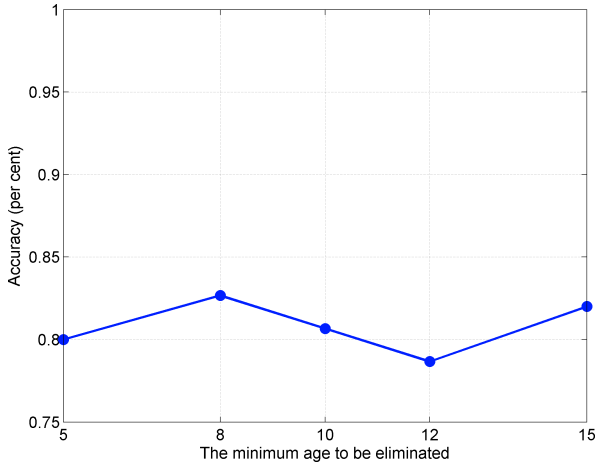

Figure 4.7. The effect of ρ to accuracy

4.3.2. Effects of the Learning Parameters

In this section, we investigate the cross effects of the learning parameters on the accuracy. We show the results on the validation set of the URADL dataset to

be consistent with the results related to the effects of the keypoint tracker parameters. We fixed the keypoint tracker parameters as $w = 1$, $\beta = 40$, $\sigma = 10$, $\nu = 2$, $\rho = 15$, $\varepsilon = 40$, $\lambda = 1$ in these tests. Then, we do tests for cross combinations of $\omega \in \{2, 4, 5, 8, 10\}$, $K \in \{1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000\}$, normalization $\in \{\text{t-statistic}, \text{min-max}\}$.

We try SVM with cost parameter $C \in \{0.1, 0.4, 1.6, 6.4, 25.6, 102.4, 409.6, 1638.4, 6553.6, 26214.4, 104857.6\}$, we also try k-NN with $k \in \{1, \ldots, 32\}$. We see that

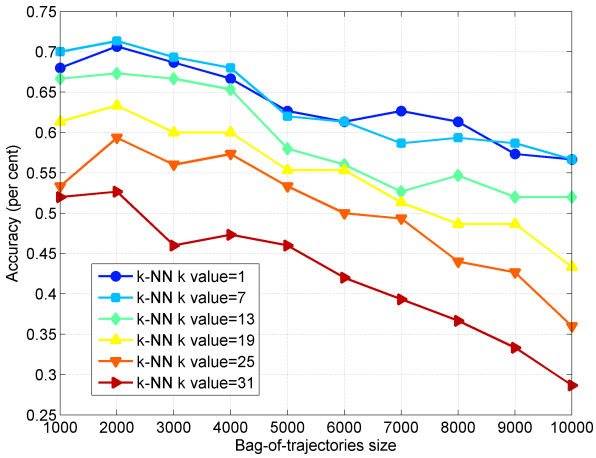

Figure 4.8. The effect of cluster count K and k-NN k value to accuracy

if we use k-NN for training, an increase in bag-of-trajectories size means a decrease in accuracy for almost all k values we tried. In Figure 4.8, we show this phenomenon. Moreover, we also see a drop in accuracy when the k value increases. If we investigate further, we see a peek in the accuracy at $k = 7$, then it decreases again for further values. In addition, we show the numerical values of these accuracies in Table A.5.

When we cross compare the change in bag-of-trajectories size and sub-trajectory length, we see that increasing the maximum sub-trajectory length threshold does not always mean an increase in accuracy. In Figure 4.9, we show this comparison. For some values of K, greater sub-trajectory length results in lower accuracy. In addition, increasing K not necessarily increases accuracy. Table A.6 gives detailed values of this comparison.

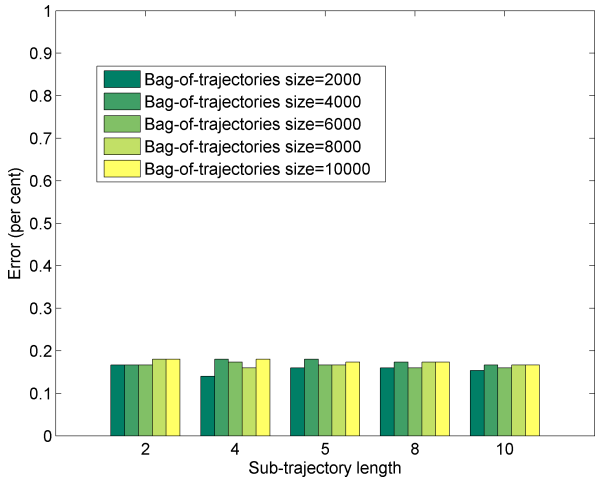

Figure 4.9. The effect of cluster count K and sub-trajectory length ω to accuracy

