Human Action Recognition in Videos with Local Representation

Michal Koperski

UNIVERSITÉ COTE D'AZUR

ECOLE DOCTORALE STIC
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE

pour l'obtention du grade de

Docteur en Sciences

de l'Université Cote d'Azur
Mention: INFORMATIQUE

presentée et soutenue par

Michał KOPERSKI

Reconnaissance d'actions humaines dans des vidéos utilisant une représentation locale

Thèse dirigée par: François BRÉMOND

INRIA Sophia Antipolis, STARS

soutenue le 9/11/2017

Jury :

Président :	Frédéric PRECIOSO	-	University of Nice
Rapporteur :	Matthieu CORD	-	UPMC - Sorbonne Universities
Rapporteur :	Leonid SIGAL	-	University of British Columbia, Disney Research
Examinateur :	Jean-Marc ODOBEZ	-	IDIAP
Directeur de thèse :	François BRÉMOND	-	INRIA (STARS)
Invité :	Gianpiero FRANCESCA	-	Toyota Motor Europe

UNIVERSITY COTE D'AZUR

DOCTORAL SCHOOL STIC
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

P H D T H E S I S

to obtain the title of

PhD of Science

of the University Cote d'Azur
Specialty : COMPUTER SCIENCE

Defended by

Michal KOPERSKI

Human Action Recognition in Videos with Local Representation

Thesis Advisor: Francois BREMOND

prepared at INRIA Sophia Antipolis, STARS Team

defended on November 9, 2017

Jury :

President :	Federic PRECIOSO	-	University of Nice
Reviewer :	Matthieu CORD	-	UPMC - Sorbonne Universities
Reviewer :	Leonid SIGAL	-	University of British Columbia, Disney Research
Examinator :	Jean-Marc ODOBEZ	-	IDIAP
Advisor :	Francois BREMOND	-	INRIA (STARS)
Invited :	Gianpiero FRANCESCA	-	Toyota Motor Europe

RECONNAISSANCE D'ACTIONS HUMAINES DANS DES VIDÉOS UTILISANT UNE REPRÉSENTATION LOCALE

Michal Koperski
Directeur de thèse: Francois Bremond
STARS, Inria Sophia Antipolis, France

RÉSUMÉ

Cette thèse étudie le problème de la reconnaissance d'actions humaines dans des vidéos. La reconnaissance d'action peut être définie comme étant la capacité à décider si une action est présente due une vidéo. Ce problème est difficile en raison de la complexité des actions humaines, dans la grande variété de leur apparence et de leur mouvement.

Les avancèes récentes dans les méthodes manuelles ou par apprentissage profond ont considérablement amélioré la précision de la reconnaissance d'action. Mais de nombreuses questions restent ouvertes, ce qui rend le problème de la reconnaissance d'actions loin d'être résolu.

Les méthodes actuelles basées sur les caractéristiques locales, donnent des résultats satisfaisants. Mais les actions humaines sont complexes, ce qui nous conduit à la question suivante: comment modéliser les relations entre les caractéristiques locales dans leur contexte spatio-temporel? Dans cette thèse nous proposons 2 méthodes pour y répondre. La première modélise les relations spatio-temporelles entre les caractéristiques images utilisant la Covariance Brownienne, et la seconde modélise la disposition spatiale des caractéristiques locales à l'intérieur de la boite englobante de chaque personne. Les méthodes que nous proposons sont générales et peuvent améliorer aussi bien les méthodes manuelles que celles avec apprentissage.

Une autre question ouverte est: l'information 3D peut-elle améliorer la reconnaissance d'actions? Plusieurs méthodes utilisent les informations 3D pour détecter les articulations du corps. Nous proposons de les améliorer avec un nouveau descripteur, utilisant la trajectoire 3D calculée à partir des informations RGB-D.

Finalement, nous affirmons que la capacité de traiter une vidéo en temps-réel sera un facteur clé pour les futures applications de reconnaissance d'actions. Toutes les méthodes proposées dans cette thèse sont prêtes à fonctionner en temps-réel. Nous avons prouvé notre affirmation empiriquement en créant un système temps-réel de détection d'actions. Ce système à été adapté avec succès par la compagnie Toyota pour leurs systèmes robotiques.

Pour l'évaluation, nous nous concentrons sur les actions quotidiennes à la maison telles que: manger, boire ou cuisiner. La reconnaissance de telles actions est importante pour le suivi des patients à l'hopital et pour les systèmes d'aide robotisée à domicile.

Dans ce but, nous avons créé une grande base de données, qui contient 160 heures d'enregistrement de 20 personnes âgées. Les vidéos ont été enregistrées dans 3 chambres avec 7 capteurs RGB-D. Nous avons annoté ces vidéos avec 28 classes d'actions. Les actions dans la base de données sont effectuées d'une manière naturelle et non supervisée, ce qui introduit des défis manquants dans les bases de données publiques.

Nous évaluons aussi nos méthodes en utilisant les bases de données publiques: CAD60, CAD120 et MSRDailyActivity3D. Les expérimentations montrent que nos méthodes améliorent les résultats de l'état de l'art.

HUMAN ACTION RECOGNITION IN VIDEOS WITH LOCAL REPRESENTATION

by
Michal Koperski
Supervisor: Francois Bremond
STARS, Inria Sophia Antipolis, France

ABSTRACT

This thesis targets recognition of human actions in videos. Action recognition can be defined as the ability to determine whether a given action occurs in the video. This problem is complicated due to the high complexity of human actions such as appearance variation, motion pattern variation, occlusions, etc.

Recent advancements in either hand-crafted or deep-learning methods significantly improved action recognition accuracy. But there are many open questions, which keep action recognition task far from being solved.

Current state-of-the-art methods achieved satisfactory results mostly base on features, which focus on a local spatio-temporal neighborhood. But human actions are complex, thus the following question that should be answered is how to model a relationship between local features, especially in spatio-temporal context. In this thesis, we propose 2 methods which try to answer that challenging problem. In the first method, we propose to measure a pairwise relationship between features with Brownian Covariance. In the second method, we propose to model spatial-layout of features *w.r.t.* person bounding box, achieving better or similar results as skeleton based methods. Our methods are generic and can improve both hand-crafted and deep-learning based methods.

Another open question is whether 3D information can improve action recognition. Currently, most of the state-of-the-art methods work on RGB data, which is missing 3D information. In addition, many methods use 3D information only to obtain body joints, which is still challenging to obtain. In this thesis, we show that 3D information can be used not only for joints detection. We propose a novel descriptor which introduces 3D trajectories computed on RGB-D information.

Finally, we claim that ability to process a video in real-time will be a key factor in future action recognition applications. All methods proposed in this thesis are ready to work in real-time. We proved our claim empirically by building a real-time action detection system, which was successfully adapted by Toyota company in their robotic systems.

In the evaluation part, we focus particularly on daily living actions – performed by people in their daily self-care routine. In the scope of our interest are actions like eating, drinking, cooking. Recognition of such actions is particularly important for patient monitoring systems in hospitals and nursing homes. Daily living action recognition is also a key component of assistive robots.

To evaluate the methods proposed in this thesis we created a large-scale dataset, which consists of 160 hours of video footage of 20 senior people. The videos were recorded in 3 different rooms by 7 RGB-D sensors. We have annotated the videos with 28 action classes. The actions in the dataset are performed in un-acted and unsupervised way, thus the dataset introduces real-world challenges, absent in many public datasets.

Finally, we have also evaluated our methods on publicly available datasets: CAD60, CAD120 and MSRDailyActivity3D. Our experiments show that the methods proposed in this thesis improve state-of-the-art results.

Dedicated to
my wife Natalia, who took the brave decision and moved with me to France,
my parents Alicja and Waldemar for all love and support,
my brother Bartosz for all help,
my uncle Krzysztof who discovered computing science for me,
and my grandfather Jozef who could not defend his PhD, due to difficult historical circumstances he had to live in.

ACKNOWLEDGMENTS

First, I would like to express my gratitude to my thesis supervisor, François Brémond. I would like to thank François, for being a great mentor, a guide and a friend. I would like to thank him that he took a risk and offered me opportunity of being his PhD student in such an interesting topic.

I would like to thank Toyota Motor Company for supporting my PhD thesis, I would like to express my gratitude to Anja Höthker and Gianpiero Francesca for these years of collaboration and support.

Thanks to my thesis reviewers, Matthieu Cord, Leonid Sigal and my examinator Jean-Marc Odobez, that they kindly agreed to serve as a members of my thesis committee and review the manuscript.

Special thanks to Sławomir Bąk, who was a precious guide in a scientific world. Collaboration with him helped me to became not only better scientist but also a better person. I also wanted to say thank you to Piotr Biliński, who convinced me start a PhD and helped me a lot in the early stages.

Thanks to Carlos Crispim-Junior and Farhood Negin for technical and scientific support. I would like to thank all my colleagues from STARS team for our collaboration, I had great time with you and I will never forget great moment that we shared together.

I would like to thank Marc Vesin and Francis Montagnac the administrator of NEF computational cluster. Thank you for all your technical support and for managing the NEF cluster, without which vast amount of experiments done in this thesis would not be possible.

I would like to thank all friends I met at French Riviera, who made me feel like at home. Special thanks to Magda and Bartosz, Magdalena and Tomek, who converted my stay in France to an incredible adventure.

I thank also my friends in Poland: Pysio, Statek, Tomuś and Wojtek. Thank you so much that you always had time to meet me during my rare and short visits.

My final words go to my family. I would like to thank my wife Natalia, who moved with me to France. I know that it was difficult and brave decision. I want to thank her for all the support through both the highs and lows of the time of my PhD. I would like to thank my parents, my brother and the whole family.

Contents

Chapter 1

Introduction

Contents

There are many high level tasks, that humans perform automatically and subconsciously. But in fact those tasks involve complicated processing, which takes place in our brain. The statement above is especially true for any computer vision related task. Tasks like object detection, people identification, people re-identification, action recognition are extremely difficult tasks for computers. On the other hand even couple years old children can handle those task easily.

Computer vision tasks are difficult due to huge variations of appearance, motion patterns, view point angles, lighting conditions, they are easy to solve by sophisticated and specifically designed tool – human brain. Computers were primarily designed to perform fast well defined computational tasks, but not complicated reasoning. Thus the main goal of computer vision is to improve computer abilities in interpretation of image and video information. Such abilities will play key role in future intelligent machines such as robots or vehicles. Action recognition can be defined as the ability to determine whether a given action occurs in the video stream. Action recognition is one of the most important components of intelligent systems, as the most of available information is visual. Ability

to interpret such information will ultimately bring computers one step closer to human skills.

This chapter introduces the problem of action recognition. In section 1.2 we provide formal definition of the problem and present key applications of action recognition. In section 1.3 we discuss major challenges and difficulties. We conclude the chapter with the list of contributions (section 1.4) and the thesis structure (section 1.5).

1.1 Problem statement

In this section we define human action, action recognition and action detection problem. To define the above terms we use terminology defined by Moeslund *et al.* [70].

Action
It is a complex body movement, which consists of several action primitives (gestures) ordered in time.

Action Recognition
Let's assume that we have a set of videos $\mathbb{V}$ and a set of corresponding action labels $\mathbb{L}$. We assume that each video $V \in \mathbb{V}$ contains only one action l_V. Thus the goal of action recognition problem is to predict label l_V based on video representation V.

Action Detection
Action recognition problem can be extended to action detection problem. In this problem we assume that video V can contain more than one action, thus each video $V \in \mathbb{V}$ has corresponding set of labels L_V. The goal is to predict all labels for given video V, as well as time when given action starts and ends.

In this thesis we focus on action recognition problem, as we claim that to solve action detection problem, first we need action recognition which achieves high accuracy. In chapter 10 we show that our action recognition framework can be extended to action detection case.

1.2 Motivation

Video Retrieval
Currently we can observe rapid development of internet broadcasting services such as

(a) Frame from "Minority Report"

(b) Kinect RGB-D sensor

(c) Leap Motion sensor

Figure 1.1: (a) – frame from "Minority Report" movie presenting computer interface controlled by gestures, (b) and (c) sensors which are used in modern human computer interfaces.

YouTube or Vimeo as well as social media services such as Facebook or Twitter. Phones and other mobile devices are equipped with high quality cameras. Many mobile phone users have access to high speed mobile internet.

All above facts result in massive amount of videos being uploaded every day. For instance every minute 300 hours of videos are uploaded to YouTube[1].

With such enormous amount of data – systems which can efficiently analyze videos and provide accurate search results or suggestions are becoming more and more important. Action recognition is a key part of such systems.

Video surveillance

Nowadays we can observe surveillance cameras almost at every corner of the city. They are installed in: train stations, metro stations, airports, banks, shopping malls, city streets. The primarily goal of surveillance cameras is to increase the security level and protect us from acts of violence, terrorism, vandalism, stealing. For instance 41 % of public premises in London have CCTV equipment. They estimate there is around 420,000 cameras in the city of London.

With such huge amount of cameras, it is impossible for human operators to observe all available video streams. As a result video surveillance footage is often used to find the suspects of a crime, but rarely used to prevent potentially dangerous situations.

Thanks to action recognition systems, video surveillance cameras could be analyzed 24/7 and alert could be triggered in case of any dangerous situation. Such system will be able to fully exploit potential of huge video surveillance camera networks and greatly improve safety thanks to prevention of potentially dangerous situations.

[1] http://www.statisticbrain.com/youtube-statistics/

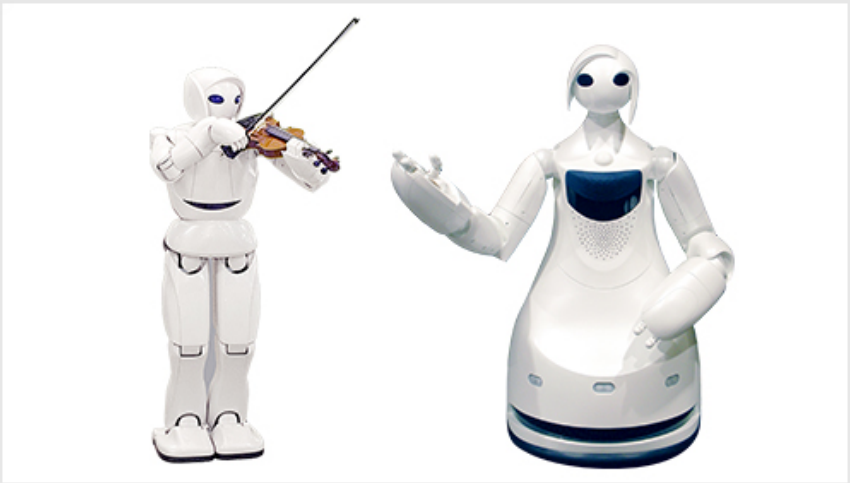


Figure 1.2: Toyota Partner Robot system.

Human Computer Interaction

From the early stages when modern computers were born, they were equipped with keyboards. At the beginning keyboard served as the only interface to communicate with computer. Further developments in computer capabilities, especially in computer graphics led to design of computer mouse. Similar trend emerged in electronic games industry, where keyboards were replaced by joysticks and later by advanced game-pads, steering wheels *etc.*

The evolution of Human Computer Interaction started with keyboards and led us to many different devices which make communication easier, faster and more comfortable. With recent advancements in computer vision and camera sensors, we have reached a point where scenes from sci-fi movies like the one from "Minority Report" are not fiction, but reality. Modern capabilities of sensors like Kinect, Leap Motion joint together with gesture or action recognition algorithms allow us to build interfaces where a user can operate computer without any need of holding any device (see fig. 1.1). The computer is able to recognize actions or gestures done by a user and trigger appropriate actions. Such solutions have been already introduced to TV sets. In entertainment industry Xbox game console equipped with Kinect sensor is able to interpret whole body motion, leading to new levels of game experience – especially in sport games.

All recent advancements in Human Computer Interaction are possible thanks to development of gesture and action recognition algorithms.

Robotics

Robotics is an interdisciplinary branch of engineering and science. Robotics deals with

the design, construction, operation, and use of robots, as well as computer systems for their control, sensory feedback, and information processing. These technologies are used to develop machines that can substitute humans. Robots can be used in any situation and for any purpose, but today many are used in dangerous environments (including bomb detection and de-activation), manufacturing processes, or where humans cannot survive.

Robots ability of human action recognition plays a key role in many robotics applications. One of them are autonomous vehicles which can be considered as specific type of robot. The ability of observation and anticipation of road situation defines a good driver. Thus efficient interpretation of intention of other traffic participants will play a key role in future of autonomous vehicles. The anticipation capabilities may concern vehicles, but also pedestrians. For instance, autonomous vehicles are required to asses and anticipate pedestrian intention to cross a road. This would let autonomous car avoid potentially dangerous situations.

Partner robot systems is another area of robotics, which can widely benefit from action recognition capabilities. Partner robot systems can be defined as robotics systems which are able to assist humans with different activities. This include support of a group of doctors, nurses and other caregivers. Other group which can benefit from assistance are sick patients, elderly and disabled people in hospital or at home. Partner robots can also assist people in ordinary housework activities. All partner robots functions mentioned above require them to understand human actions and behave accordingly.

Health care

According to survey[2] there were 810 millions people of age 60+ in 2012 on the Earth. The survey predicts that there will be 1 billion people of age 60+ by 2022 and 1.375 billion by 2030.

Japan is a sharp example of this global trend. The number of Japanese people aged 65+ nearly quadrupled in last 40 years reaching 33 millions in 2014[3], which stands for 26% of Japanese population. According to projection[4] people of age 65+ will account for 40% of Japanese society by 2060. This demographic trend translates to dramatic need of increase of workforce in health care. According to survey[5] 2.9% of Japanese people between 75-79 are permanently hospitalized, while 13.4% visited a doctor at least once in a year.