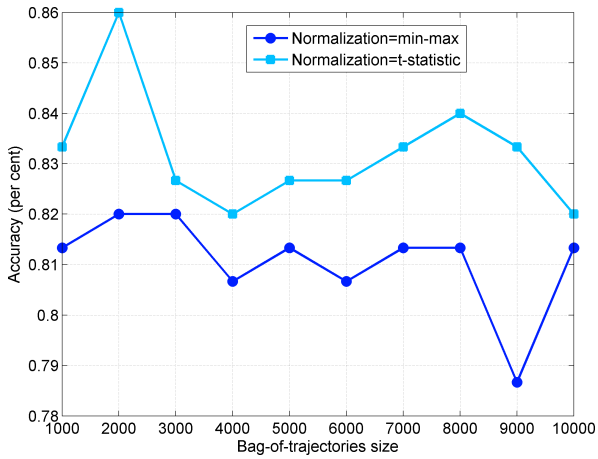

Figure 4.10. The effect of cluster count K and normalization to accuracy

We also observe that t-statistic normalization is better then min-max normalization for any value of bag-of-trajectory size when we use SVM for training. Figure 4.10 shows this result. Here, the value of SVM cost is set to 1.6 and sub-trajectory length is set to 4. Table A.7 shows some accuracies of these experiments. If we change the sub-trajectory length, we see that for some values of sub-trajectory length, min-max normalization can over perform t-statistic normalization. In Figure 4.11, it is seen that for a maximum sub-trajectory length of eight, min-max normalization is better. Table A.11 shows the related accuracies.

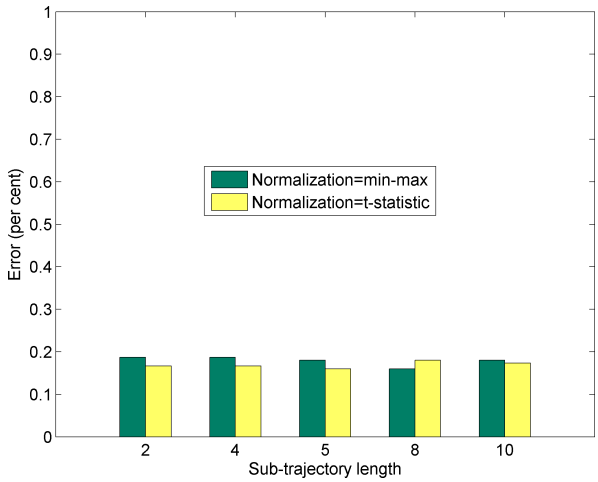

Figure 4.11. The effect of sub-trajectory length ω and normalization to accuracy

For low values of SVM cost, increasing bag-of-trajectory size is preferable since it means an increase in accuracy. However, when SVM cost exceeds a value, in our case 1.6, increasing bag-of-trajectories is not a good choice. In Figure 4.12, we also see increasing SVM cost further does not bring any increase in accuracy. Table A.9 show the comparison of bag-of-trajectory size and SVM cost parameters.

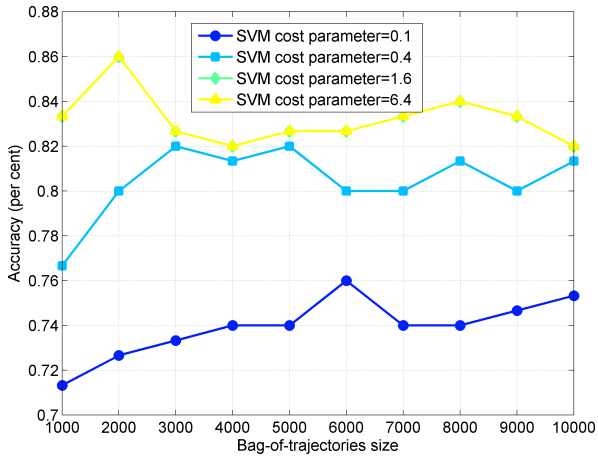

Figure 4.12. The effect of cluster count K and SVM cost C to accuracy

For almost all k values for k-NN, smaller sub-trajectory lengths are preferable. In Figure 4.13, we show this effect for some k values and in Table A.8 we give numerical values.

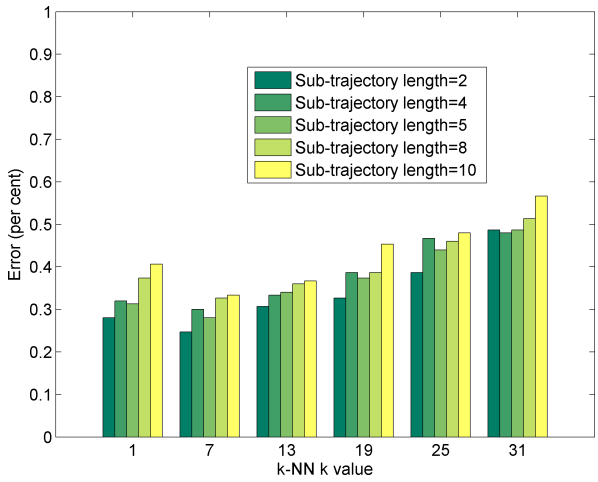

Figure 4.13. The effect of k-NN k value and sub-trajectory length ω to accuracy

Unlike the SVM case, min-max normalization gives better accuracies than t-statistic normalization with k-NN training. We give detailed information in Figure 4.14 and Table A.10.