Due to above facts action recognition is becoming more and more important and is used in medicine in patient monitoring. Action recognition systems can monitor health state of the patient and early detect potential physical or mental problems. For instance

[2] http://www.helpage.org/
[3] http://www.stat.go.jp/english/data/nenkan/1431-02.htm
[4] http://www.ipss.go.jp/site-ad/index_english/esuikei/ppfj2012.pdf
[5] http://www.ilcjapan.org/agingE/doc/POJ_2013_4.pdf

monitoring patient eating habits, housekeeping routine or sleep allow doctors to monitor state of a patient and react before serious health conditions arise. Thanks to such systems seniors can stay longer at home without a need of being hospitalized, which greatly improve their comfort and health.

Partner robot systems mentioned in this section are also aiming to improve health care level of aging societies. Both partner robot systems and patient monitoring are in great interest of this thesis. Thus, although proposed action recognition methods are generic and can be applied to any action recognition problem, we decided to select in evaluation daily-living datasets. Such datasets contain action set, which allows to monitor patient state. In addition we recorded Smarthomes dataset which contains action set useful from patient monitoring perspective. But more importantly, actions are performed by 20 senior people. This way we introduce many challenges, which do not exist in state-of-the-art datasets.

1.3 Research challenges

Action recognition is a very challenging research problem and many unanswered questions keep action recognition from being solved. In this section we list and discuss some of most important research challenges in action recognition.

Intra-class variation

Each action can be done in many different ways. For long and more complex actions, there are many different unimaginable ways in which action can be done. The variation comes from people habits: for instance right-handed vs left-handed. For long actions there might be a different temporal order of taking different steps. For instance for making coffee action one may first put coffee to machine while other might start with pouring the water. For same action class people may use different objects: drinking from cup vs. drinking from bottle.

As we can see from the examples above, there is not a simple way to describe the actions, this makes it hard in terms of features design and as a classification problem.

Detection

Some action recognition methods take advantage of information about object (object detection) or pose (skeleton detection). The recognition accuracy of such methods can be significantly affected by failing detection. In such case, action recognition algorithms have to deal with noise (false positives) and missing detection.

Discriminative features

Human brain can perform well with problems which require ability of association and discrimination. In learning process human brain does not treat whole information equally focusing on more important parts for given task [11]. This leads us to open question: how to build a computer system, which can recognize (associate) human actions, using discriminative cues (features).

Spatio-temporal context association

Due to mentioned complexity and variation in human actions, one of the most important problems to address is how to model complicated relationships between location and time that different things happened while action was performed. The mentioned relationships could be ignored for simple actions like hands clapping or hand waving. But in order to recognize more complicated actions it will be important to model "what", "where", "when" relations and their order.

1.4 Contributions

As we mentioned in the beginning of this chapter, although methods proposed in this thesis are generic and can be used with any action recognition task. We are particularly interested in daily-living action recognition. Such problem introduces specific challenges: high intra-class variance, high amount of actions which are similar, actions are performed in the same environment *i.e.* apartment. We can notice that some datasets such as UCF-101 [103] or HMDB51 [49]which are focused on action recognition from videos uploaded to the internet are different in a way that their inter-class variance is high (*i.e.* "ride a bike" vs. "sword exercise"). But they introduce other challenges such as camera motion and noisy background. Decisions and scientific contributions proposed in this thesis were driven by the intention to improve daily-living action recognition.

One of the most important characteristic of the proposed methods is that they do not require skeleton detection. The contributions of this thesis can be divided into 4 main groups described below:

1. **Improved Fisher Vector representation** – we propose 3 method which incorporate spatial information into Fisher Vector. We also provide computational efficient Fisher Vector representation based on 1D integral image, which is suitable for real-time applications (*eg.* robotic systems).

2. **New appearance and motion descriptors** – we propose 2 new appearance descriptors which improves recognition of actions with similar motion patterns. In addition

we propose motion descriptor which take advantage of 3D information available from RGB-D sensor.

3. **Recognition of actions with low amount of motion** – we address the problem of local features methods which often fail when action is static and low number of points of interest is detected. We use people detection to model rough pose information, thus our method do not require skeleton detection.

4. **Extensive evaluation and new dataset** – we provide extensive evaluation of two baseline methods based on local features on daily-living datasets. In addition we built new large scale daily-living action recognition dataset.

Below we list and describe all groups of contributions with more details.

1.4.1 Improved Fisher Vector representation

Spatio-temporal Fisher Vector representation

In recent years local features methods (see chapter 2) were the most successful in action recognition. Those methods use simple feature detectors and model the features in local neighborhood of detected points of interest. The advantage of such methods is that they do not require complicated detection algorithm (*i.e.* skeleton detection) as a pre-processing step. One of the drawbacks of local features method is that they usually ignore spatio-temporal layout of detected features. Work of [115, 26, 76, 130] show that skeleton information can be especially useful in daily-living action recognition. But due to our experience with Smarthomes (section 2.3.4 and appendix A), we claim that skeleton detection in real-world scenarios is still challenging. Thus we propose to model spatio-temporal layout of features *w.r.t.* person bounding box, because people detection is a much less error prone task. We propose 3 different strategies of layout modeling. The method was published in [44], the detailed description is available in chapter 7.

Real-time action detection framework

In chapter 10 we provide details of real-time action detection framework. We use 1D integral image representation, which let us create efficient Fisher Vector encoding.

1.4.2 Appearance and motion descriptors

3D Trajectories

We propose to model action as 3D trajectory shape of locally detected points of interest. Such 3D representation is able to catch more details of action motion pattern *i.e.* distinguish actions whose projected 2D trajectories would look the same. To obtain 3D

information we use RGB-D sensor. We provide a method which deals with missing depth information and noise, which is often a problem in depth-map obtained from RGB-D sensors in real world scenarios. This work was published in [43] and is described in details in chapter 4.

Histogram normalization

Many successful descriptors proposed in recent years (HOG, HOF, MBH) in action recognition are based on histogram representation. Histograms are usually normalized with $L1$ or $L2$ norms. We show that incautious application of L-like norms to the histogram based descriptors might produce a hallucination effect. Specific pattern, which was not present in unnormalized histogram, may appear in normalized representation. We propose to add an extra bin to histogram representation to avoid hallucination effect. We also propose an algorithm to find the value of extra bin. Our method is generic and can be applied to any histogram based descriptors, but in our analysis we focus on HOG descriptor. Our decision is motivated by the intention to improve appearance cues. The method was published in [79] and is described in details in chapter 5.

Brownian Covariance

Many descriptors proposed in computer vision are gathering simple statistics (*i.e.* histograms) of hand-crafted features (HOG, HOF, MBH). We propose to model pixel-level features with higher level statistics. In our method we model relationship between pixel-level features using Brownian Covariance. Brownian Covariance is a new statistical method which models Brownian motion of particles. Using Brownian Covariance we are able to measure all type of relationships (non-linear, non-monotone). The method was published in [10] and is described in details in chapter 6.

1.4.3 Recognition of action with low amount of motion

Some actions do not involve high amount of motion while being performed. For instance "using computer" action involves only motion coming from fingers and hands. Such actions are usually not recognized by local features methods, because they cannot detect enough features to describe the action, due to lack of motion. We propose simple descriptor which capture rough pose information from person bounding box. The method was published in [44] and is described in chapter 8.

1.4.4 Evaluation and new Smarthomes dataset

Evaluation of local features methods on daily-living datasets
We provide deep evaluation of local features methods on daily-living datasets. We evaluate Dense Trajectories [113] and Trajectory Pooled Deep Convolutional Descriptor (TDD) [117] on 4 datasets: CAD-60 [107], CAD-120 [107], MSRDailyActivity3D [116], Smarthomes.

Smarthomes – large scale daily-living dataset
In this thesis we have built a new dataset, which contains 28 action classes. The actions are performed by 20 senior people during the day. The actions were recorded in different locations (dining room, kitchen, living room). We used 7 RBG-D sensors to record the dataset, thus we introduce different viewpoint angles. Action were performed in unsupervised and completely natural way. Therefore we introduce many challenges which were missing in many public datasets. The dataset can be used to evaluate action recognition methods. In this setup it contains over 17,000 video clips. Each clip contains one annotated action. Smarthomes dataset can be also used to evaluate action detection methods. In this setup we provide over 700 hours of video stream. The details of the dataset are available in section 2.3.4 and appendix A.

1.5 Thesis structure

Chapter 1 introduces the action recognition problem. We describe problem motivation and provide key possible applications. Then we discuss main challenges in action recognition problem and summarize our contributions. The remaining chapters are organized as follows:

- **Chapter 2** "**Related Work**" introduces state-of-the-art methods. We divide state-of-the-art methods, based on how they address challenges we defined. We review different RGB, RGB-D, skeleton based methods. We also review hand-crafted based and CNN based methods. We also provide comparison of public datasets and we describe datasets selected for evaluation in this thesis.

- **Chapter 3** "**Action Recognition Framework**" defines the action recognition framework based on Fisher Vector representation and SVM classifier. We provide detailed description of two baseline methods: Dense Trajectories (DT) [113] and Trajectory-Pooled Deep-Convolutional Descriptors (TDD) [117]. We evaluate the baseline methods on 4 datasets: CAD-60 [107], CAD-120 [107], MSRDailyActivity3D [116], Smarthomes.

- **Chapter 4** "**3D Trajectories**" proposes a new method for action recognition based on 3D Trajectory Shape Descriptor (3DTSD). In this chapter, we describe how to obtain motion 3D trajectory points, we describe how to handle noise and missing values. Finally we show that proposed representation improves recognition accuracy.

- **Chapter 5** "**Removing Hallucinations by histogram normalization**" describes possible problems with L2-normalization. We show that an incautious application of L-like norms to the HOG descriptor might produce a hallucination effect, i.e. specific properties of high texture can appear while describing a texture-less image. To address this problem, we propose new normalization schema, where we add extra bin (EB) to the histogram representation. We show that value of EB has an impact on descriptor performance, thus we propose algorithm, which is able to find EB value automatically.

- **Chapter 6** "**Brownian Covariance**" proposes new appearance descriptor for action recognition which measures relationship between pixel-level features. To measure the relationship we employ Brownian Covariance, which is able to measure non-linear, non-monotonic relationships. In the experiments we show that Brownian descriptor is carrying complementary information to HOG descriptor.

- **Chapter 7** "**Modeling spatial layout with Fisher Vector and people detection**" proposes 3 methods which model spatial layout of local features. The aim of the proposed methods is to improve Fisher Vector representation which by default ignores spatio-temporal layout. We model a layout of features *w.r.t.* person bounding box, while skeleton detection is still not an easy task. We provide extensive evaluation and show that proposed methods are competitive to skeleton based methods.

- **Chapter 8** "**Pose and skeleton based models**" describes new GHOG descriptor, which models static pose information of a detected person. The proposed descriptor does not require skeleton detection. The pose information is encoded using appearance information inside person bounding box. Local features methods often suffer from low number of detected features with GHOG we address this problem. In experiments section we provide results of fusion with all previously proposed methods.

- **Chapter 9** "**Comparison with state-of-the-art**" presents a detailed performance analysis of proposed descriptors for action recognition and shows a performance comparison of these approaches with state of the art techniques. We evaluate our approaches on publicly available datasets. We discuss challenges that each dataset introduces. We also investigate advantages and limitations of proposed methods.

- **Chapter 10** "**Online Action Detection**" describes a framework for online action detection. In this chapter we show that our methods can be used in online action detection system. We show that effective use of 1D integral image allows to compute Fisher Vector representation in computationally effective way. We describe components of our action detection system which was successfully applied to Toyota Partner Robot system.

- **Chapter 11** "Conclusion" outlines our approaches. We provide discussion about advantages and limitations of our work. In this chapter we also suggest future directions, presenting short-term and long-term perspectives.

Chapter 2

Related Work

Contents

2.1 Introduction

In this chapter we review the methods for Action Recognition published in recent years. The exhaustive reviews are available in [88, 92, 54, 128]. In our review we would like to point out 4 challenges in Action Recognition and show how they are approached in state-of-the-art.

1. **How to select data input type?** This challenge is one of the most fundamentals, because input data type strongly determines the recognition capabilities. RGB input for instance provides a lot of texture information and it seems to be close to the input information that humans process. On the other hand RGB encodes also a lot of information which is not important in Action Recognition. For instance RGB appearance depends strongly on lighting conditions. While in action recognition preferable representation should be invariant to illumination. In such case methods

which work on RGB data have to find relevant regularities and distinguish them form other irrelevant regularities such as illumination changes. This is a non- trivial task. Depth-map input on the other hand is free of some problems which exist in RGB. For instance depth-map is invariant to illumination changes, it is easier to distinguish forward scene from background, and it provides 3D information about the scene. But depth-map has also some limitations: it does not provide texture information and usually introduces a lot of measurement noise (this is especially true for low-cost sensors *eg.* Microsoft Kinect, Asus Xtion). Segmented human skeleton is the last input type that we consider. This input type is usually obtained by applying different methods [98, 124, 83, 14] either on RGB or depth-map. Thus skeleton might be consider as higher level feature, but studies of Johansson [42] show that humans can recognize a lot of actions based only on skeleton information. That is why we decided to consider skeleton as separate input type for Action Recognition methods.

2. **How to model features?** The problem here is the granularity level of the features. Some authors [12, 33, 100] propose to model the input as a whole (holistic methods). Other authors [132, 26, 76, 35] propose to use detailed segmentation (skeleton methods). Finally some authors [51, 52, 53, 67] propose to search for salient Points Of Interest (local features methods).

3. **How to model motion?** Motion seems to be the most important information in Action Recognition. Many authors proposed different ways to model motion in Action Recognition. Some authors proposed to design low level features (*eg.* optical flow) or higher level features (*eg.* trajectories [113, 114, 117]) and use then with classifiers which ignore temporal nature of data. Other authors [63, 127] proposed to use classifiers which can model sequential data (HMMs, CRFs, RNNs) and in such way they model motion in videos.

4. **How to design features?** There are two common ways to approach this challenge. We can either use expert knowledge which would lead us to design of handcrafted features. On the other hand we can apply method which would automatically find features based on input data. Recent advancements in Deep Learning show that for image classification the second approach lead to superior performance over handcrafted features. This conclusion is not obvious for Action Recognition problem. Handcrafted features achieve competitive results comparing with automatically learnt features with CNN (Convolutional Neural Net).

The state-of-the-art methods intersect each other in many different ways based on challenges defined above. In the following chapters we partition methods based on challenge 1 and 2. But in 2.1 we provide overview of selected state-of-the-art methods *w.r.t.* all

challenges described above.

2.2 Input data

2.2.1 RGB

Holistic Representation In holistic representation the image Region Of Interest (ROI) is treated as a whole, which means that all pixels in the ROI are potentially used to compute the descriptor. The ROI is usually bounding box of detected person. So in fact the holistic methods involve two steps: a person detection using either background subtraction and/or tracking and descriptor computation. Common representations are silhouettes, edges and optical flow based descriptors. These methods are generally sensitive to noise, variation in viewpoint and occlusion [84].

The foreground of person in the image forms a human silhouette, which provides shape information about the human pose. The evolution of such silhouette over the time is used in action recognition. In the work of [21] the differences of binary silhouettes are accumulated over spatial domain; constructing Motion Energy Image (MEI), and over temporal domain; constructing Motion History Image (MHI). Such representation forms action template. The statistical model of moments is further used to match unknown action instance to closes template.

In [33] the actions are modeled as 3D shapes obtained by stacking the 2D silhouettes in spatio-temporal volume. Zhu *et al.* [141] employed an extended Radon transform on the binary silhouette. Such representation is invariant to geometrical transformations such as scaling and translation. In [136] the extremities of a human body like head, hands and feet are used to model the action. These extremities are detected from a body contour. Finally Qian *et al.* [86] used contours of the MEI and obtained descriptor which

Method	**(1) Input type**	**(2) Features Model**	**(3) Motion Model**	**(4) Features Design**
Harris3D [51]	RGB	Local	Stacked RGB volume	Handcrafted
3D-MHI [74]	Depth-map	Holistic	Stacked Depth-map volume	Handcrafted
IDT [114]	RGB	Local	Optical flow	Handcrafted
Two-streams Net [100]	RGB	Holistic	Optical flow	Learnt
TDD [117]	RGB	Local	Optical flow	Learnt
Lv *et al.* [63]	Skeleton	Joints	Sequential Classifier	Handcrafted
Wu *et al.* [127]	RGB	Holistic	Sequential Classifier	Learnt

Table 2.1: Comparison of selected state-of-the-art methods, *w.r.t.* to Action Recognition challenges.

is invariant to scale changes and translations.

The mentioned Holistic Methods relay heavily on people detection (to detect ROI) and background subtraction (to extract features). Recent advancements in object detection in RGB images makes people detection a minor issue. But computing stable and accurate silhouettes based on RGB images is still a challenge, especially in real-world scenario videos.

Local Features Representation The key idea behind Local Representation is to capture characteristic features from a local spatio-temporal volume of a video. Same idea was successfully applied to object detection, scene recognition etc. The main advantage of Local Representation based methods is that, they do not require neither human segmentation nor people detection, which are to some extend challenging tasks on RGB videos. In 7 we propose a method which improves performance of Local Representation by incorporating information from person bounding box.

So in fact the holistic methods involve two steps: Point Of Interest (POI) detection and descriptor computation. The goal of feature detector is to select a video region which maximizes the saliency. Laptev *et al.* [51] proposed a feature detector which extends the Harris corner detector [36] to spatio-temporal case – Harris3D. Dollar *et al.* [23] claimed that cases when Harris3D triggers are rare. They claimed also that salient motion is present in other video regions. They proposed detector which uses spatial Gaussian kernels and temporal Gabor filters. Oikonomopoulos *et al.* [77] proposed detector which computes entropy in a cylindrical neighborhood around a given space-time position for the temporal derivative of a video. Willems *et al.* [125] proposed Hessian3D detector as spatio-temporal extension of the Hessian saliency measure. The detector measures saliency using the determinant of the 3D Hessian matrix. An integral video structure allows a speed up of computations by approximating derivatives with box-filter operations. A non-maximum suppression algorithm selects joint extrema over space, time and different scales. Most feature detectors determine the saliency of a point with respect to its local neighborhood. Wong and Cipolla [126] proposed method which finds salient features by considering global information. Video sequences are represented as a dynamic texture with a latent representation and a dynamic generation model. The dynamic model is approximated as a linear transformation. A sub-space representation is computed via non-negative matrix factorization.

When salient POIs are detected, the common strategy for Local Representation based methods, is to compute descriptor for given local volume around POI. Many authors proposed different descriptors: Laptev *et al.* [52] used HOG [20] and HOF [113] descriptor. Dollar *et al.* [23] used different local descriptors based on brightness, gradient, and optical flow. They investigate different descriptor fusion methods: simple concatenation of

pixel values, a grid of local histograms, and a single global histogram. They report that, concatenated gradient information leads to best performance. An extension of the SIFT descriptor [60] was proposed by Scovanner *et al.* [93]. For a set of randomly sampled positions, spatio-temporal gradients are computed in the local neighborhood of each position. Each pixel in the neighborhood is weighted by a Gaussian centered on the given position. For orientation quantization, the authors represent gradients in spherical coordinates that are divided into an 8×4 histogram. Willems *et al.* [125] proposed the Extended SURF (ESURF) descriptor, which extends the image SURF descriptor [5] to videos. The authors divide 3D patches into a grid of local $M \times M \times M$ histograms. Each cell is represented by a vector of weighted sums of uniformly sampled responses of Haar-wavelets along the three axes.

Feature trajectories is an idea that arisen on top of Local Representation. Many authors claimed [67, 68, 106, 113, 114] that **2D spatial domain and temporal domain in videos have have a very different characteristics**. Because of that POI should not be detected in joint 3D spatio-temporal space. Based on conclusion above many authors [67, 68, 106, 113, 114] proposed methods where they track detected spatial POI across time. The trajectory shape and descriptors computed based on volume around the trajectory points are then used as video representation. Messing *et al.* [68] extracted feature trajectories by tracking Harris3D interest points [51] with the KLT tracker [62]. Trajectories are represented as sequences of log-polar quantized velocities. Matikainen *et al.* [67] used a standard KLT tracker. Trajectories in a video are clustered, and an affine transformation matrix is computed for each cluster center. The elements of the matrix are used to represent the trajectories. Sun *et al.* [106] extracted trajectories by matching SIFT descriptors between two consecutive frames. They imposed a unique-match constraint among the descriptors and discarded matches that are too far apart. Wang *et al.* [113, 114] proposed Dense Trajectories method where they densely sample POI from the grid. Then the POIs were tracked using optical flow. Action was represented by trajectory shape descriptor, HOG [20], HOF [113], MBH [20, 113] computed in a cropped volume around each trajectory points. The methods which leverage features trajectory information showed impressive results in action recognition. Many further extension of Dense Trajectories were proposed [9, 117, 8, 7, 6, 137].

Deep Learning The Deep Learning methods in action recognition are closely related to efficient use of CNNs in image classification. In general one of the main advantage of CNNs is that they learn in end-to-end process both: features (filters) and classification boundaries. Thanks to that we can obtained specialized features which describe the data better and make classification task easier. Of course such approach is prone to overfitting so the training process of CNN requires careful regularization and typically significant amount of

labeled examples. It is now clear that CNN-based approaches outperform most state-of-the-art handcrafted features for image classification [48, 89]. For the action recognition such conclusion it is not obvious. Videos introduce additional temporal dimension, while CNNs initially were designed to model static images and is not clear how motion dynamics should be encoded in CNN. Thus in action recognition most of authors use CNNs trained on still images to model static appearance. Then they proposed different ways – how to model motion in the video. Ji *et al.* [41] proposed to used stacked video frames as an input and replace 2D convolutions with 3D. They reported improvement over 2D convolution architectures, but their approach did not outperformed hand-crafted based features. Simonyan *et al.* [101] proposed to model motion with CNN trained on optical flow input, together with second CNN trained on still images. This approach was later extended by Feichtenhofer *et al.* [28], they replaced late fusion approach, by fusing two nets at lower layers and perform joint fine tuning. Wu *et al.* proposed to use (Recurrent Neural Net) RNN, to model motion in the video. RNN is a special architecture which can handle sequential input. For each sequence item a copy of Neural Net is created. The RNN contains connection between neurons in hidden layers between each replica and each neural net replica shares the same model weights with the others. Wu *et al.* claim that actions in the video can be decomposed to local motion which can be handled effectively by optical based CNN. And the global motion which can be effectively modeled by RNNs. They compute CNN feature for each video frame and then they feed them into RNN. Wang *et al.* [119] introduced sparsely sampled temporal fragments to model long term global motion. Ma *et al.* used LSTM for early activity recognition [65]. Xu *et al.* proposed to use transfer learning and feature embedding [131].

In general proposed Deep Learning methods perform on the same level or slightly better than models with hand-crafted features (see 2.2). Another issue is that recent state-of-the-art datasets [49, 103], introduce quite high inter-class variation. In such case static appearance information is favored over motion information. For instance on UCF-101 dataset CNN trained on static images achieves 60% of classification accuracy based on single randomly sampled frame of the video. Because of the reasons above many authors [100, 117, 28] proposed to fuse hand-crafted features together with Deep Learning models, achieving superior results.

2.2.2 Depth-map

Spatial 3D information was not in the limelight of the research, until the introduction of low-cost RGB-D sensors (*eg.* Microsoft Kinect, Asus Xtion). The introduced devices are able to output both RGB and depth-map stream in real-time. The depth-map provides information about the distance of each pixel to the sensor. The reported distances are

Features	Method/Dataset	UCF-101	HMDB51
Hand Crafted	Improved Dense Trajectories (IDT) [114]	88.0	57.2
	Modeling Video Evolution for Action Recognition [31]	-	61.5
	Multiskip features [50]	**89.1**	**66.8**
CNN	Two-streams Net [100]	91.5	65.9
	Hybrid Deep learning net [127]	90.1	-
	TGP Net [118]	91.4	-
	CNN for unconstrained videos [138]	89.2	-
	Two-stream fusion [28]	92.5	65.4
	Temporal Segment Network [117]	**94.2**	**69.4**
Hybrid	Two-stream fusion + IDT [28]	**93.5**	**69.2**
	Trajectory-Pooled Deep-Convolutional Descriptors [117]	91.5	65.9

Table 2.2: Comparison of state-of-the-art methods which use hand-crafted features vs. methods which use CNN. We compare the results on two popular datasets UCF-101 [103] and HMDB51 [49]

usually in millimeters, and sensor working range is from 0.5 m upto 7 m. The sensor's accuracy is a non-linear function of distance, so the further given object is located from sensor, the less accurate is the measurement. The depth-map obtained from sensor is usually noisy and contains many missing values due to occlusions and the fact that many materials absorb beams sent from the sensor. Nevertheless information obtained from RGB-D sensors played a key role in improvement of scene segmentation people detection and people segmentation.

Holistic methods As mentioned above depth information makes scene segmentation task easier. In particular it is easier to obtain reliable background subtraction. Due to this fact many authors extended existing RGB Holistic methods (see 2.2.1) to RGB-D case.

Ni *et al.* [74], extended MHI including the depth information. They proposed 3D-MHI which extends MHI which encodes motion history in depth direction. In [55], a small set of representative 3D points sampled from the depth silhouette is used to characterize the shape of salient postures. The idea here is that the points inside the silhouette carry redundant information and the body shape can be described sufficiently by a small number of extreme points of the contour. The depth map is projected on to the three orthogonal Cartesian planes XY, YZ and XZ and points are sampled at equal distance along the contours of the projection. The temporal dynamics of these sampled points are used to infer the actions. A similar planar projection method is used in [135]. The depth maps are projected on to the three orthogonal Cartesian planes. Then motion energy obtained from the projected maps are stacked together to form Depth Motion Maps (DMM). The DMM representation encodes information about the body shape and motion in three projected planes. The approach in [117], where the 2D silhouettes are stacked to create a 3D space-

time volume, has been extended to depth sequences as well. Viera *et al.* [111] extended the [33]. They divided space and time axes into multiple cells to define a 4D space-time grid for a depth image sequence. A saturation scheme is used to enhance the role of the cells and make them suitable for recognition. The obtained feature vectors, called Space-Time Occupancy Pattern (STOP). Oreifej *et al.* [78] proposed to compute normal vectors to the surface planes. The normal vectors are compute for for each frame. Then to describe video a 4D histogram of surface normal orientations (HON4D) was proposed. The features capture the distribution of the surface normal direction in the 4D space of spatial, depth and time axes.

Depth information makes people detection and background subtraction tasks less challenging, which made holistic methods on RGB-D images more robust. Nevertheless some drawbacks of holistic methods are still true. The mentioned methods are still sensitive to occlusions and missed defections.

Local methods Local methods on depth-map follow same principle as on RGB videos. The key idea is to find the salient Point Of Interest (POI) and then compute the descriptor around given POI. In [87], a descriptor called Histogram of Oriented Principal Components (HOPC) is proposed. It captures the local geometric characteristics around each point within a sequence of 3D point clouds. First Principal Component Analysis (PCA) is performed on a spatio-temporal volume around the point, then the resulting Eigenvectors are projected onto a number of directions corresponding to the vertices of a polyhedron. The descriptor formed by concatenating these projected Eigenvectors is used when performing action recognition. The authors claim that proposed HOPC descriptor is invariant to changes in viewpoints. The Comparative Coding Descriptor (CCD) was proposed in [15]. This descriptor encodes the structural relations of points in space and time. The video is treated as a spatio-temporal volume of depth values and a set of small atomic cuboids extracted from this volume is used to construct a sequence of codes that define the descriptor. The occupancy patterns introduced in previous paragraph were applied to describe cell of spatial grid [116]. The numbers of points that fall into the cells of a a spatial grid are used to form descriptor called Local Occupancy Pattern (LOP). The LOP is used to describe the appearance in a sub region of the depth image and is useful for characterizing the interactions with objects when an action is performed. Wang *et al.* [115] proposed Random Occupancy Pattern (ROP). In this method a depth sequence is considered as a 4D spatio-temporal volume in which the pixel values are binary. The ROP features are defined by the sum of the pixel values in a sub-volume. There are a number of sub-volumes with different sizes and at different locations. The set of possible sub-volumes is prohibitively large, so a random sampling approach is used to efficiently explore the sub- volumes.

In general many Local methods are used together with different input *eg.* with skele-

ton – where skeleton joints are used as a detector, or RGB where POI are detected on RGB video frame. Depth information does not provide a lot of texture information, that is why very often is more efficient to find salient POI on RGB frame.

Deep Learning There are only few Deep Learning based methods applied to depth-map input. There are two reasons behind it. The action recognition datasets which provides depth-map information are even smaller than RGB-only datasets. This makes training deep neural nets from scratch, a very difficult task. In addition depth-map have very different characteristics than RGB frames, which makes it also difficult re-use pre-trained RGB neural nets.

Pichao *et al.* [120] proposed pseudo-coloring scheme, which they applied on depth-maps. This allowed them to use RGB based pre-trained models. In [121] authors proposed Depth Motion Maps, where they compute difference between consecutive video frames and use this as an input to CNN.

2.2.3 Skeleton

Skeleton based methods became an active area of a research after introduction of low-cost RGB-D sensors and introduction of robust skeleton detection method [98], which was able to detect 20 skeleton joints from RGB-D data in real time. Many authors were motivated by study of Johansson [42], in which he demonstrated that humans can recognize actions, only based on glowing landmarks attached to joints. For the long time RGB-D based skeleton detection was the only to provide reliable and real-time results. Recent advancements in Deep Neural Networks resulted in methods [83, 40, 124] which can detect skeleton form RGB data in nearly real-time processing time, when run on GPU (Graphics Processing Unit).

Many skeleton based methods focus on joint position representation. They model relative joints positions, pairwise joints position or angles between joints. Such representations focus on pose information correlated with given action.

In [115] the feature for a joint is determined by taking the difference between the position of a joint and all the other joints. The overall feature is determined by enumerating all the pairwise joints. Yang *et al.* [132] proposed the relative joint positions computed from several video frames are used as features. Apart from the differences between the joints in the current frame, the pairwise differences are computed between the current frame and a preceding frame to capture the motion properties. Ellis *et al.* [26] proposed to compute Euclidean distances between every pair of points in the current frame and previous frame. In [76], the relative azimuth and elevation angles of each joint with respect to it‘s parent in the skeleton hierarchy are used to compute the features. For example, in

order to calculate the feature at the left elbow joint, first the sensor coordinate system at this joint is translated such that the origin is at the left shoulder. Then a local spherical coordinate system is constructed in terms of an elevation angle from the XY plane and an azimuth angle from the positive X axis. The methods in [75, 95] also employ joint angles as features for action recognition with [95] using quaternions for representing rotations. In [130] the azimuth and elevation angles of the hip centre joint are divided into equal sized bins and the angles corresponding to the other joints are assigned to the bins in a probabilistic way. The final descriptor called Histogram of Oriented Joints 3D (HOJ3D) is computed from the histogram bins. Ma *et al.* proposed tree ensemble model to model space and time relations [64]. In [34], a histogram of the directions between joints in the current frame and adjacent frames is used. The resulting descriptor called Histogram of Oriented Displacements (HOD) represents the motion of an object based on the distance it moves. The covariance matrix of the joint positions is used to derive a Covariance of 3D Joints (Cov3DJ) descriptor in [39]. In some methods, authors claim that not all joints provide useful information. Some of them propose to manually select the informative joints [130], while others select sub-set of joints automatically. For instance [75] the most informative joints in a time window are identified based on the relative informativeness of all the joints in that time window. The joints that have high variance of their angular changes are defined as the most informative joints.

It is worth noting that in skeleton based methods many authors decided to use classification methods which can model sequential data (*eg.* HMM, CRFs). The idea behind it is that it is better to let classifier learn how to model temporal evolution of skeleton, than design feature which captures that evolution manually. Lv *et al.* [63] proposed to use Hidden Markov Model (HMM) with set of spatially local features based on single joints, and a combination of joints (they form hierarchical structure). Koppula *et al.* [46] evaluated the interaction between a human and and object. The authors encode a Markov Random Fields using a spatio-temporal sequences. They encode two types of nodes, namely object nodes and sub-activity-nodes, and edges representing the relationship between an object and the human.

2.2.4 Multimodal representation

In previous sections methods which use single modalities were described. In practice it happens very often that given data source provides more than one modality. For instance RGB+depth-map or depth-map+skeleton or all of them. Many authors [140, 116] merge different modalities using "late fusion" strategy, where they use completely separate framework for each modality and then they use weighted average of classification scores obtained for each modality.

Zhao *et al.* [140], proposed a Local Depth Pattern (LDP) descriptor, which is obtained by computing the average depth values in a spatial cell. The cell is constructed from the POI detected in a colour image. Wang *et al.* use skeleton as a POI detector, than they compute Local Occupancy Pattern (LOP) which models appearance on depth-map around each joint position. Finally they classify action using LOP features and joint position features. Cheron *et al.* [16] proposed to use skeleton information with RGB frames. They crop image around manually selected subset of joints. The cropped image are used as an input to pre-trained CNN. Features obtained from the convolutional layer of CNN are then used as an input to SVM classifier.

In 4 we propose to extend trajectory shape descriptor [113] computed on RGB to 3D by incorporating information from depth-map.

2.3 Datasets

As mentioned in chapter 1 we are particularly interested in daily-living action recognition due it's application to health care and robotics. Thus we selected 3 public data-sets which contains daily-living actions to evaluate our methods. In addition we recorded large scale Smarthomes dataset. In table 2.3 we compare datasets used in this thesis with other state-of-the-art datasets to provide the context.

2.3.1 CAD-60

This dataset [107] contains the RGB frames, depth sequences and skeleton. The data was captured Microsoft Kinect sensor. The data set consists of 12 actions performed by 4 subjects. The actions are performed in 5 different environments: office, kitchen, bedroom, bathroom, and living room. All together data-set contain 60 videos.

2.3.2 CAD-120

This dataset [107] contains the RGB frames, depth sequences and skeleton. All together there are 120 videos available. Actions are performed by 4 different subjects performing 10 high-level activities. Each high-level activity was performed three times with different objects. The activities vary from subject to subject significantly in terms of length.

2.3.3 MSRDailyActivity3D

This dataset [116] consists of 16 actions such as: drink, eat, read book, call cellphone, write on a paper, use laptop, use vacuum cleaner, cheer up, sit still, toss paper, play game, lie down on sofa, walk, play guitar, stand up, sit down. Each action is performed by 10

Dataset	Input type	# Subject	# Classes	# Views	# Examples
UCF-101 [103]	RGB	-	101	-	13320
HMDB51 [49]	RGB	-	51	-	6766
MSRDailyActivity3D [116]	RGB+D+Skeleton	10	16	1	320
CAD-60 [107]	RGB+D+Skeleton	4	12	5	60
CAD-120 [107]	RGB+D+Skeleton	4	10	5	120
NTU [96]	RGB+D+IR+Skeleton	40	60	80	56880
Smarthomes	RGB+D	20	28	7	16974

Table 2.3: Comparison of selected datasets for Action Recognition. First two datasets (HMDB51 and UCF-101) focus on action recognition of videos uploaded to the internet, while others focus on daily-living action recognition.

subjects, and each subject performs each action in standing and sitting position, what adds an additional intra-class variation. In total, the dataset contains 320 videos recorded with 640×480 pixels spatial resolution. RGB frames, depth-map and skeleton are available for all videos.

2.3.4 Toyota Smarthomes

This dataset consist of 28 action classes, performed by 20 senior people. The action were recorded in different locations (dining room, kitchen, living room). We used 7 RBG-D sensors to record the dataset, thus we introduce different viewpoint angles. The most important characteristics of Smarthomes dataset is fact that all actions were performed in unsupervised and completely natural way. To achieve that we used the following recording protocol. Each person stayed in the apartment for 8 hours. Day was divided into 1 hour sessions, and in each session person was asked to perform selected actions. For instance in lunch time sessions person was asked to prepare and eat the meal, but no more detailed instructions were given. In this way we managed to gather dataset with actions that are done in completely un-acted and unsupervised way, which was not a case in most of public daily-living datasets. The dataset contains almost 17,000 video clips. The details of the dataset and sample frames are available in appendix A.