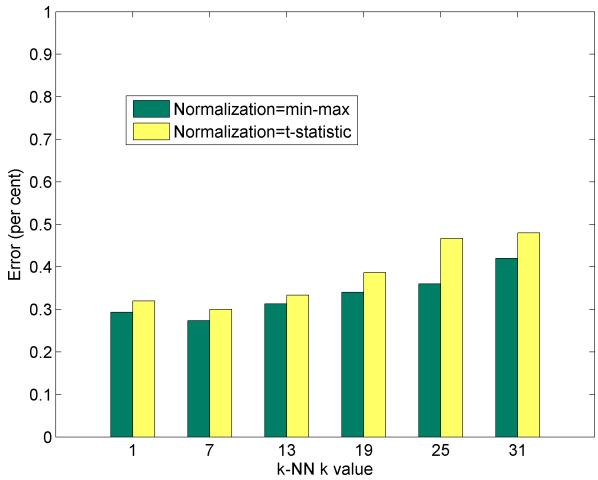

Figure 4.14. The effect of k-NN k value and normalization to accuracy

The change in sub-trajectory length does not result in a uniform change in accuracy. This trend can also be observed in Figure 4.15. The saturation point of SVM cost parameter in terms of accuracy differs when using different sub-trajectory lengths. However, the saturation point of accuracy never requires the SVM cost parameter to be

larger than approximately 6.4. The details of the results are given in Table A.12. For

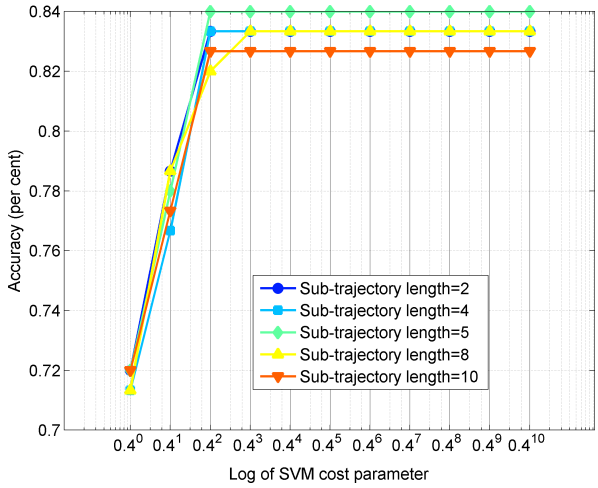

Figure 4.15. The effect of SVM cost C and sub-trajectory length ω to accuracy

low values of SVM cost, t-statistic normalization is better than min-max normalization. When the SVM cost increases, min-max normalization has a more rapid increase than t-statistic normalization and outperforms it on middle values. Nevertheless, t-statistic normalization outperforms min-max normalization at the saturation point. Figure 4.16 and Table A.13 shows this phenomenon.

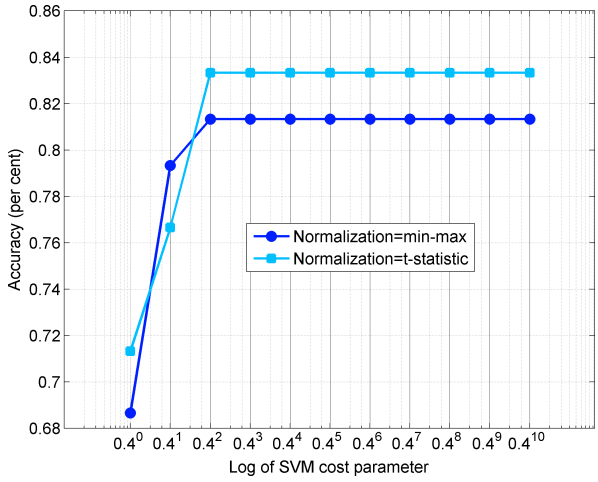

Figure 4.16. The effect of SVM cost C and normalization to accuracy

4.4. Recognition results on the test sets

In this section, we select parameter sets based on our tests on validation sets for each dataset and use the selected parameters during the testing stage. We select the parameter sets giving the maximum validation accuracy. Then, we used those parameters to train the system on the combined set of training and validation sets. For KTH and WeCare datasets, the results that we report in this section are obtained on test sets of the related dataset. However, for making our results comparable to the results of Messing et al. [3], we report the results of the URADL dataset as the average of each fold of the validation set.

The test set results of the KTH dataset are obtained using the parameters: $w = 1$, $\beta = 15$, $\sigma = 1$, $\nu = 2$, $\rho = 3$, $\varepsilon = 45$, $\lambda = 1$, $\omega = 4$, normalization=t-statistic, $K = 6000$, $C = 1.6$. The training is done using SVM with χ^2 kernel. We get 87.25 per cent accuracy using these parameters. The confusion matrix of the test results of the KTH dataset is given in Table 4.3.