Figure 2.1: Sample frames of each action class for CAD60 dataset.

Figure 2.2: Sample frames of each action class for CAD120 dataset.

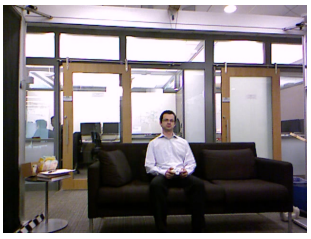

Call cell phone


Cheer up


Drink


Eat


Lie down

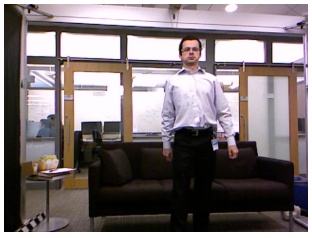

Play game


Play guitar


Readbook


Sit down

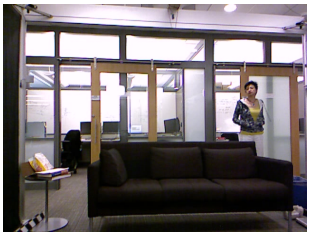

Sit still


Stand up


Toss paper


Use laptop


Use vacuum cleaner


Walk


Write on paper

Figure 2.3: Sample frames of each action class for MSRDailyActivity3D dataset.

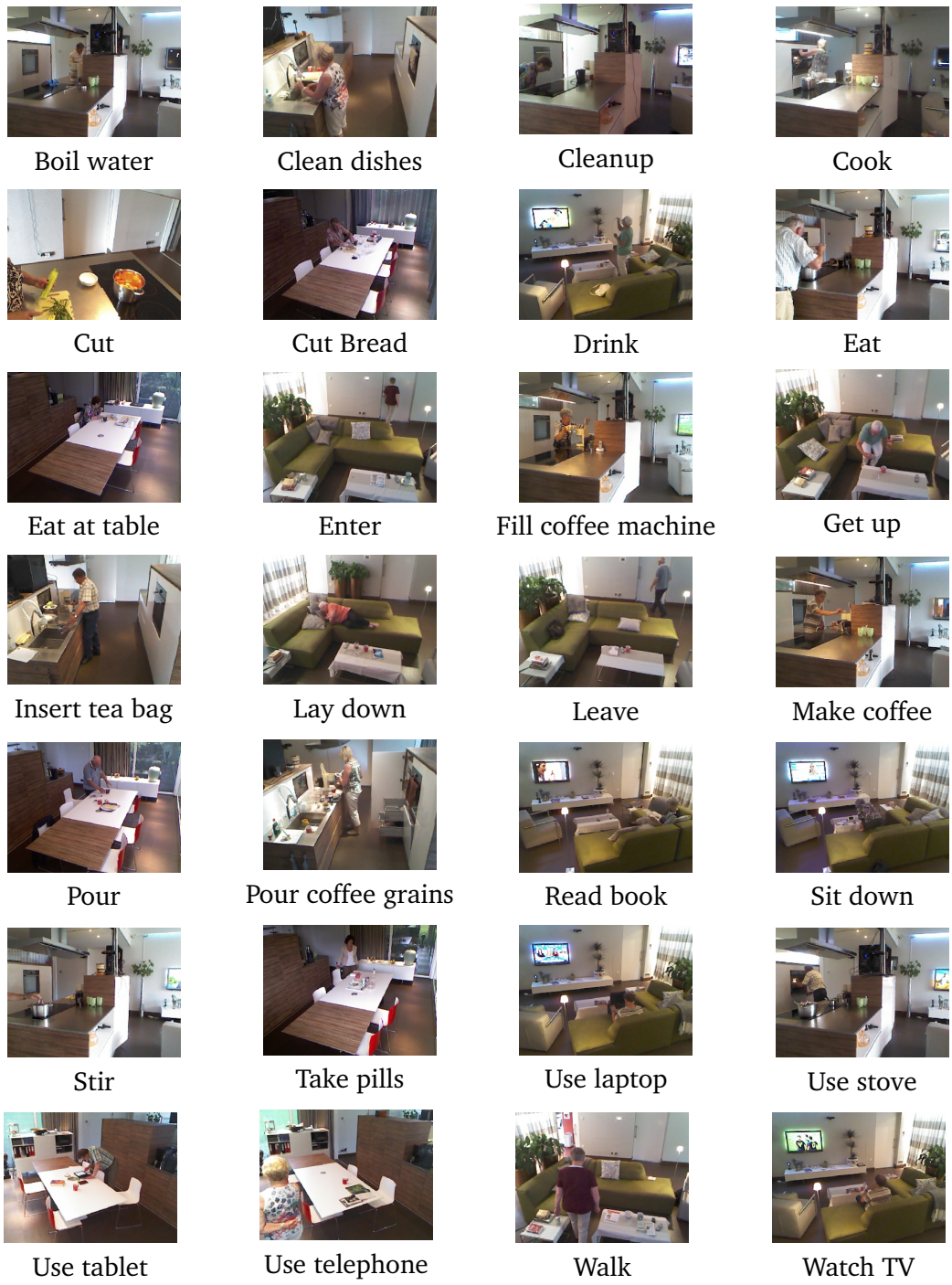


Figure 2.4: Sample frames of each action class for Smarthomes dataset.

Chapter 3

Action Recognition Framework

Contents

3.1 Introduction

In this chapter we introduce the framework for action recognition, which will be used throughout the thesis. The mentioned framework belongs to local feature methods (see chapter 2). This kind of framework has shown good accuracy on various action recognition datasets [113, 52], in addition is robust to scale and view changes. The framework consists of 4 steps:

- features detection,

- descriptor extraction,
- Fisher Vector encoding,
- SVM classification,

the outline of the framework is presented in fig. 3.1.

In this chapter we evaluate two baselines methods: Dense Trajectories (DT) [113], and Trajectory-Pooled Deep-Convolutional Descriptors (TDD) [117]. Both methods use same mechanism for feature detection, Fisher Vector encoding and classification. The main difference is that DT use hand-crafted descriptors, while TDD use CNN feature maps. This gives us opportunity to compare hand-crafted and learnt features based framework and further compare them with methods proposed in this thesis. We compare a performance of both methods on 4 datasets: CAD-60, CAD-120, MSRDailyActivity3D, Smarthomes.

Next we will describe the action recognition framework in section 3.2, then in section 3.3 we will describe TDD feature extraction details, in section 3.4 we describe evaluation metrics details and finally in section 3.5 we provide evaluations results.

3.2 Local Features with Fisher Vector Encoding

In this section we describe a framework for action recognition with local points of interest detection, Fisher Vector encoding and SVM classifier. The framework will be used for action classification with methods proposed in this thesis. The described framework can be break up into 4 parts (see fig. 3.1): local features detection, descriptor extraction, Fisher Vector encoding, SVM classification.

3.2.1 Local Features Detection

Different local features detectors were proposed throughout the years [51, 23]. The key idea of feature detection is to find salient local points of interest. In our action recognition framework we selected Dense Trajectories proposed by Wang *et al.* [113]. This method densely samples points of interest and then track them in subsequent video frames.

In Dense Trajectories local points of interest (POI) are sampled from the dense grid spaced by W pixels. To make sure that POIs cover all spatial positions and scales, the samples are taken from different spatial scales separately. The sampling is carried out with step $W = 5$ and we use the same value as original authors. The POIs sampled from homogeneous region will be difficult to track, so to remove POIs sampled from such areas, the following criterion is applied: if the eigenvalues of auto-correlation matrix are below

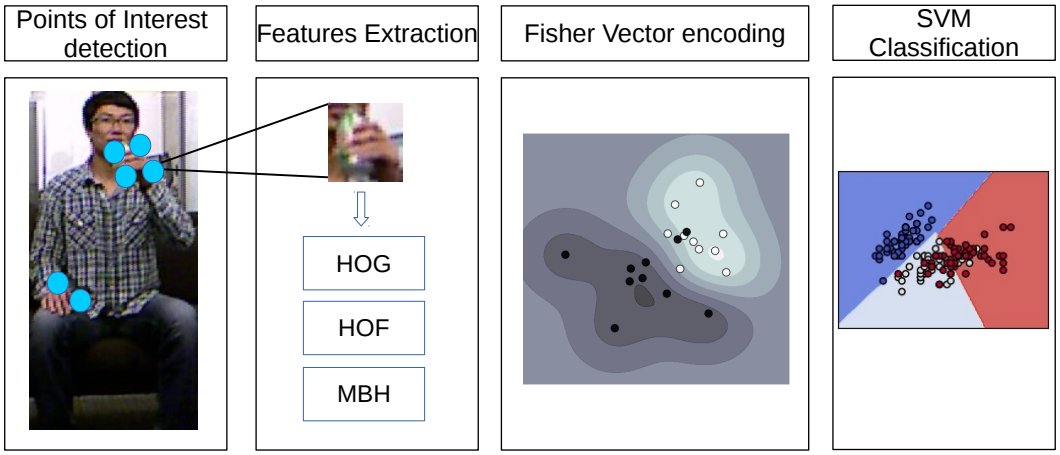


Figure 3.1: Action recognition with: local features extraction, Fisher Vector encoding and SVM classifier – outline. In first step local points of interest are detected. In second step descriptors (*eg.* HOG, HOF, MBH) are computed in local neighborhood of detected points of interest. In third step GMM model is computed and features are encoded to Fisher Vector. Videos are classified using linear SVM with Fisher Vector representation.

the threshold I – then the POI is removed:

$$T = 0.001 \times \max_{i \in I} \min(\lambda_i^1, \lambda_i^2), \tag{3.1}$$

where $(\lambda_i^1, \lambda_i^2)$ are the eigenvalues of point i in the image I. The original authors claim that constant 0.001 provides good compromise between saliency and density of the sampled points.

Once POIs are extracted, they are tracked in the subsequent video frames. For each frame I_t – optical flow field w_t is extracted based on current I_t and next frame I_{t+1}. POI $\boldsymbol{p_t} = (x_t, y_t)$ from frame I_t is tracked in frame I_{t+1} based on optical flow field, and smoothed by median filter on w_t:

$$\boldsymbol{p}_{t+1} = (x_{t+1}, y_{t+1}) = (x_t, y_t) + (\boldsymbol{M} * \boldsymbol{\omega})|_{\overline{x}_t, \overline{y}_t} \tag{3.2}$$

where $\boldsymbol{M}$ is the median filtering kernel, and $(\overline{x}_t, \overline{y}_t)$ is the rounded position of (x_t, y_t).

3.2.2 Descriptor Extraction

Once points of interest are detected the descriptors are computed in the local neighborhood of each detected point. In our case trajectory points are treated as points of interest

around which descriptors are computed. In next section we will describe different kind of hand-crafted descriptors that are computed around trajectory points.

3.2.2.1 Trajectory Shape Descriptor

The shape of a trajectory encodes local motion patterns. Given a trajectory of length L it's shape can be described by a sequence of displacement vectors normalized by sum of magnitudes:

$$\boldsymbol{s'} = \frac{(\Delta \boldsymbol{p}_t, \Delta \boldsymbol{p}_{(t+1)}, ..., \Delta \boldsymbol{p}_{t+L-1})}{\sum_{j=t}^{t+L-1} ||\Delta \boldsymbol{p}_j||}$$

The above sequence will be referred Trajectory Shape Descriptor(TSD). In the original work authors set length of the trajectory to $L = 15$, thus TSD size is equal to 30.

3.2.2.2 Motion descriptors

Apart from trajectory shape descriptor two other motion descriptors are computed. First spatio-temporal volume of size $N \times N$ pixels and L frames long is constructed. The descriptors (*eg.* HOF or MBH) are computed in each cell of the spatio-temporal volume, and the final descriptor is a concatenation of these descriptors. The default parameter for our experiments is $N = 32$ and $L = 15$.

Both HOF and MBH [20] are computed based on optical flow. HOF descriptor computes direction of optical flow in each spatio-temporal volume. The direction are grouped in 8 bins, and additional zero is added, thus an histogram with 9 bins is created. The spatio-temporal volume is divided in 4 spatial and 3 temporal grids, thus final size of HOF descriptor is 109 (*i.e.* $2 \times 2 \times 3 \times 9$).

In case of MBH descriptor first derivatives of optical flow *w.r.t.* x, and y directions are computed. Then orientation information is encoded into 8 bins, for each direction separately (x and y). Similarly to HOF spatio-temporal volume is divided into 12 grids, thus final MBHx and MBHy descriptor size is equal to 96 (*i.e.* $2 \times 2 \times 3 \times 8$). Final MBH descriptor is obtained by concatenation of MBHx and MBHy resulting in 192 dimensions.

3.2.2.3 Appearance descriptors

To model appearance information involved in action HOG [20] is computed. It is done in similar manner as for HOF and MBH, but this time descriptor is computed on RGB frame instead of optical flow.

3.2.3 Fisher Vector Encoding

Once the descriptors are extracted, we use them to create video representations. We encode a video sequence using first and second order statistics of a distribution of a feature set $\mathbb{X}$, based on Fisher vectors [81, 82]. We model features with a generative model and compute the gradient of their likelihood with respect to the parameters of the model, *i.e.* $\Delta_\lambda \log p(\mathbb{X}|\lambda)$. We describe how the set of features deviates from an average distribution of features, modeled by a parametric generative model. Firstly, during the preliminary learning stage, we fit a K-centroid Gaussian Mixture Model (GMM) to our training features, which can be regarded as a soft visual vocabulary:

$$p(x_i|\lambda) = \sum_{j=1}^{M} w_j g(x_i|\mu_j, \Sigma_j), \tag{3.3}$$

$$\mathbf{s.t.} \quad \forall_j : w_j \geq 0, \quad \sum_{j=1}^{M} w_j = 1, \tag{3.4}$$

$$g(x_i|\mu_j, \Sigma_j) = \frac{1}{(2\pi)^{D/2}|\Sigma_j|^{1/2}} e^{-\frac{1}{2}(x_i-\mu_j)^T \Sigma_j^{-1}(x_i-\mu_j)}, \tag{3.5}$$

where $x_i \in \mathbb{X}$ is a D-dimensional feature vector, $\{g(x_i|\mu_j, \Sigma_j)\}_{j=1}^{M}$ are the component Gaussian densities and $\lambda = \{w_j, \mu_j, \Sigma_j\}_{j=1}^{M}$ are the parameters of the model, respectively the mixture weights $w_j \in \mathbb{R}_+$, the mean vector $\mu_j \in \mathbb{R}^D$ and the positive definite covariance matrices $\Sigma_j \in \mathbb{R}^{D\times D}$ of each Gaussian component. We learn the parameters λ using the Expectation Maximization restricting the covariance of the distribution to be diagonal. To estimate the GMM parameters, we randomly sample a subset of $256,000$ features from the training set. To increase the precision, we initialize GMM ten times and we keep the codebook with the lowest error. We define the soft assignment of descriptor x_i to the Gaussian j as a posteriori probability $\gamma(j|x_i, \lambda)$ for component j:

$$\gamma(j|x_i, \lambda) = \frac{w_j g(x_i|\mu_j, \Sigma_j)}{\sum_{l=1}^{M} w_l g(x_i|\mu_l, \Sigma_l)}, \tag{3.6}$$

Then, we compute the gradients of the j-th component with respect to μ and σ, using the following derivations:

$$G_{\mu,j}^{\mathbb{X}} = \frac{1}{N_x\sqrt{w_j}} \sum_{l=1}^{N_x} \gamma(j|x_l, \lambda) \left(\frac{x_l - \mu_j}{\sigma_j}\right), \tag{3.7}$$

$$G_{\sigma,j}^{\mathbb{X}} = \frac{1}{N_x\sqrt{2w_j}} \sum_{l=1}^{N_x} \gamma(j|x_l, \lambda) \left(\frac{(x_l - \mu_j)^2}{\sigma_j^2} - 1\right), \tag{3.8}$$

where N_x is the cardinality of the set $\mathbb{X}$. Finally, we encode a set of local descriptors $\mathbb{X}$ as a concatenation of partial derivatives with respect to the mean $G^{\mathbb{X}}_{\mu,j}$ and standard deviation $G^{\mathbb{X}}_{\sigma,j}$ parameters for all M components:

$$V = [G^{\mathbb{X}}_{\mu,1}, G^{\mathbb{X}}_{\sigma,1}, ..., G^{\mathbb{X}}_{\mu,M}, G^{\mathbb{X}}_{\sigma,M}]^T. \tag{3.9}$$

As a final step, we apply the power normalization and L2-normalization. The dimension of the Fisher vector representation is $2KD$.

3.2.4 SVM

In this section we describe the Support Vector Machines (SVM) classifier which is used to assign action categories for each video represented as Fisher Vector. SVM is a supervised machine learning technique, which means that in training process input features as well as ground truth labels have to be provided. The idea of SVM was proposed by Vapnik *et al.* [110]. The method became popular in computer vision community [25, 24, 123] in many different areas such as: object recognition [81, 82] and action recognition [113, 51]. The SVM objective is to find hyperplane, which separates two classes with maximum margin. Rationale behind maximum margin separation hyperplane is as follows. If we pick separation hyperplane with small margin, then any slight perturbations to the decision boundary can have significant impact on classification results. Thus classifiers which produce boundaries with small decision boundary with small margin are susceptible to overfitting. A more formal explanation which relates decision boundary of linear classifiers to their generalization error is based on Structural Risk Minimization (SRM) theory, the detail can found in [109]. It is worth noting that in principle SVM is a linear classifier, but it can be extended to non-linear by kernel transformation.