Table 4.3. Confusion matrix of the results on the test set of the KTH dataset. The classes are C_1: walking, C_2: jogging, C_3: running, C_4: boxing, C_5: hand clapping, C_6: hand waving

		Actual Class					
		C_1	C_2	C_3	C_4	C_5	C_6
Prediction	C_1	130	8	0	0	0	0
	C_2	14	126	26	0	0	0
	C_3	0	10	118	0	0	0
	C_4	0	0	0	139	15	5
	C_5	0	0	0	4	129	28
	C_6	0	0	0	0	0	111

As can be seen from the confusion matrix of the KTH results, jogging action is confused with walking and running actions. Running action is confused with the jogging action. These results are expected since they do not differ very much except from their speed. The boxing action is confused with the hand clapping action, and vice versa. In addition, the hand clapping and the hand waving actions are also confused with each other. These results arise from the fact that they involve lots of arm movement.

The reported results of the URADL dataset are obtained using the parameters: $w = 1$, $\beta = 40$, $\sigma = 10$, $\nu = 4$, $\rho = 15$, $\varepsilon = 40$, $\lambda = 1$, $\omega = 4$, normalization=t-statistic, $K = 1000$, $C = 1.6$. The training is done using SVM with χ^2 kernel. We get 88 per cent accuracy with these parameters. The confusion matrix of the results of the URADL dataset is given in Table 4.4.

The test set results of the WeCare dataset are obtained using the parameters: $w = 1$, $\beta = 30$, $\sigma = 1$, $\nu = 2$, $\rho = 3$, $\varepsilon = 60$, $\lambda = 1$, $\omega = 8$, normalization=min-max, $K = 2000$, $C = 1.6$. The training is done using SVM with χ^2 kernel. We get 98.75 per cent accuracy using this setting. The confusion matrix of the test results of the WeCare dataset is given in Table 4.5. As can be seen from the confusion matrix falling onto the armchair action is confused with sitting on the armchair action and lying on the gym mat action is confused with falling onto the gym mat and walking actions.

Table 4.4. Confusion matrix of the results on the URADL dataset. The classes are C_1: answer phone, C_2: chop banana, C_3: dial phone, C_4: drink water, C_5: eat banana, C_6: eat snack, C_7: lookup in phonebook, C_8: peel banana, C_9: use silverware, C_{10}: write on whiteboard

		Actual Class									
		C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
Prediction	C_1	15	0	1	0	0	0	0	0	0	0
	C_2	0	14	0	0	0	0	0	0	0	0
	C_3	0	1	14	0	2	0	0	1	0	0
	C_4	0	0	0	14	0	0	0	1	0	0
	C_5	0	0	0	0	12	0	0	2	0	0
	C_6	0	0	0	0	0	12	0	2	0	0
	C_7	0	0	0	0	0	2	14	0	1	0
	C_8	0	0	0	1	1	1	0	8	0	0
	C_9	0	0	0	0	0	0	1	1	14	0
	C_{10}	0	0	0	0	0	0	0	0	0	15

Table 4.5. Confusion matrix of the results on the test set of the WeCare dataset. The classes are C_1: walking, C_2: jumping, C_3: sitting on the armchair, C_4: standing up from the armchair, C_5: lying on the gym mat, C_6: standing up from the gym mat, C_7: falling onto the armchair, C_8: falling onto the gym mat

		Actual Class							
		C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
Prediction	C_1	80	0	0	0	1	0	0	0
	C_2	0	20	0	0	0	0	0	0
	C_3	0	0	20	0	0	0	1	0
	C_4	0	0	0	40	0	0	0	0
	C_5	0	0	0	0	18	0	0	0
	C_6	0	0	0	0	0	20	0	0
	C_7	0	0	0	0	0	0	19	0
	C_8	0	0	0	0	1	0	0	20

5. CONCLUSIONS

In this study, we present a new system for computer vision-based recognition of human actions. The proposed system uses videos as input. Our approach is invariant of the appearance of the person, partial occlusions including self-occlusions and some viewpoint changes. Even though it does not require background subtraction and accurate localization of people, it can benefit from these. In addition, it is robust against temporal length variations, since we normalize the trajectories against time. Furthermore, it is invariant of the location of the action and zoom levels, since the trajectories are normalized against spatial position.

The system has three main steps for recognizing actions in videos. First, we track keypoints, also known as interest points, throughout time. The tracking stage has major challenges. It must be robust and handle noise. The trajectories of tracked keypoints are used for interpreting the human action in the image sequence.

In the second step, features from videos are extracted. First, we divide trajectories into smaller parts, which we call sub-trajectories. We propose ten features for describing a trajectory. Using the trajectory features, sub-trajectories are categorized into clusters. Then, the clusters of these sub-trajectories are used for describing an image sequence. The used image sequence descriptor is the normalized histogram of the clusters of trajectories.

At the final step, the proposed system uses the descriptors of image sequence in a supervised learning approach. The system is trained using some ground truth videos and is tested on other videos.

Using different combinations of the keypoint tracker parameters result in a wide range of recognition accuracies. Depending on different parameter sets, we can have shorter, longer, noisy, a few or a lot of trajectories. These parameters should be fine tuned because of their effect on the recognition rate.

In addition, we introduce a new dataset, called WeCare, to the human action recognition literature which is focused on elderly care systems. The main objective of the dataset is to detect falls of humans. For attaining this goal, some other actions that can be confused with the falling action are included in the dataset. The proposed dataset is multi modal. In addition to video data, the dataset contains accelerometer data. Thus, the actions in the dataset are chosen considering both video and accelerometer data. However, we have not used the accelerometer data in this study.

We use three datasets including our newly proposed one for evaluating our approach. We chose the other two datasets according to their use in the literature. The KTH dataset is a widely used dataset for human action recognition purposes in the literature and the last dataset used in our work, URADL dataset, is used in a similar trajectory based work [3] in the literature.

The performance of our approach is comparable to the methods in the literature. We have 87.25 per cent accuracy on the KTH dataset. The first result using the KTH dataset has an accuracy of 71.72 percent by Schuldt et al. [2]. Niebles et al. [25] has 81.5 per cent accuracy, Laptev et al. [57] has an accuracy of 91.8 per cent, and Messing et al. [3] has 74 per cent accuracy.

We have 88 per cent accuracy on the URADL dataset. Messing et al. [3] tried three methods on this dataset. Their methods have 63 per cent, 67 per cent, and 89 per cent accuracies. However, the last method they used is highly dependent on the position of the action on the frame and it requires the position of the face.

Our performance on the WeCare dataset is 98.75 per cent. In the future, we need to add more challenging scenarios to the WeCare dataset for testing our work.

As future directions, the elimination module of the Generic Keypoint Tracker must be made more robust. The elimination rate of significant trajectories must be reduced and the elimination rate of noisy trajectory must be increased. The recognition performance must be improved. Human action detection must be done in cooperation

with the current human action recognition system. This way, the need for isolated single actioned videos must be overcome.

The system must be made invariant of viewpoint changes. Currently, the system needs to be trained with action samples from different viewpoints for recognition of the same action from different viewpoints. The concept of trajectories with 3D spatial positions must be investigated. This way, we expect to make the system invariant of viewpoint changes.

New trajectory features must be extracted and their performance to the current trajectory features must be evaluated. Effects of different subsets of the current trajectory features must be investigated. The use of a subset of the features is expected to bring the system performance closer to real time.

The system runs close to real time. Improvements and optimizations must be done to enable it to run in real time.

APPENDIX A: TABLES OF THE VALIDATION RESULTS

In this appendix, we present the numerical values of the validation results reported in Section 4.3.

Table A.1. Change of validation results relative to ν and σ

		σ = 8	σ = 12
ν	2	83.33%	84.00%
	4	82.00%	78.67%
	5	**86.00%**	79.33%
	8	82.67%	78.67%
	10	83.33%	80.00%

Table A.2. Change of validation results relative to σ and ν

		ν = 2	ν = 4	ν = 5
σ	5	82.00%	78.00%	75.33%
	8	83.33%	82.00%	86.00%
	10	83.33%	**88.00%**	88.00%
	12	84.00%	78.67%	79.33%
	15	76.00%	76.00%	71.33%

Table A.3. Change of validation results relative to K and σ

		σ		
		5	8	10
K	1000	82.00%	83.33%	83.33%
	2000	82.00%	84.67%	**86.00%**
	3000	83.33%	84.00%	82.67%

Table A.4. Change of validation results relative to ρ

		ρ				
		5	8	10	12	15
ν	2	80.00%	**82.67%**	80.67%	78.67%	82.00%

Table A.5. Change of validation results relative to K and k

Bag-of-trajectories size	k-NN k value					
	1	7	13	19	25	31
1000	68.00%	70.00%	66.67%	61.33%	53.33%	52.00%
2000	70.67%	**71.33%**	67.33%	63.33%	59.33%	52.67%
3000	68.67%	69.33%	66.67%	60.00%	56.00%	46.00%
4000	66.67%	68.00%	65.33%	60.00%	57.33%	47.33%
5000	62.67%	62.00%	58.00%	55.33%	53.33%	46.00%
6000	61.33%	61.33%	56.00%	55.33%	50.00%	42.00%
7000	62.67%	58.67%	52.67%	51.33%	49.33%	39.33%
8000	61.33%	59.33%	54.67%	48.67%	44.00%	36.67%
9000	57.33%	58.67%	52.00%	48.67%	42.67%	33.33%
10000	56.67%	56.67%	52.00%	43.33%	36.00%	28.67%

Table A.6. Change of validation results relative to K and ω

Bag-of-trajectories size \ Sub-trajectory length	2	4	5	8	10
1000	83.33%	83.33%	84.00%	82.00%	82.67%
2000	83.33%	**86.00%**	84.00%	84.00%	84.67%
3000	85.33%	82.67%	84.00%	84.00%	84.67%
4000	83.33%	82.00%	82.00%	82.67%	83.33%
5000	82.67%	82.67%	82.67%	84.00%	84.00%
6000	83.33%	82.67%	83.33%	84.00%	84.00%
7000	82.00%	83.33%	84.00%	84.00%	84.00%
8000	82.00%	84.00%	83.33%	82.67%	83.33%
9000	82.00%	83.33%	83.33%	82.67%	82.00%
10000	82.00%	82.00%	82.67%	82.67%	83.33%