In action recognition framework used in this thesis we will use linear SVM, in particular C-SVM formulation proposed by [17]. Let's assume that we have N training examples $\{(\boldsymbol{x_i}, y_i), \boldsymbol{x_i} \in \mathbb{R}^p, y_i \in \{-1, 1\}, i = 1, ...N\}$. The learning task in SVM can be defined as the following optimization problem:

$$\min_{\boldsymbol{w}} \frac{\|\boldsymbol{w}\|^2}{2} + C \sum_{i=1}^{N} \xi_i \tag{3.10}$$

$$\text{subject to } y_i(\boldsymbol{w} * \boldsymbol{x_i} + b) \geq 1, i = 1, 2, ..., N. \tag{3.11}$$

Where $\boldsymbol{w}$ is a normal vector to separation hyperplane, ξ_i is slack variable which controls the amount of misclassification on training set, C is regularization parameter, which control the penalty cost for each misclassified example in training set. Because the objective function is quadratic and constraints are linear, the optimization problem is convex.

Linear SVM has also two other key properties: works well with high dimensional data [109] and is computationally effective, because training complexity is proportional to $O(Np)$ and testing complexity is $O(p)$. The above properties of SVM makes it suitable for the framework used in this thesis as Fisher Vector representing video is usually high dimensional.

SVM is a binary classifier, but in action recognition we typically deal with multiclass problems. To overcome this limitation in our classification framework we use one-versus-all strategy. In this case we construct C binary classifier one for each class. One-versus-one strategy could be considered as alternative, but it was not chosen because of its big memory footprint. In addition Zhang *et al.* [139] claim that one-versus-one strategy does not lead to improved accuracy rate over one-versus-all.

In our experiments we use SVM implementation provided by scikit-learn [80] library.

3.3 Trajectory-Pooled Deep-Convolutional Descriptors (TDD)

In this section we introduce Trajectory-Pooled Deep-Convolutional Descriptors (TDD) [117] that will be used as a second baseline in this thesis. The authors utilize deep architectures to learn discriminative convolutional feature maps. Then they pool values of the convolutional map around the trajectory points detected with Dense Trajectories method. To enhance the robustness of TDDs, they design two normalization methods to transform convolutional feature maps, namely spatio-temporal normalization and channel normalization. As we can see the described TDD method is closely related to framework described in section 3.2, the crucial difference arises in feature extraction step. In Dense Trajectories hand-crafted features are extracted while in TDD values of convolutional filters obtained from Convolutional Neural Network (CNN) are used to form the descriptor.

To extract the motion features the conv3 and conv4 layer of CNN is used. The CNN in this case was trained on optical flow and we will refer it as motion stream CNN. To extract appearance features the conv4 and conv5 layer of CNN is used. The CNN is trained on RGB frames and we will refer it as appearance stream CNN. The overview of whole framework is illustrated in fig. 3.2.

Both CNN streams are trained based on The Two Stream Network proposed by Simonyan *et al.* [100]. Below we briefly describe training process of CNNs used with TDD method.

3.3.1 Appearance stream

The appearance net operates on single RGB frames as an input. In our experiments we use VGG-16 net [102], pre-trained on ImageNet [22] and on UCF-101dataset. The mentioned

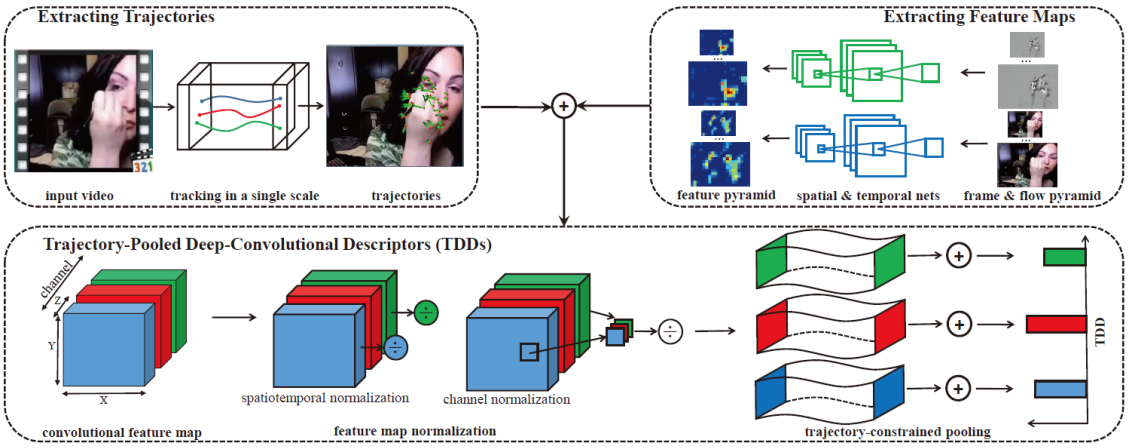


Figure 3.2: TDD overview. The descriptor computation involves 3 steps: (i) trajectories extraction using Dense Trajectories method, (ii) multi-scale convolutional feature maps extraction, and (iii) TDD calculation by stacking convolutional feature maps obtained from trajectory points. After TDD descriptor is computed framework with Fisher Vector encoding and linear SVM is applied.

VGG-16 network consists of 16 weighted layers. The input image is expected to be frame of 224×224 pixels followed by 13 convolutional layers and 3 fully connected layers. The key feature of the net is that it uses small filters of size 3×3, which allows to build deeper neural net. In training phase we sample frames M from each video of the dataset. Then we crop random patch of size 224×224 from each sampled frame. The patch can be cropped either from top-left, top-right, bottom-left, bottom-right, or center. Each patch can be randomly flipped from left to right.

3.3.2 Temporal stream

The temporal net consists of same VGG-16 architecture, but is trained on stacked optical-flow. The optical-flow can be seen as a set of displacement fields $\boldsymbol{d}_t$ between consecutive frames t and $t+1$. Then $\boldsymbol{d}_t(u, v)$ represents displacement vector of pixel at point (u, v) from frame t to frame $t + 1$. The input to the temporal net is a stack of $2L$ optical-flow fields: L computed in forward temporal direction *w.r.t.* to frame of interest and L computed in backward temporal direction. Training process is similar to appearance net: we randomly sample M frames from each video. For each frame we compute stacked optical-flow of size $L = 10$. Then we apply random cropping and flipping in the same way as in appearance net.

3.4 Evaluation Metric

3.4.1 Mean Class Accuracy Metric

To evaluate a performance of proposed and baseline methods we will use Mean Class Accuracy metric. Let's first define accuracy metric for selected class c:

$$acc_c = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} 1(y_i = c) \tag{3.12}$$

where y_i label assigned to sample i, $1(x)$ is the indicator function. Then Mean Class Accuracy metric can be defined as:

$$Acc = \frac{1}{|\mathbb{C}|} \sum_{c \in \mathbb{C}} acc_c \tag{3.13}$$

where $\mathbb{C}$ is set of action classes.

Action recognition framework used in this thesis introduces random factors such as: different codebook assignment which depends on random initialization in Fisher Vector representation, or in case of deep neural networks convergence to different local optimums. Thus to be able to compare relative improvement of selected methods we run all experiments at least 2 times and report also standard deviation of Mean Class Accuracy metric.

3.4.2 Data Splits and Cross-Validation

For each experiment we divide data into three splits: training split with examples used for training the classifier, validation split used to tune the parameters such as learning rate, regularization rate, *etc.* and test split which contains data used for final assessment of the performance.

If training, testing partition information is provided for given dataset, we follow same partition in our experiments. To split training set into training split and validation split we follow either k-fold cross-validation or leave-one-person out schema depending on dataset size.

3.5 Experiments

3.5.1 Dense Trajectories

In this section we present a evaluation of the baseline method based on Dense Trajectories, Fisher Vector representation and linear SVM. We will use 6 different sizes of codebooks:

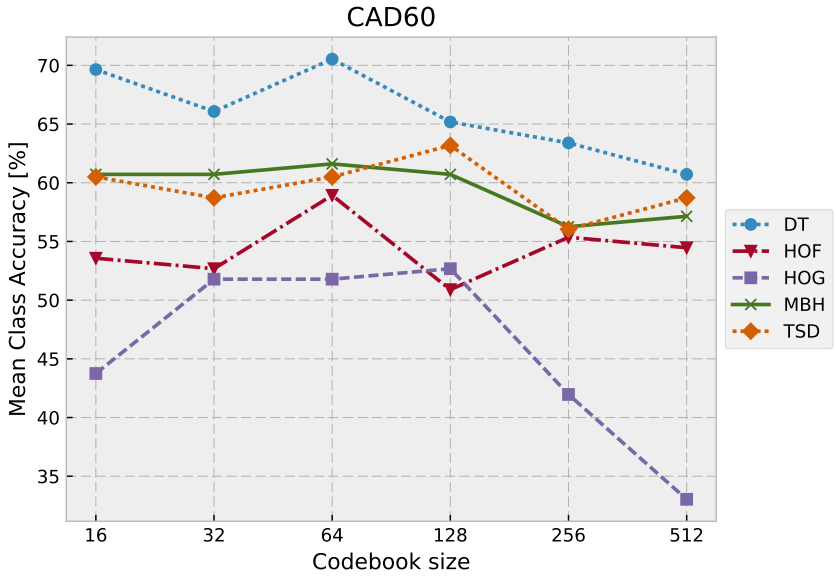
16, 32, 64 128, 256, 512, 7 different descriptors: Trajectory Shape Descriptor (TSD), HOG, HOG, MBHx, MBHy, MBH and fusion of all mentioned descriptor refereed as DT (Dense Trajectories). The experiments will be performed on 4 datasets: CAD-60, CAD-120, MSR-DailyActivity3D, Smarthomes. The datasets differ in terms of number of action classes, examples and challenges (detailed description of dataset is available in section 2.3).

3.5.1.1 CAD-60

The results are reported in table 3.1. By analyzing the experiments we can draw some of conclusions:

- Small codebook sizes work best on this data set. There is trend where accuracy decreases, while codebook size increases. The reason behind it, might be small size of dataset. With low amount of data and big size of codebook the overfitting is becoming an issue.
- Feature fusion reports the best accuracy, while MBH, TSD descriptors achieve comparable results.
- HOG achieves the worst results. The reason why HOG descriptor itself does not perform well is that it focuses only on appearance, which is not a key information in action recognition in general. In addition actions are performed in apartment so there is also less information that can be encoded from the background appearance.

Table 3.1: Results of Dense Trajectories method on CAD-60 dataset. The plot shows Mean Class Accuracy *w.r.t.* codebook size. The σ provided in table states for standard deviation of the results. Last column indicates best result across all codebook sizes.



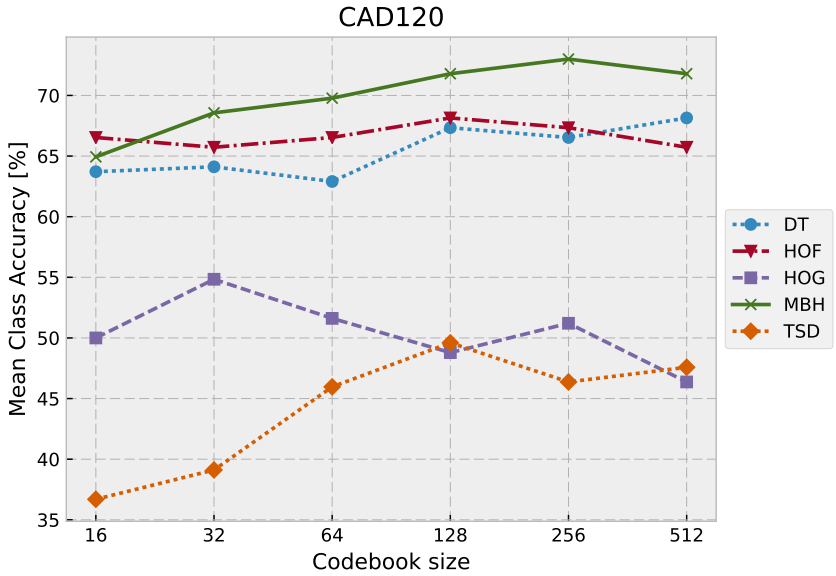
	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
DT	69.64	2.53	66.07	2.53	70.54	1.26	65.18	1.26	63.39	1.26	60.71	2.53	70.54	1.26
TSD	60.50	2.53	58.71	5.05	60.50	0.00	63.18	1.26	56.04	3.79	58.71	0.00	63.18	1.26
MBH	60.71	2.53	60.71	2.53	61.61	1.26	60.71	2.53	56.25	3.79	57.14	0.00	61.61	1.26
HOF	53.57	2.53	52.68	1.26	58.93	0.00	50.89	1.26	55.36	5.05	54.46	1.26	58.93	0.00
HOG	43.75	1.26	51.79	0.00	51.79	0.00	52.68	3.79	41.96	6.31	33.04	1.26	52.68	3.79

3.5.1.2 CAD-120

The results are reported in table 3.2. The CAD-120 dataset differs from CAD-60 in the way that it contains more examples, but less action classes.

- The accuracy starts to drop when codebook size is bigger then 256 for most of the descriptors.
- We can observe that despite of size of the codebook – the best performing descriptor is MBH, followed by DT. The reason for such a performance of MBH is that CAD-120 contains 4 out of 10 action classes which involve a lot of vertical motion (arranging objects, stacking objects, unstacking objects, picking objects). The MBH descriptor consist on X and Y component, thus is able to to model the actions better than for instance HOF descriptor.
- Fusion of all descriptors (DT) did not achieve the best results. The reason for that is underperfomance of TSD descriptor, which does not contain complementary information, thus the fusion of TSD with other descriptors leads to performance drop.
- TSD achieve low performance on CAD-120 dataset, because it contains many actions with similar motion patterns *i.e.* all actions which involve object manipulation. It seems that trajectory shape is not sufficient to encode the motion information, while detected trajectory points itself are informative, because other descriptors computed around the trajectory points achieve better results.

Table 3.2: Results of Dense Trajectories method on CAD-120 dataset. The plot shows Mean Class Accuracy *w.r.t.* codebook size. The σ provided in table states for standard deviation of the results. Last column indicates best result across all codebook sizes.



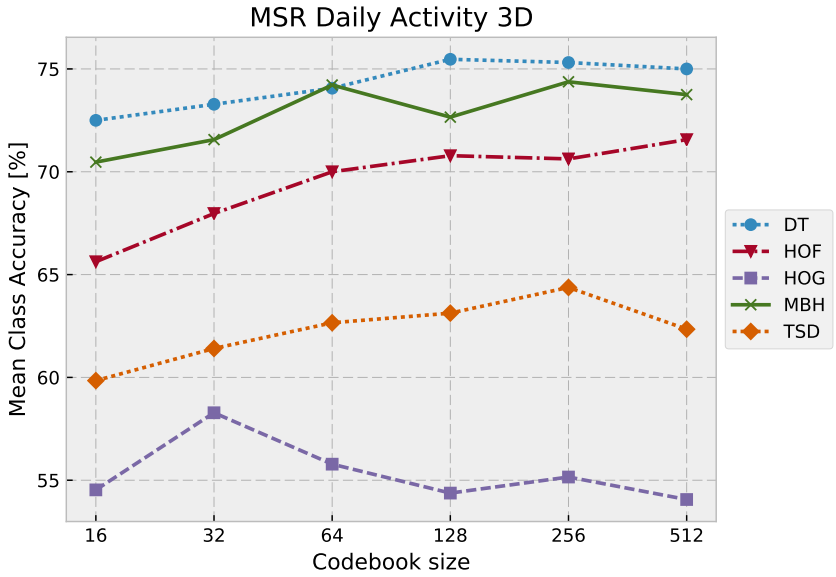
	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
MBH	64.94	1.14	68.56	1.71	69.77	0.00	71.79	2.85	73.00	1.14	71.79	3.99	73.00	1.14
DT	63.71	0.00	64.11	1.71	62.90	1.14	67.34	1.71	66.53	1.71	68.15	2.85	68.15	2.85
HOF	66.53	5.13	65.73	3.99	66.53	0.57	68.15	0.57	67.34	0.57	65.73	0.57	68.15	0.57
HOG	50.00	2.28	54.84	1.14	51.61	1.14	48.79	2.85	51.21	5.13	46.37	3.99	54.84	1.14
TSD	36.69	0.57	39.11	1.71	45.97	1.14	49.60	1.71	46.37	0.57	47.58	1.14	49.60	1.71

3.5.1.3 MSRDailyActivity3D

The results are reported in table 3.3. The MSRDailyActivity3D dataset contains more action classes and more examples than CAD-60 and CAD-120.

- Similarly to CAD-120 the performance drops when codebook size is bigger than 256.
- We can observe that despite of size of the codebook that the best performing descriptor is DT, followed by MBH. Thus fusion of descriptors leads to performance gain. It is worth to point out that fusion works, in spite of the fact that HOG is significantly underperforming. The reason why it works, is that HOG encodes appearance information, while other descriptors encode motion information, thus fusion leads to performance gain, even though HOG alone achieves low accuracy. This is the opposite situation comparing to CAD-120 dataset, as there TSD was underperfoming and did not encode complementary information, thus fusion did not work.

Table 3.3: Results of Dense Trajectories method on MSRDailyActivity3D dataset. The plot shows Mean Class Accuracy *w.r.t.* codebook size. The σ provided in table states for standard deviation of the results. Last column indicates best result across all codebook sizes.