Table A.7. Change of validation results relative to K and normalization

Bag-of-trajectories size \ Normalization	min-max	t-statistic
1000	81.33%	83.33%
2000	82.00%	**86.00%**
3000	82.00%	82.67%
4000	80.67%	82.00%
5000	81.33%	82.67%
6000	80.67%	82.67%
7000	81.33%	83.33%
8000	81.33%	84.00%
9000	78.67%	83.33%
10000	81.33%	82.00%

Table A.8. Change of validation results relative to k and ω

k-NN k value \ Sub-trajectory length	2	4	5	8	10
1	72.00%	68.00%	68.67%	62.67%	59.33%
3	72.00%	69.33%	67.33%	68.00%	65.33%
5	**76.67%**	68.67%	70.67%	64.67%	66.67%
7	75.33%	70.00%	72.00%	67.33%	66.67%
9	72.67%	67.33%	68.67%	66.00%	66.67%
11	70.00%	70.00%	64.00%	62.67%	63.33%
13	69.33%	66.67%	66.00%	64.00%	63.33%
15	66.00%	64.67%	62.67%	64.67%	63.33%
17	68.67%	61.33%	64.00%	64.00%	56.67%
19	67.33%	61.33%	62.67%	61.33%	54.67%
21	66.67%	59.33%	58.67%	57.33%	50.00%
23	62.00%	57.33%	58.67%	54.67%	52.67%
25	61.33%	53.33%	56.00%	54.00%	52.00%
27	58.00%	58.00%	58.67%	56.00%	52.67%
29	57.33%	54.00%	55.33%	52.67%	50.00%
31	51.33%	52.00%	51.33%	48.67%	43.33%

Table A.9. Change of validation results relative to K and C

Bag-of-trajectories size \ SVM cost parameter	0.1	0.4	1.6	6.4	25.6
1000	71.33%	76.67%	83.33%	83.33%	83.33%
2000	72.67%	80.00%	**86.00%**	86.00%	86.00%
3000	73.33%	82.00%	82.67%	82.67%	82.67%
4000	74.00%	81.33%	82.00%	82.00%	82.00%
5000	74.00%	82.00%	82.67%	82.67%	82.67%
6000	76.00%	80.00%	82.67%	82.67%	82.67%
7000	74.00%	80.00%	83.33%	83.33%	83.33%
8000	74.00%	81.33%	84.00%	84.00%	84.00%
9000	74.67%	80.00%	83.33%	83.33%	83.33%
10000	75.33%	81.33%	82.00%	82.00%	82.00%

Table A.10. Change of validation results relative to k and normalization

k-NN k value	Normalization: min-max	Normalization: t-statistic
1	70.67%	68.00%
3	72.00%	69.33%
5	**72.67%**	68.67%
7	72.67%	70.00%
9	72.00%	67.33%
11	67.33%	70.00%
13	68.67%	66.67%
15	67.33%	64.67%
17	67.33%	61.33%
19	66.00%	61.33%
21	62.00%	59.33%
23	62.67%	57.33%
25	64.00%	53.33%
27	61.33%	58.00%
29	59.33%	54.00%
31	58.00%	52.00%

Table A.11. Change of validation results relative to ω and normalization

		Normalization	
		min-max	t-statistic
ω	2	81.33%	83.33%
	4	81.33%	83.33%
	5	82.00%	84.00%
	8	**84.00%**	82.00%
	10	82.00%	82.67%

Table A.12. Change of validation results relative to C and ω

		Sub-trajectory length				
		2	4	5	8	10
SVM cost parameter	0.1	72.00%	71.33%	72.00%	71.33%	72.00%
	0.4	78.67%	76.67%	78.00%	78.67%	77.33%
	1.6	83.33%	83.33%	**84.00%**	82.00%	82.67%
	6.4	83.33%	83.33%	84.00%	83.33%	82.67%
	25.6	83.33%	83.33%	84.00%	83.33%	82.67%
	102.4	83.33%	83.33%	84.00%	83.33%	82.67%
	409.6	83.33%	83.33%	84.00%	83.33%	82.67%
	1638.4	83.33%	83.33%	84.00%	83.33%	82.67%
	6553.6	83.33%	83.33%	84.00%	83.33%	82.67%
	26214.4	83.33%	83.33%	84.00%	83.33%	82.67%
	104857.6	83.33%	83.33%	84.00%	83.33%	82.67%

Table A.13. Change of validation results relative to C and normalization

		Normalization	
		min-max	t-statistic
SVM cost parameter	0.1	68.67%	71.33%
	0.4	79.33%	76.67%
	1.6	81.33%	**83.33%**
	6.4	81.33%	83.33%
	25.6	81.33%	83.33%
	102.4	81.33%	83.33%
	409.6	81.33%	83.33%
	1638.4	81.33%	83.33%
	6553.6	81.33%	83.33%
	26214.4	81.33%	83.33%
	104857.6	81.33%	83.33%

REFERENCES

1. Bay, H., A. Ess, T. Tuytelaars and L. V. Gool, "Speeded-Up Robust Features (SURF)", *Computer Vision and Image Understanding*, Vol. 110, No. 3, pp. 346–359, 2008.