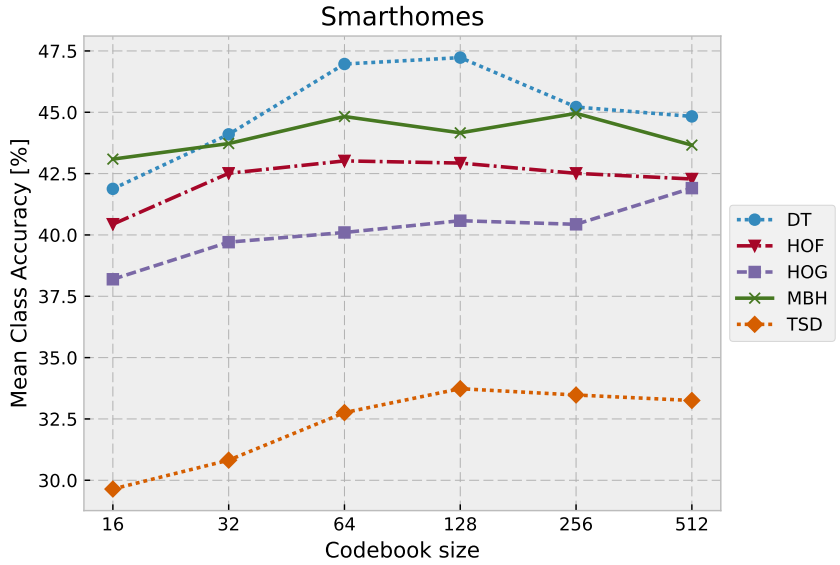
	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
DT	72.50	0.88	73.28	0.22	74.06	0.88	75.47	0.22	75.31	1.33	75.00	0.00	75.47	0.22
MBH	70.47	1.55	71.56	0.88	74.22	1.10	72.66	0.66	74.38	0.00	73.75	0.00	74.38	0.00
HOF	65.62	0.88	67.97	0.22	70.00	1.33	70.78	0.22	70.62	0.88	71.56	0.44	71.56	0.44
TSD	59.84	1.10	61.41	1.55	62.66	1.10	63.12	1.33	64.38	0.00	62.34	1.55	64.38	0.00
HOG	54.53	0.22	58.28	0.66	55.78	0.66	54.37	0.44	55.16	0.22	54.06	0.00	58.28	0.66

3.5.1.4 Smarthomes

The results are reported in table 3.4. The Smarthomes dataset is the biggest datasets among ones we run our experiments. It has the biggest class number (28), the biggest number of different people (20) and by far the biggest number of examples (17k).

- Big codebook sizes work best on this data set. There is clear trend where accuracy increases, while codebook size increases. This can be explained with the fact that the dataset contains 28 action classes in addition many actions of same class are performed in different ways. This introduces high variance in the descriptors, which has to be encoded with bigger number of codewords. In addition with 17k examples the overfitting is not a big problem.
- We can observe that despite of size of the codebook the best performing descriptor is DT, followed by MBH, and HOF.
- Again HOG is one of the worst performing descriptor, but it does not harm fusion accuracy, because HOG encodes appearance information, which is complementary to other motion descriptors.
- Detailed analysis of shows that Dense Trajectories have problem with recognition actions such as: "eat snack", which is confused with "drink", "take pills" confused with "drink" or "use tablet" confused with "use laptop". The first two pairs of actions share similar motion pattern, while last pair shares also similar appearance.

Table 3.4: Results on Smarthomes dataset, the plot shows Mean Class Accuracy *w.r.t.* codebook size.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
DT	41.88	0.07	44.10	0.04	46.97	0.77	47.23	0.20	45.21	2.36	44.83	0.23	47.23	0.20
MBH	43.09	0.25	43.73	0.29	44.83	0.34	44.16	0.49	44.96	0.07	43.66	0.16	44.96	0.07
HOF	40.44	0.11	42.51	0.37	43.02	0.67	42.93	0.33	42.51	0.12	42.28	0.41	43.02	0.67
HOG	38.19	0.08	39.71	0.13	40.10	0.60	40.58	0.79	40.43	0.74	41.91	0.34	41.91	0.34
TSD	29.64	0.01	30.82	0.29	32.76	0.08	33.73	0.56	33.48	0.92	33.25	0.36	33.73	0.56

3.5.2 Trajectory-Pooled Deep-Convolutional Descriptors (TDD)

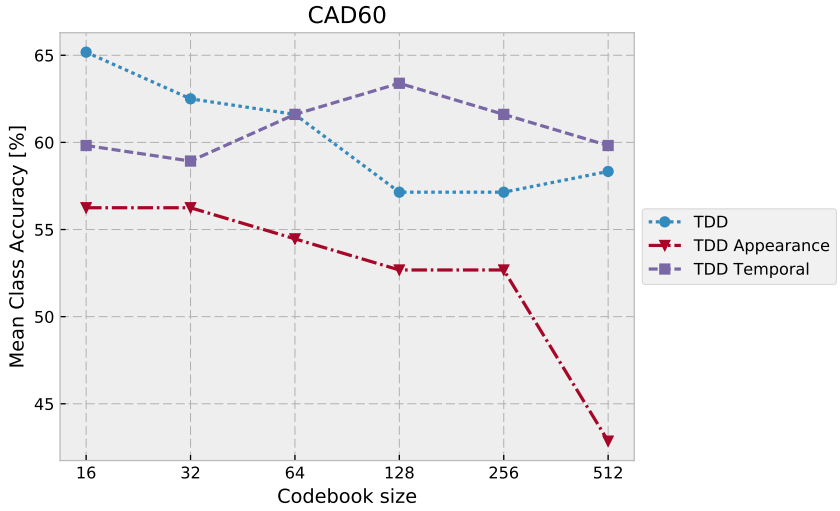
In this section we present an evaluation of the baseline method based on Trajectory-Pooled Deep-Convolutional Descriptors, Fisher Vector representation and linear SVM. We will use 6 different sizes of codebooks: 16, 32, 64 128, 256, 512, 3 different descriptors: Spatial TDD, Temporal TDD and fusion of two above – TDD descriptor. The experiments will be performed on 4 datasets: CAD-60, CAD-120, MSRDailyActivity3D, Smarthomes. The datasets differs in terms of number of action classes, examples and challenges (detailed description of dataset is available in section 2.3).

3.5.2.1 CAD-60

The results are reported in table 3.5. As in previous section we can draw some conclusion from the experimental results

- Accuracy drops for codebook sizes bigger than 128.
- Temporal TDD descriptor performs better than Spatial TDD. This is another confirmation of the fact that motion features play more important role in action recognition.
- Further fusion of Temporal TDD and Appearance TDD leads to performance improvements.

Table 3.5: Results of Trajectory-Pooled Deep-Convolutional Descriptors method on CAD-60 dataset. The plot shows Mean Class Accuracy *w.r.t.* codebook size. The σ provided in table states for standard deviation of the results. Last column indicates best result across all codebook sizes.



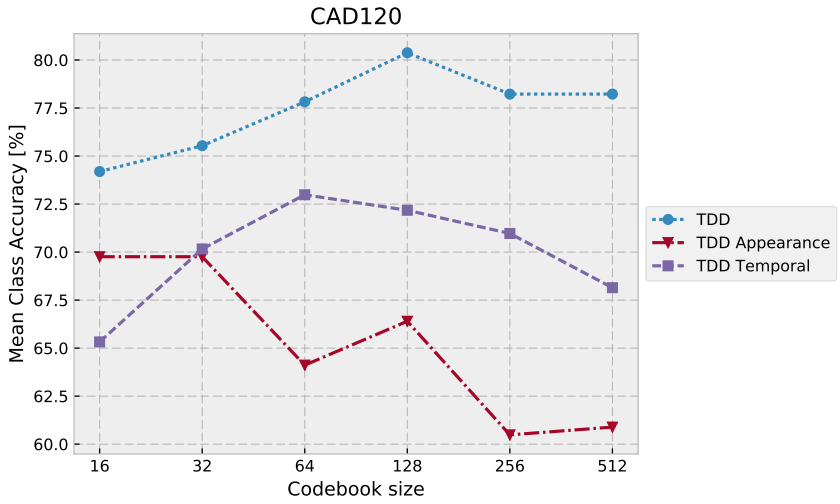
	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
TDD	65.18	1.26	62.50	2.53	61.61	1.26	57.14	2.53	57.14	0.00	58.33	1.68	65.18	1.26
TDD Temporal	59.82	3.79	58.93	0.00	61.61	1.26	63.39	1.26	61.61	1.26	59.82	1.26	63.39	1.26
TDD Appearance	56.25	3.79	56.25	1.26	54.46	1.26	52.68	1.26	52.68	1.26	42.86	6.73	56.25	3.79

3.5.2.2 CAD-120

The results are reported in table 3.6. The conclusion from experimental results are pretty aligned with ones for CAD-60.

- Accuracy drops for codebook sizes bigger than 128.
- Temporal TDD descriptor performs better than Appearance TDD. This is another confirmation of the fact that motion features play more important role in action recognition.
- Further fusion of Temporal TDD and Appearance TDD leads to performance improvements. It seems that Appearance TDD and Temporal TDD carry a lot of complementary information, because fusion leads to significant performance improvement.

Table 3.6: Results of Trajectory-Pooled Deep-Convolutional Descriptors method on CAD-120 dataset. The plot shows Mean Class Accuracy *w.r.t.* codebook size. The σ provided in table states for standard deviation of the results. Last column indicates best result across all codebook sizes.



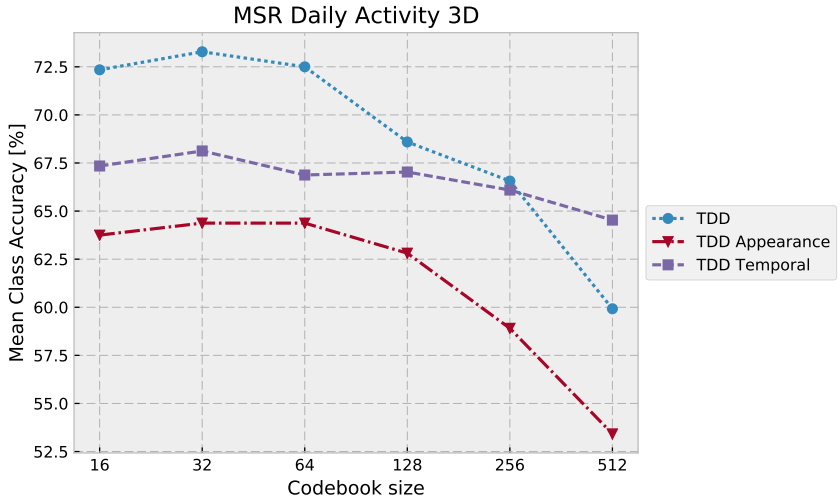
	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
TDD	74.19	1.14	75.54	0.38	77.82	1.71	80.38	1.90	78.23	1.14	78.23	3.42	80.38	1.90
TDD Temporal	65.32	1.14	70.16	1.14	72.98	1.71	72.18	1.71	70.97	1.14	68.15	1.71	72.98	1.71
TDD Appearance	69.76	3.99	69.76	2.85	64.11	1.71	66.40	4.18	60.48	3.42	60.89	0.57	69.76	3.99

3.5.2.3 MSRDailyActivity3D

The results are reported in table 3.7. The conclusion from experimental results are pretty aligned with ones for CAD-60.

- Accuracy drops for codebook sizes bigger than 64.
- Temporal TDD descriptor performs better than Appearance TDD.
- Further fusion of Temporal TDD and Appearance TDD leads to performance improvements. It seems that Appearance TDD and Temporal TDD carry a lot of complementary information, because fusion leads to significant performance improvement.

Table 3.7: Results of Trajectory-Pooled Deep-Convolutional Descriptors method on MSRDailyActivity3D dataset. The plot shows Mean Class Accuracy *w.r.t.* codebook size. The σ provided in table states for standard deviation of the results. Last column indicates best result across all codebook sizes.



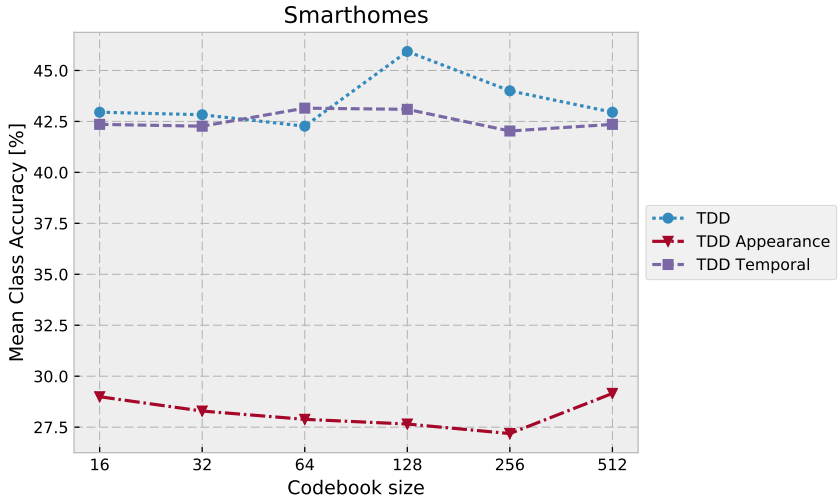
	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
TDD	72.34	1.10	73.28	0.22	72.50	0.44	68.59	0.22	66.56	0.44	59.92	1.88	73.28	0.22
TDD Temporal	67.34	0.22	68.12	0.44	66.88	0.44	67.03	0.22	66.09	0.66	64.53	0.66	68.12	0.44
TDD Appearance	63.75	0.44	64.38	0.44	64.38	1.33	62.81	0.88	58.91	0.22	53.42	0.91	64.38	0.44

3.5.2.4 Smarthomes

The results are reported in table 3.8. In TDD descriptor performs very close to DT descriptor in many aspects:

- It achieves better performance with bigger codebook sizes (128, 256), similarly as DT descriptor.
- Motion features "TDD Temporal" perform better than appearance features, similarly as MBH and HOG in DT. The difference here is that performance gape between descriptors is bigger, but "TDD Appearance" still caries complementary information as fusion leads to accuracy improvement.
- Detailed analysis of also shows that TDD descriptor makes errors on similar action pairs as DT.

Table 3.8: Results on Smarthomes dataset, the plot shows Mean Class Accuracy *w.r.t.* codebook size.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
TDD	42.95	0.16	42.83	0.16	42.27	0.16	45.93	0.16	44.00	0.16	42.95	0.16	45.93	0.16
TDD Temporal	42.36	0.13	42.27	1.33	43.15	0.18	43.09	0.60	42.03	0.12	42.36	0.13	43.15	0.18
TDD Appearance	28.99	0.77	28.29	0.63	27.88	0.29	27.65	0.50	27.19	0.73	29.16	0.52	29.16	0.52

Table 3.9: Accuracy comparison between Dense Trajectories (DT) and Trajectory-Pooled Deep-Convolutional Descriptors (TDD). The DT method uses hand-crafted features while TDD uses features from deep Convolutional Neural Network.

	CAD60	CAD120	MSR DailyActivity3D	Smarthomes
DT	70.54	73.00	75.47	47.23
TDD	65.18	80.38	73.28	45.93

3.6 Conclusions

In this chapter we have introduced action recognition framework which will be used throughout the thesis. The framework is based on local features detection and extraction. We use Fisher Vector encoding and linear SVM. We evaluate two baseline methods which fit to our framework: Dense Trajectories, Trajectory-Pooled Deep-Convolutional Descriptors which use same detection method as Dense Trajectories, but extract features based on convolutional maps from neural network. Based on the results we can draw couple of general conclusions. First, the bigger is dataset the bigger codebook size can be used. In case of small datasets codebook with big number of codewords tend to overfit the training data. Second, temporal features (HOF, MBH, Temporal TDD) outperform appearance descriptors (HOG, Appearance TDD). Third, fusion of temporal and appearance descriptors in most cases leads to accuracy improvements.

In table table 3.9 we provide comparison of DT and TDD methods. The results from table show that hand-crafted features and than CNN features achieve competitive results.

All baseline methods ignore spatial and temporal location of features – information is lost in Fisher Vector encoding. In addition feature detection is based on optical flow, thus action which do not involve much amount of motion are not detected due to low number of detected features.

Chapter 4

3D Trajectories

Contents

4.1 Introduction

The space that we live in is 3 dimensional, but when we use RGB camera our sensing capabilities are limited to only 2 dimensions. Encouraged by recent development in affordable depth sensors (*eg.* Kinect, Asus Xtion) and inspired by the fact that many animals and humans are equipped with binocular vision system, which provides depth information about the observed scene, we propose new feature for action recognition which takes advantage of 3D information obtained from RGB-D sensor.

The depth-map obtained from RGB-D sensor introduces many challenges: noise of measured depth-value grows exponentially with the distance from the sensor, missing depth-values for areas where IR beam from the sensor was occluded or absorbed (*eg.* by

black cloth). Due to above reasons the depth-map introduces different modality comparing to RGB and cannot be treated in the same way as colour channels.

Most effective state-of-the-art action recognition approaches use depth information to obtain pose information. This is done by applying skeleton detection to obtain body joints positions. Then the detected body joints are used to model the actions. Such strategy is effective because skeleton detection is much easier on depth-map than on RGB image (more information can be found in section 2.2.3). Apart from recent advancements in skeleton detection [14, 124, 40], we claim that it is already a complicated task and it is difficult to assume that the skeleton will be available at any time. Thus inspired by efficacy of Local Methods (see section 2.2.1), we propose to replace modeling of joint positions with modeling of positions of detected Local Points of Interest (POIs). In such case we will loose semantic information – we would not know if POI belongs to head or hand, as it was with skeleton joints. But we will be able to use much simpler and robust detector. Another issue with Local Methods is that they use POI detectors which work on RGB images, thus provide only 2D information about the detected point, while skeleton joints obtained from depth-map provide 3D information. There are two solutions to this problem: to propose a new POI detector which works on depth-map to obtain 3D POIs, but this task is quite difficult due to the fact that depth-map does not carry much texture information and POI detectors are limited in that terms. That is why we propose to use RGB POI detector and then extend detected POIs with information from depth-map to form 3D POIs (see fig. 4.1). Finally inspired by [113] we propose to track 3D POIs and encode their relative displacements to from 3D Trajectory Shape Descriptor (3DTSD) [43]. Such a descriptor combines advantages of RGB based point of interest detection with 3D information obtained from depth-map. In addition is fast to compute (see table 10.1). Our experiments show that 3DTSD performs better than 2D descriptor standalone, as well as fused together with other different descriptors. In this chapter we also explain how to handle noise and missing values in depth-map, which is an important step and has a significant impact on descriptor performance.

4.2 2D Trajectories Detection

In the proposed method we use Dense Trajectories [113] as POI detector, but our method is generic enough to be used together with any other POI detector, which allows to track detected points and provides their spatial positions. The 2D trajectories detection used with the proposed method works as follows: feature points are sampled on a grid spaced by w pixels. Each point $\boldsymbol{p}_t = (x_t, y_t)$ at frame t is tracked to the next frame $t+1$ by median

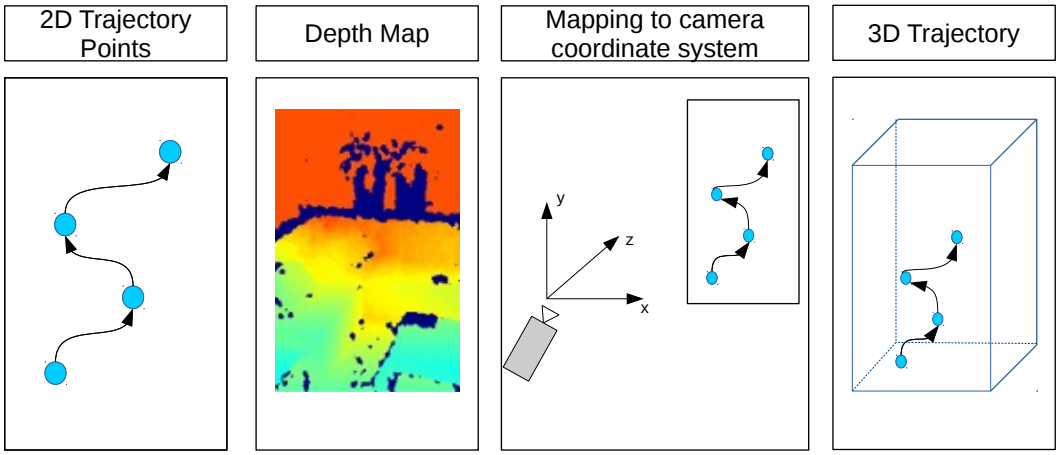


Figure 4.1: This figure shows the steps to obtaining 3D Trajectory points. First 2D trajectories based on RGB image are obtained. Then depth-map is processed with median filtering. In third step 2D points are transformed to 3D camera coordinate system, image x_{img}, y_{img} positions, z from depth-map are transformed using pinhole camera model. In the last step 3D points are stacked together to form trajectory.

filtering in a dense optical flow field $\boldsymbol{\omega}$

$$\boldsymbol{p}_{t+1} = (x_{t+1}, y_{t+1}) = (x_t, y_t) + (\boldsymbol{M} * \boldsymbol{\omega})|_{\overline{x}_t, \overline{y}_t} \tag{4.1}$$

where $\boldsymbol{M}$ is the median filtering kernel and $\boldsymbol{\omega}$ is a optical flow field, and $(\overline{x}_t, \overline{y}_t)$ is the rounded position of (x_t, y_t). Once the dense optical flow field is computed, points can be detected very densely, based on the motion criterion (optical flow). Because of the fact that in action recognition we are mainly interested in dynamic information, the detected 2D points $\boldsymbol{p}_t$ are tracked over the time to form trajectories. Detected static trajectories are pruned in a pre-processing stage. Trajectories with sudden large displacements, most likely to be erroneous, are also removed. This post processing step is explained in section 3.2 and [113].

4.3 2D Trajectory Shape Descriptor (TSD)

Local motion patterns can be encoded by the shape of a trajectory. The shape of the 2D trajectory of length L can be described as sequence $\boldsymbol{s} = (\Delta \boldsymbol{p}_t, \Delta \boldsymbol{p}_{t+1}, ..., \Delta \boldsymbol{p}_{t+L-1})$, where $\Delta \boldsymbol{p}_t$ is a displacement vector $\Delta \boldsymbol{p}_t = (x_{t+1} - x_t, y_{t+1} - y_t)$. Vector $\boldsymbol{s}$ is normalized by sum

of the magnitudes of displacement vectors:

$$\boldsymbol{s'} = \frac{(\Delta \boldsymbol{p}_t, \Delta \boldsymbol{p}_{t+1}, ..., \Delta \boldsymbol{p}_{t+L-1})}{\sum_{j=t}^{t+L-1} ||\Delta \boldsymbol{p}_j||} \tag{4.2}$$

4.4 3D Trajectories

To extend the defined above 2DSTD descriptor we add the corresponding depth value to each detected 2D trajectory point. Let's define $\boldsymbol{p}_t^*$, the 3D trajectory point as:

$$\boldsymbol{p}_t^* = (x_t^{cam}, y_t^{cam}, z^*), \tag{4.3}$$

where x_t^{cam} and y_t^{cam} are coordinates of detected POI in camera coordinate system. The method to obtain camera coordinates is described in fig. 4.2. The z^* is a depth value measured at point x_t^{cam}, y_t^{cam} obtained from filtered depth-map $\boldsymbol{V'}$. The filtering method is described in section 4.4.2.

Please note that adding depth information to 2D trajectory point is not a straightforward operation due to three problems: measured depth value and position of 2D trajectory point are in different coordinates system and in different units. Secondly measured depth-value contain significant amount of noise and finally depth value is not measured for all pixels. In sections below we propose solutions to the mentioned problems.

4.4.1 Mapping to a common coordinate system

It is important to notice that points positions obtained from RGB are expressed in pixels *w.r.t.* top-left corner of the screen, which is the center of the image coordinate system. The depth-map obtained from RBG-D sensor contains the distance from the sensor measured usually in millimeters, thus in this case center of coordinate system is the camera (see fig. 4.2). Our goal is to unify 2D points (x_{im}, y_{im}) expressed in image coordinates with z expressed in camera coordinates, as well as unify the units in which they are expressed.

The relationship between camera coordinates and image coordinates can be defined with the pinhole camera model:

$$\begin{bmatrix} x_{im} \\ y_{im} \\ 1 \end{bmatrix} = \boldsymbol{I} * \begin{bmatrix} x_{cam} \\ y_{cam} \\ z \end{bmatrix} \tag{4.4}$$

where x_{cam}, y_{cam} are 3D point coordinates in camera coordinate system, z is measured depth value (already in camera coordinate system), (x_{im}, y_{im}) are 2D point coordinates in

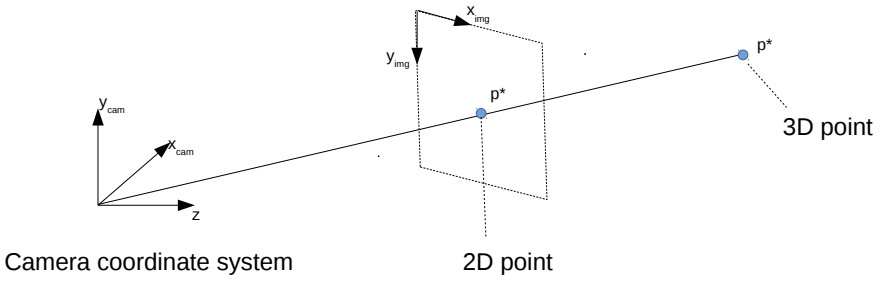


Figure 4.2: This figure shows graphically, the relation between 2D point $\boldsymbol{p}$ obtained from RGB detector that is defined in an image coordinate system and $\boldsymbol{p}^*$ 3D point defined in camera coordinate system.

image coordinate system and I is a intrinsic matrix defined as follows:

$$\boldsymbol{I} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

where f_x, f_y are the focal lengths expressed in pixels $f_x = f * m_x$. The f is physical lens focal length and m_x is the term that relates pixel to distance. The c_x, c_y parameters represent principal point. The mentioned parameters are characteristic to the sensor and can be obtained from the documentation or estimated via camera calibration.

If we solve eq. (4.4) we can find mapping between image coordinates and camera coordinates:

$$x_{img} = \frac{x_{cam} * f_x}{z} + c_x \tag{4.6}$$

$$y_{img} = \frac{y_{cam} * f_y}{z} + c_y \tag{4.7}$$

but we are interested in x_{cam} and y_{cam}, which can be found by simple transformation of

eq. (4.6) and eq. (4.7):

$$x_{cam} = \frac{(x_{img} - c_x) * z}{f_x} \tag{4.8}$$

$$y_{cam} = \frac{(y_{img} - c_y) * z}{f_y} \tag{4.9}$$

please note that in this case in one step we transform 2D point to camera coordinate system and we do the units unification (now all coordinates are expressed in millimeters).

4.4.2 Depth-map filtering

The depth-map obtained from RGB-D sensor contains noise which appears as sharp changes of depth values in places where surface of actual object is smooth. In addition, measured depth value of a object which did not move may differ over time. To obtain more stable depth-map we propose to apply median filter. Let's define $\boldsymbol{V}$ as a depth-map obtained from RGB-D sensor and $\boldsymbol{M}$ as a median filter kernel. Then filtered depth-map $\boldsymbol{V}'$ can be defined as convolution of original depth-map $\boldsymbol{V}$ and median filter kernel $\boldsymbol{M}$:

$$\boldsymbol{V}' = \boldsymbol{M} * \boldsymbol{V} \tag{4.10}$$

4.4.3 Handling missing depth values

In addition to problem with noise in a depth values measurement (section 4.4.2), the depth-map obtained from RGB sensor typically contains areas where depth values were not measured. For instance black materials absorb infrared beam, which cause missing depth values. Also objects or people cover part of infrared beam, causing "shadow" effect of missing depth values. Unfortunately it happens at the border of a posture, thus many trajectory points detected around hands or arms suffer from missing depth values. This fact has impact on 3D trajectories modeling. If depth value z^* is missing in trajectory point, then we cannot neither convert this point to camera coordinates (section 4.4.1) nor compute descriptor (section 4.5), thus such trajectory has to be removed. The more trajectories are remove the less features are available to describe the action, which may affect the final recognition accuracy.

To overcome the above problem, we propose to interpolate missing depth values, by fitting polynomial curve to trajectory points with measured depth value. Let's assume that we have trajectory: $(\boldsymbol{p}_t^*, t = 1, 2, ..., T)$, where $\boldsymbol{p}_t^* = (x_t, y_t, z_t^*)$, with n missing z_t^* depth values. Let's also define $\boldsymbol{Z}$ as a subset of z_t^* that contain valid depth values. To interpolate

the missing values we propose to find polynomial p of degree k such that:

$$p(t) = z_t^*, z_t^* \in \boldsymbol{Z} \tag{4.11}$$

Please note that proposed interpolation schema exploits fact that we construct trajectory, which contains points tracked over the time. The rational of polynomial interpolation is that trajectory point with missing depth value, should be located in some neighborhood of trajectory points observed before and after. Alternatively we could extend a size of median filter proposed in section 4.4.2, but too big filter size may cause that we will capture depth values from the background. Thus we claim that interpolation based on trajectory shape is more robust.

4.5 3D Trajectory Shape Descriptor (3DTSD)

The 3D Trajectory Shape Descriptor (3DTSD) is computed in a similar way as 2DTSD in section 4.3. The difference is that 2D trajectory point $\boldsymbol{p_t}$ is now a 3D point, thus $\Delta \boldsymbol{p_t^*}$ is 3D displacement vector, which forms the following 3D descriptor $\boldsymbol{s^*} = (\Delta \boldsymbol{p}_t^*, \Delta \boldsymbol{p}_{t+1}^*, ..., \Delta \boldsymbol{p}_{t+L-1}^*)$. Similarly the descriptor is normalized by sum of the magnitudes of the displacement vectors.

In section 4.4.3 we proposed method which is able to recover missing depth values for the detected POIs. However if the trajectory contains too many missing values, our method is not able to recover them. In such case trajectories which contain at least one point with missing depth value are discarded.

4.6 Experiments

In this section we will report the performance of the proposed 3DTSD and 3DRTD descriptors on 3 datasets:

- CAD-60
- CAD-120
- MSRDailyActivity3D

Since we use Fisher Vector encoding we provide ablation analysis for different codebook sizes: 16, 32, 64, 128, 256, 512. In the experiments we report the performance of each TSD (Trajectory Shape Descriptor) and 3DTSD (3D Trajectory shape descriptor) descriptor separately. We also compare Dense Trajectories descriptor (DT) which is a fusion of TSD, MBH, HOG, HOF descriptors with DT;3DTSD descriptor where TSD was replaced

with 3DTSD. For CAD-60 dataset we provide detailed analysis (section 4.6.1.1) how depth map filtering and recovery of missing depth values affect the 3DTSD descriptor and further action recognition accuracy. We decided not to evaluate the described method on Smarthomes dataset, because as described in appendix A, the depth-map contains much amount of noise, thus 3DTSD in the form proposed in this chapter is not suitable for such situation.

4.6.1 CAD-60

The details of CAD-60 dataset are presented in section 2.3.1. This section consists of two parts: in first part we provide analysis of number of missing depth values and it's influence on recognition accuracy. We show how recovery method proposed in section 4.4.3 improves recognition accuracy. In second part we compare proposed 3DTSD descriptor with TSD, then we compare DT descriptor (which is a fusion of TSD and MBH, HOG, HOF), with DT;3DTSD where we replace TSD with 3DTSD.

In this section we use the following abbreviations:

- TSD – Trajectory Shape Descriptor,
- 3DTSD – 3D Trajectory Shape Descriptor,
- DT – Dense Trajectories *i.e.* fusion of TSD, HOG, HOF, MBH,
- DT;3DTSD – fusion of 3DTSD, HOG, HOG, MBH,
- 3DTSD w/o depth recovery – 3D Trajectory Shape Descriptor where we removed all trajectories which contains at least one missing depth-value (we did not apply any method which recovers missing depth value),
- TSD trimmed – Trajectory Shape Descriptor where we removed same trajectories, as in "3DTSD w/o depth recovery"

4.6.1.1 Impact of Missing Depth Values Analysis

Figure 4.3 shows fraction of trajectories in CAD-60 dataset *w.r.t.* number of missing depth values in trajectory. In our case trajectory length $L = 15$. We can see from fig. 4.3 that for 60% of trajectories all their points have valid depth value. Thus means, that if we want to compute 3DTSD descriptor without trying to recover missing depth values we would have 40% less features available. Our experiments show it has serious impact on action recognition accuracy (table 4.1).

First let's construct "TSD trimmed" descriptor which contains only 60% of trajectories comparing to TSD. We keep only trajectories that have valid depth value for all trajectory

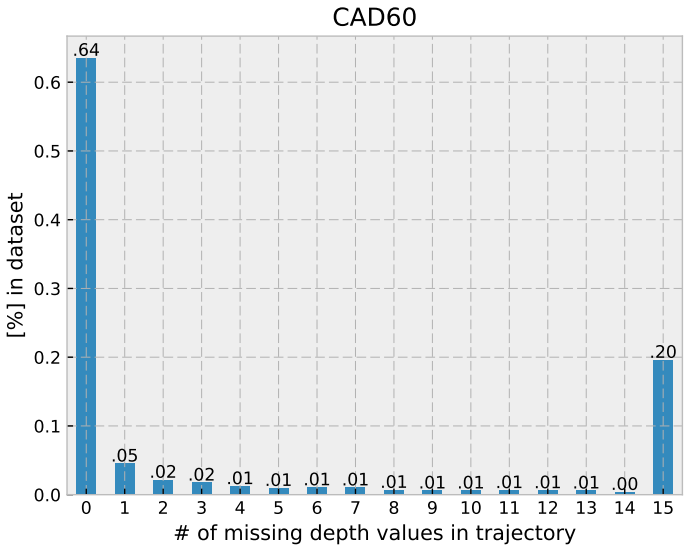


Figure 4.3: Distribution of trajectories *w.r.t.* number of missing depth values on CAD-60 dataset. Only trajectories with 0 missing values can be used to compute 3DTSD. Recovery of missing depth values from trajectories extends number of trajectories that can be used in 3DTSD descriptor construction. The more trajectories are used in 3DTSD generation the more information is encoded.

points. In table 4.1 we show that "TSD trimmed" descriptor achieves worse accuracy than TSD (59.82% vs 63.18%). Because "TSD trimmed" contains 60% of TSD features, it cannot outperform TSD descriptor which has more information.

The next interesting result is that "3DTSD w/o depth recovery" outperforms "TSD trimmed" descriptor, but performs worse than TSD descriptor. This result show that 3D descriptor outperforms 2D descriptor if they have same amount of features. Then if we apply depth values recovery method proposed in section 4.4.3 we show that 3DTSD descriptor outperforms TSD.

4.6.1.2 Comparison with TSD and DT

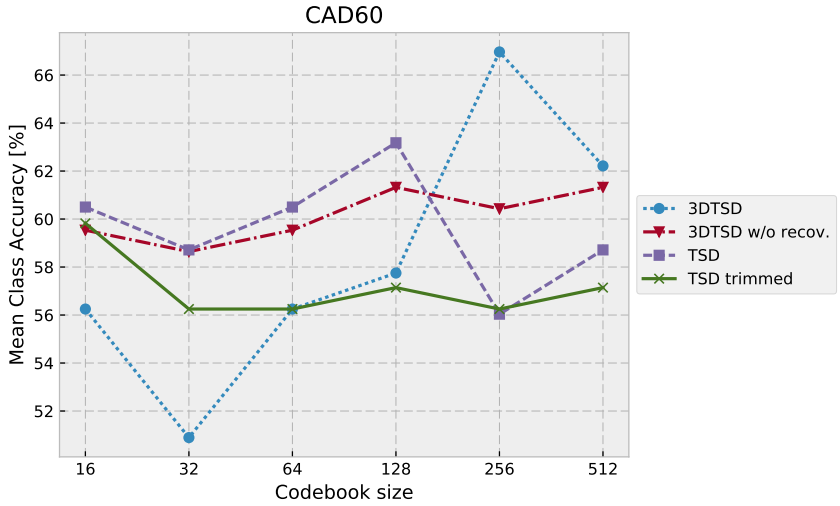
The results are presented in table 4.2, in this section we focus on comparison of 3DTSD vs. TSD descriptor and their fusion with DT descriptor.

- 3D Trajectory Shape Descriptor (3DTSD) outperforms 2D Trajectory Shape Descriptor (TSD).
- TSD performs better on smaller codebook sizes comparing to 3DTSD. This can be explained by the fact that 3D information encoded in 3DTSD introduces bigger vari-

ance in data, thus bigger codebook is need.