2. Schuldt, C., I. Laptev and B. Caputo, "Recognizing Human Actions: A Local SVM Approach", *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, Vol. 3, pp. 32–36 Vol.3, Ieee, 2004.

3. Messing, R., C. Pal and H. Kautz, "Activity recognition using the velocity histories of tracked keypoints", *IEEE 12th International Conference on Computer Vision*, pp. 104–111, Sep. 2009.

4. Alemdar, H. O., Y. E. Kara, M. O. Ozen, G. R. Yavuz, O. D. Incel, L. Akarun and C. Ersoy, "A robust multimodal fall detection method for ambient assisted living applications", *2010 IEEE 18th Signal Processing and Communications Applications Conference*, pp. 204–207, IEEE, Apr. 2010.

5. Hu, W., T. Tan, L. Wang and S. Maybank, "A Survey on Visual Surveillance of Object Motion and Behaviors", *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, Vol. 34, No. 3, pp. 334–352, 2004.

6. Moeslund, T. B., A. Hilton and V. Krüger, "A survey of advances in vision-based human motion capture and analysis", *Computer Vision and Image Understanding*, Vol. 104, No. 2-3, pp. 90–126, 2006.

7. Poppe, R., "A survey on vision-based human action recognition", *Image and Vision Computing*, Vol. 28, No. 6, pp. 976–990, 2010.

8. Poppe, R., "Vision-based human motion analysis: An overview", *Computer Vision and Image Understanding*, Vol. 108, No. 1-2, pp. 4–18, 2007.

9. Bobick, A. and J. Davis, "The recognition of human movement using temporal templates", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23, No. 3, pp. 257–267, Mar. 2001.

10. Weinland, D. and E. Boyer, "Action recognition using exemplar-based embedding", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–7, Jun. 2008.

11. Efros, A. A., A. C. Berg, G. Mori and J. Malik, "Recognizing action at a distance", *Proceedings Ninth IEEE International Conference on Computer Vision*, Vol. 2, No. October, pp. 726–733, 2003.

12. Tran, D. and A. Sorokin, "Human activity recognition with metric learning", *Computer Vision–ECCV 2008*, , No. Section 2, pp. 548–561, 2008.

13. Lucas, B. and T. Kanade, "An iterative image registration technique with an application to stereo vision", *International joint conference on artificial intelligence*, Vol. 3, pp. 674–679, Citeseer, 1981.

14. Gorelick, L., M. Blank, E. Shechtman, M. Irani and R. Basri, "Actions as space-time shapes.", *IEEE transactions on pattern analysis and machine intelligence*, Vol. 29, No. 12, pp. 2247–2253, Dec. 2007.

15. Batra, D., T. Chen and R. Sukthankar, "Space-Time Shapelets for Action Recognition", *IEEE Workshop on Motion and video Computing*, pp. 1–6, Jan. 2008.

16. Ogata, T., W. Christmas, J. Kittler and S. Ishikawa, "Improving human activity detection by combining multi-dimensional motion descriptors with boosting", *18th International Conference on Pattern Recognition (ICPR'06)*, pp. 295–298, 2006.

17. Laptev, I., "On Space-Time Interest Points", *International Journal of Computer Vision*, Vol. 64, No. 2-3, pp. 107–123, Sep. 2005.

18. Oikonomopoulos, A., I. Patras and M. Pantic, "Spatiotemporal salient points for visual recognition of human actions.", *IEEE transactions on systems, man, and*

cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society, Vol. 36, No. 3, pp. 710–9, Jun. 2006.

19. Dollar, P., V. Rabaud, G. Cottrell and S. Belongie, "Behavior Recognition via Sparse Spatio-Temporal Features", *IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pp. 65–72, 2005.

20. Willems, G., T. Tuytelaars and L. Van Gool, "An efficient dense and scale-invariant spatio-temporal interest point detector", *Computer Vision–ECCV 2008*, pp. 650–663, 2008.

21. Wang, H., M. Ullah, A. Kläser, I. Laptev and C. Schmid, "Evaluation of local spatio-temporal features for action recognition", *British Machine Vision Conference*, 2009.

22. Sun, J., X. Wu, S. Yan, L. Cheong, T. Chua and J. Li, "Hierarchical spatio-temporal context modeling for action recognition", *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 2004–2011, 2009.

23. Lowe, D. G., "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal of Computer Vision*, Vol. 60, No. 2, pp. 91–110, 2004.

24. Viola, P. and M. Jones, "Robust real-time object detection", *International Journal of Computer Vision*, Vol. 57, No. 2, pp. 137–154, 2002.

25. Niebles, J. C., H. Wang and L. Fei-Fei, "Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words", *International Journal of Computer Vision*, Vol. 79, No. 3, pp. 299–318, Mar. 2008.

26. Stauffer, C. and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 246–252, 1999.

27. McKenna, S. J., S. Jabri, Z. Duric, A. Rosenfeld and H. Wechsler, "Tracking Groups of People", *Computer Vision and Image Understanding*, Vol. 80, pp. 42–56, 2000.

28. Haritaoglu, I., D. Harwood and L. S. Davis, "W4: Real-Time Surveillance of People and Their Activities", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, pp. 809–830, 2000.

29. Lipton, A. J., H. Fujiyoshi and R. S. Patil, "Moving Target Classification and Tracking from Real-time Video", *IEEE Workshop on Applications of Computer Vision*, pp. 8–14, 1998.

30. Paragios, N. and R. Deriche, "Geodesic Active Contours and Level Sets for the Detection and Tracking of Moving Objects", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 3, pp. 266–280, 2000.

31. Deutscher, J., A. Blake and I. Reid, "Articulated Body Motion Capture by Annealed Particle Filtering", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 126–133, 2000.

32. Kehl, R., M. Bray and L. V. Gool, "Full Body Tracking from Multiple Views Using Stochastic Sampling", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 2, p. 129, 2005.

33. Maccormick, J. and M. Isard, "Partitioned Sampling, Articulated Objects, and Interface-Quality Hand Tracking", *In Proc. 6th European Conf. on Computer Vision*, pp. 3–19, 2000.

34. Plänkers, R. and P. Fua, "Articulated Soft Objects for Multi-View Shape and Motion Capture", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 9, pp. 1182–1187, 2003.

35. Barr'on, C. and I. A. Kakadiaris, "Monocular Human Motion Tracking", *Multime-*

dia Systems, Vol. 10, pp. 118–130, 2004.

36. Delamarre, Q. and O. Faugeras, "3D Articulated Models and Multiview Tracking with Physical Forces", *Computer Vision and Image Understanding*, Vol. 81, No. 3, pp. 328–357, 2001.

37. Ioffe, S. and D. A. Forsyth, "Probabilistic Methods for Finding People", *International Journal of Computer Vision*, Vol. 43, No. 1, pp. 45–68, 2001.

38. Mori, G., X. Ren, A. A. Efros and J. Malik, "Recovering Human Body Configurations: Combining Segmentation and Recognition", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004.

39. Navaratnam, R., A. Thayananthan, P. H. S. Torr and R. Cipolla, "Hierarchical Part-Based Human Body Pose Estimation", *British Machine Vision Conference*, 2005.

40. Hua, G., M.-h. Yang and Y. Wu, "Learning to Estimate Human Pose with Data Driven Belief Propagation", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, p. 747, 2005.

41. Lee, M. W. and I. Cohen, "Proposal Maps driven MCMC for Estimating Human Body Pose in Static Images", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 00, pp. 334–341, Washington, DC, 2004.

42. Bay, H., T. Tuytelaars and L. V. Gool, "SURF: Speeded Up Robust Features", *European Conference on Computer Vision*, pp. 404–417, 2006.

43. Lowe, D. G., "Object recognition from local scale-invariant features", *International Conference on Computer Vision*, pp. 1150–1157, Published by the IEEE Computer Society, Corfu, Greece, 1999.

44. Crow, F. C., "Summed-area tables for texture mapping", *ACM SIGGRAPH Computer Graphics*, Vol. 18, No. 3, pp. 207–212, Jul. 1984.

45. Evans, C., "Notes on the OpenSURF Library", *University of Bristol, Tech. Rep. CSTR-09-001, January*, , No. 1, 2009.

46. Brown, M., "Invariant features from interest point groups", *British Machine Vision Conference, Cardiff, Wales*, 2002.

47. MacQueen, J., "Some methods for classification and analysis of multivariate observations", *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Vol. 1, p. 14, California, USA, 1967.

48. Alpaydin, E., *Introduction to machine learning*, The MIT Press, 2004.

49. Arthur, D. and S. Vassilvitskii, "k-means++: The advantages of careful seeding", *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035, Society for Industrial and Applied Mathematics, 2007.

50. Vapnik, V. N., *Estimation of dependences based on empirical data*, Springer New York, 2 edn., 2006.

51. Cortes, C. and V. Vapnik, "Support-vector networks", *Machine Learning*, Vol. 20, No. 3, pp. 273–297, Sep. 1995.

52. Schiele, B. and J. Crowley, "Object recognition using multidimensional receptive field histograms", *Computer Vision - ECCV '96, 4th European Conference on Computer Vision*, Vol. 1, pp. 610–619, 1996.

53. Chapelle, O., P. Haffner and V. N. Vapnik, "Support vector machines for histogram-based image classification.", *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, Vol. 10, No. 5, pp. 1055–64, Jan. 1999.

54. Tomasi, C. and T. Kanade, "Detection and tracking of point features", *CMU-CS-91-132, Carnegie Mellon University Technical Report*, Carnegie Mellon University, 1991.

55. Shi, J. and C. Tomasi, "Good features to track", *TR 93-1399, Cornell U.*, IEEE Comput. Soc. Press, 1993.

56. Bouguet, J.-Y., "Pyramidal implementation of the lucas kanade feature tracker description of the algorithm", *Intel Corporation, Microprocessor Research Labs, OpenCV Documents*, Vol. 3, No. 2, pp. 1–9, 1999.

57. Laptev, I., M. Marszalek, C. Schmid and B. Rozenfeld, "Learning realistic human actions from movies", *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, IEEE, 2008.