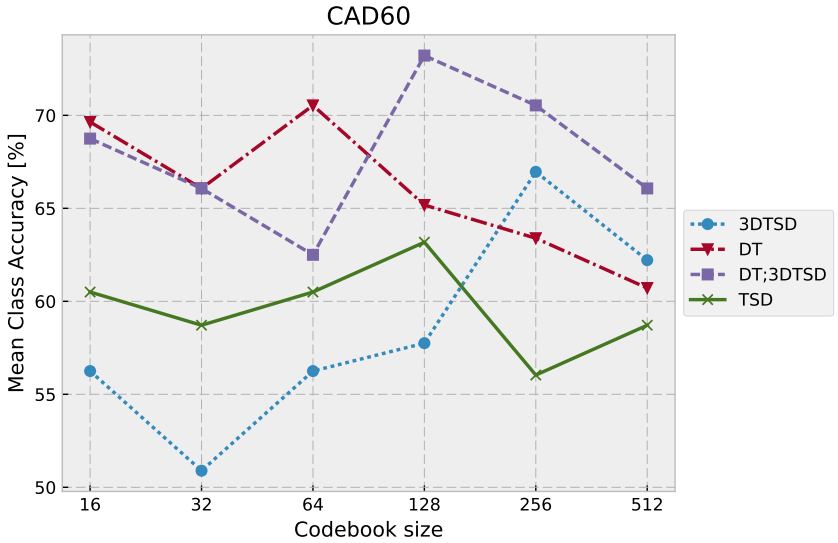
- DT;3DTSD descriptor outperforms DT descriptor. The fusion of 3DTSD with DT descriptor leads to further recognition accuracy improvement.

Table 4.1: Results on CAD-60 dataset. The results show that "3DTSD w/o recovery" outperforms "TSD trimmed" descriptor. The "TSD trimmed" descriptor was obtained by removing trajectories which contain at least one missing depth-value. Thus "3DTSD w/o recovery" and "TSD trimmed" contain same number of trajectories. The results suggest that "3DTSD w/o recovery" could outperform TSD if it contained same number of trajectories. That is why our 3DTSD descriptor with proposed missing depth values recovery is able to outperform TSD.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
3DTSD	56.25	1.26	50.89	1.26	56.25	3.79	57.75	5.05	66.96	2.53	62.21	2.53	66.96	2.53
TSD	60.50	2.53	58.71	5.05	60.50	0.00	63.18	1.26	56.04	3.79	58.71	0.00	63.18	1.26
3DTSD w/o recov.	59.54	1.26	58.64	3.79	59.54	3.79	61.32	3.79	60.43	2.53	61.32	3.79	61.32	3.79
TSD trimmed	59.82	1.26	56.25	1.26	56.25	3.79	57.14	0.00	56.25	3.79	57.14	2.53	59.82	1.26

Table 4.2: Results on CAD-60 dataset, the plot shows Mean Class Accuracy *w.r.t.* codebook size. The result show that proposed 3DTSD descriptor outperforms TSD descriptor standalone, as well as fused together with DT descriptors.



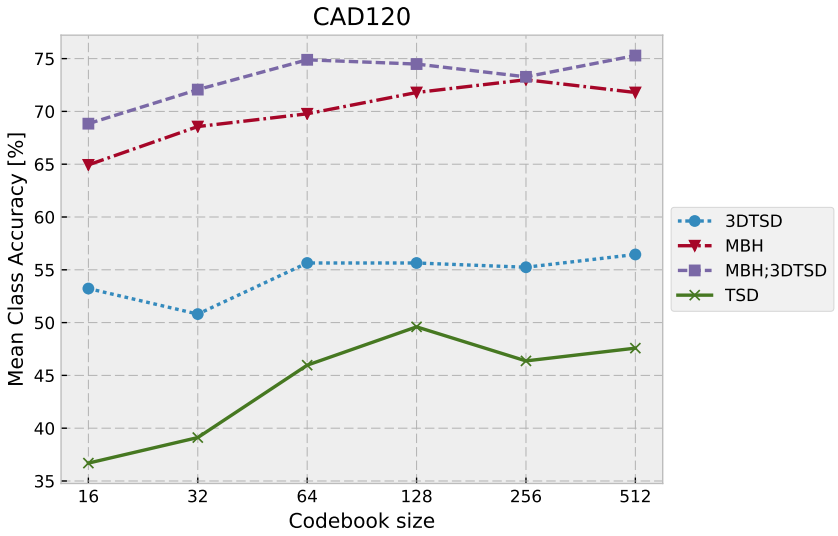
	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
DT;3DTSD	68.75	1.26	66.07	0.00	62.50	2.53	73.21	0.00	70.54	6.31	66.07	2.53	73.21	0.00
DT	69.64	2.53	66.07	2.53	70.54	1.26	65.18	1.26	63.39	1.26	60.71	2.53	70.54	1.26
3DTSD	56.25	1.26	50.89	1.26	56.25	3.79	57.75	5.05	66.96	2.53	62.21	2.53	66.96	2.53
TSD	60.50	2.53	58.71	5.05	60.50	0.00	63.18	1.26	56.04	3.79	58.71	0.00	63.18	1.26

4.6.2 CAD-120

The details of CAD-120 dataset are presented in section 2.3.2. The results are presented in table 4.3, in case of CAD-120 dataset we present only performance of 3DTSD (with missing values recovery), and it's fusion with MBH descriptors. It is worth noting that in case of CAD-120 dataset analysis and fusions are done with MBH dataset, because it was best performing baseline descriptor.

- 3D Trajectory Shape Descriptor (3DTSD) outperforms 2D Trajectory Shape Descriptor (TSD).
- 3DTSD performs better than TSD on all codebook sizes.
- DT;3DTSD performs better than DT on all codebook sizes, the fusion of 3DTSD and DT leads to further accuracy improvement.
- 3DTSD and DT;3DTSD perform better with bigger codebooks. We believe that this is connected to the fact that CAD-120 introduces frontal and lateral view, which increase variance in 3DTSD descriptor.
- 3DTSD reports much higher improvement over TSD descriptor on CAD-120, than on CAD-60 and MSRDailyActivity3D. The reason behind it, is that CAD-120 introduces frontal and lateral viewpoint. TSD descriptor is not invariant to such viewpoint changes, while 3DTSD models trajectory shape in camera coordinates and handle this situation much better.
- Thanks to relatively big improvement of 3DTSD, also fusion with MBH descriptor is working and leading to performance improvement.

Table 4.3: Results on CAD-120 dataset, the plot shows Mean Class Accuracy *w.r.t.* codebook size. The result show that proposed 3DTSD descriptor outperforms TSD descriptor standalone, as well as fused together with DT descriptors.



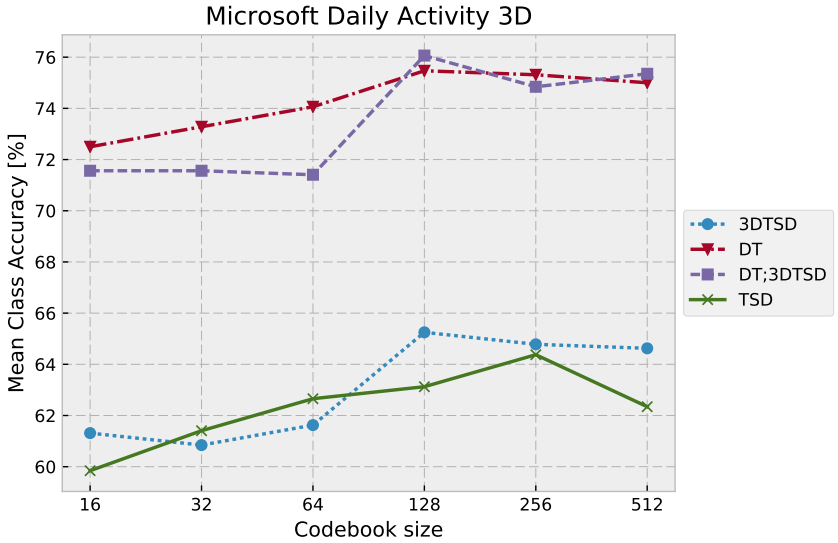
	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
MBH;3DTSD	68.84	1.14	72.06	0.57	74.89	3.42	74.48	0.57	73.27	1.14	75.29	1.27	75.29	1.27
MBH	64.94	1.14	68.56	1.71	69.77	0.00	71.79	2.85	73.00	1.14	71.79	3.99	73.00	1.14
3DTSD	53.23	2.85	50.81	1.71	55.65	0.00	55.65	4.56	55.24	1.14	56.45	2.53	56.45	2.53
TSD	36.69	0.57	39.11	1.71	45.97	1.14	49.60	1.71	46.37	0.57	47.58	1.14	49.60	1.71

4.6.3 MSRDailyActivity3D

The details of MSRDailyActivity3D dataset are presented in section 2.3.3. The results are presented in table 4.4. We summarize the results with the following conclusions:

- 3D Trajectory Shape Descriptor (3DTSD) outperforms TSD descriptor by small margin.
- DT;3DTSD descriptor outperforms DT descriptor, but the improvement is much smaller then in previously presented datasets.
- The reason of relatively small improvement of 3DTSD is that the authors of MSRDailyActivity3D dataset did not provide calibration parameters. In such case we estimated the calibration parameters from the dataset, but our estimations might have introduced some noise and this might be a reason of relatively small performance improvement.

Table 4.4: Results on MSRDailyActivity3D dataset, the plot shows Mean Class Accuracy *w.r.t.* codebook size. The result show that proposed 3DTSD descriptor outperforms TSD descriptor standalone, as well as fused together with DT descriptors.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
DT;3DTSD	71.56	0.44	71.56	0.88	71.41	1.55	76.06	0.66	74.84	1.10	75.35	2.87	76.06	0.66
DT	72.50	0.88	73.28	0.22	74.06	0.88	75.47	0.22	75.31	1.33	75.00	0.00	75.47	0.22
3DTSD	61.31	0.88	60.84	1.77	61.62	0.66	65.25	0.66	64.78	1.10	64.62	1.33	65.25	0.66
TSD	59.84	1.10	61.41	1.55	62.66	1.10	63.12	1.33	64.38	0.00	62.34	1.55	64.38	0.00

4.7 Conclusions

In this chapter we proposed 3DTSD (3D Trajectory Shape Descriptor) which joins advantages of POIs detection on RBG image with 3D information available from depth-map. The proposed descriptor is low dimensional comparing to other state-of-the-art descriptors and requires relatively low computational complexity in further processing, this is an advantage in real-time systems (see chapter 10). Another advantage of the proposed descriptor is that it explicitly models geometrical positions, thus it makes it easy to apply any kind of geometrical transformations of trajectory points position. This is a clear advantage in a multiview system where our descriptor can be easily transformed from one camera view-point to another. In our experiments we show that the proposed method outperforms 2D Trajectory Shape Descriptor, also fusion of 3DTSD with Dense Trajectories descriptor leads to accuracy improvement over standard Dense Trajectories. The limitation of proposed method is that it requires good quality of measured depth-map, as too much noise or too many missing values can affect the performance.

Chapter 5

Removing hallucinations by histogram normalization

Contents

5.1 Introduction

Histogram of Oriented Gradients (HOG) is one of the most popular descriptors for characterizing image regions [69, 112]. It has been successfully applied to various vision tasks such as localization, classification and recognition. It shows especially good performance while describing human appearance [20, 29]. As HOG mainly captures gradient strengths in an image *w.r.t.* their orientation, it is sensitive to variations in illumination and contrast. Consequently, a normalization of HOG turns out to be essential for obtaining good

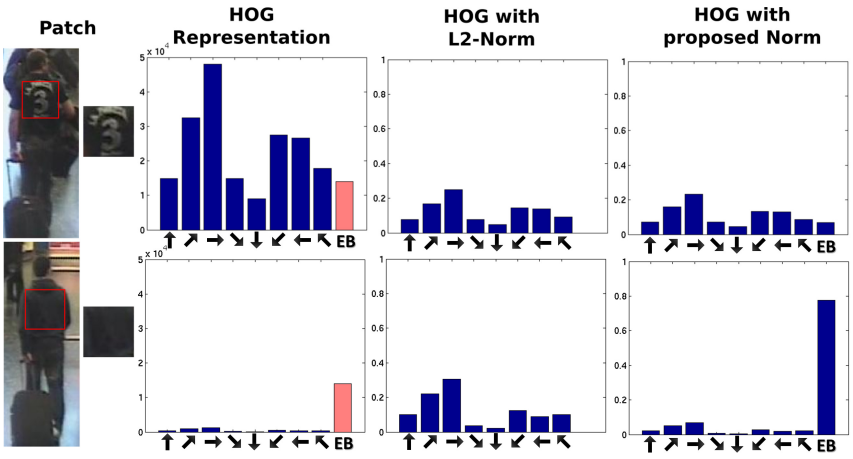


Figure 5.1: HOG representations of two patches with different amount of texture. Each bin in blue represents the sum of magnitudes of edges for a particular orientation. The bin marked as EB represents the proposed extra bin.

performance. In [20], authors cover the importance of the normalization step, illustrating that without proper normalization scheme the performance of their human detection framework drops significantly.

Although many different normalization schemes have been investigated, the core of all of them is L1 or L2-norm. In this chapter we show that an incautious application of L-like norms to the HOG descriptor might produce a hallucination effect, *i.e.* specific properties of high texture can appear while describing a texture-less image. As an example see fig. 5.1, it illustrates histogram representations of two significantly different image regions; high texture (top row) and low texture (bottom row). From the second column (HOG REPRESENTATION) it is clear that there is a big difference between HOG representations (in blue color) because the number of edges in the texture-less patch is at least one order of magnitude lower. However, after L2-normalization, this difference disappears making both image regions indistinguishable. We call this undesired effect as *hallucination*, because some elements (in this case edges) appear in the HOG representation while they do not actually exist in the original image. As a result, two image regions can have similar final HOG representation while being actually very different. This might decrease the accuracy in any recognition system based on the HOG descriptor.

To overcome this issue, we propose to add an extra bin (EB) with a specific value to the HOG histogram before the normalization step. We claim that this helps to differentiate between patches that contain texture from the ones that do not. EB is illustrated in red in fig. 5.1. Note that the histograms in the last column maintain the difference after the L2 normalization. The representation for the textured patch did not change considerably while in the non-textured patch, the extra bin went significantly higher compared to the

rest of the bins.

Normalization with EB adds two types of information for differentiating between textured and non-textured patches: (1) it helps to maintain the scale of bins when their value is low; (2) after the normalization EB contains information on the scale of the histogram, being higher for non-textured patches and lower for textured ones. We discovered that the value of EB can have a significant impact on the performance *w.r.t.* the recognition task.

We perform a detailed analysis of EB *w.r.t.* its value and the descriptor performance. We offer a learning scheme of the EB value. Our learning process is based on modeling an integral of gradient as a Mixture of two Gaussian distributions. We found that selecting value of EB that corresponds to the intersection of these distributions, leads to recognition accuracy improvement over standard HOG and it is close to optimal value found via cross-validation.

The issue related to the hallucination effect has recently been mentioned in OTC descriptor [66] for scene categorization problem. Authors proposed to add a fixed value bin (0.05) to the histogram before applying *L2-norm*. We claim that the value of additional histogram bin has impact on the performance. In addition claim that additional bin should be kept in the final histogram representation, as it contains information on the scale of the histogram.

Our normalization schema can be applied to any histogram based descriptor. We apply it on HOG, because HOG models appearance which helps to distinguish actions with similar motion patterns, but different objects involved. This is frequent in daily-living action recognition, which is an area of our interest.

5.2 Approach Overview

5.2.1 Histogram of Oriented Gradients

In HOG descriptor each bin represents the magnitude of edges oriented in specific direction. Using HOG with absolute magnitude values is not practical, as similar patterns with different gradient magnitude will result in different histogram representation. Histogram normalization unifies the value range for each representation. This allows HOG comparison and improves training process when HOG is used with machine learning algorithms.

As mentioned in section 5.1 – application of *L2-norm* can result in *hallucination* problem. To overcome this issue we propose to add an extra bin (EB) with a specific value. In the following sections we explain how (EB) value is selected.

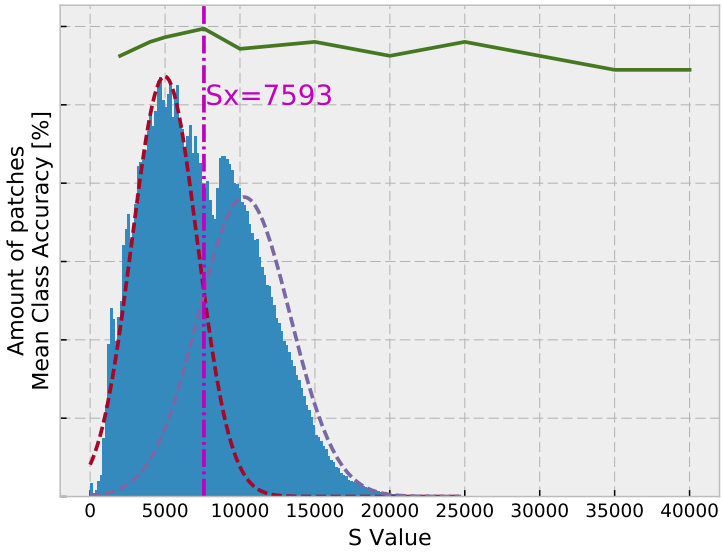


Figure 5.2: In blue: histogram of S (sum of bins) for HOG descriptors on CAD-60 dataset. Green full curve shows the recognition rate *w.r.t.* to different EB. Red and purple dash curves illustrate two Gaussian distributions obtained by employing Expectation Maximization. Peak of performance can be seen for EB $= s_x$, where two distributions intersect.

5.2.2 Distribution of the sum of bins

To find the (EB) value we investigate the distribution of aggregated HOG magnitudes. In the first step we calculate HOG representation for considerable number of random patches from training set. Then for each HOG descriptor we sum up the magnitudes across all bins. Finally we form the histogram which describes the frequency of HOG magnitudes in selected dataset.

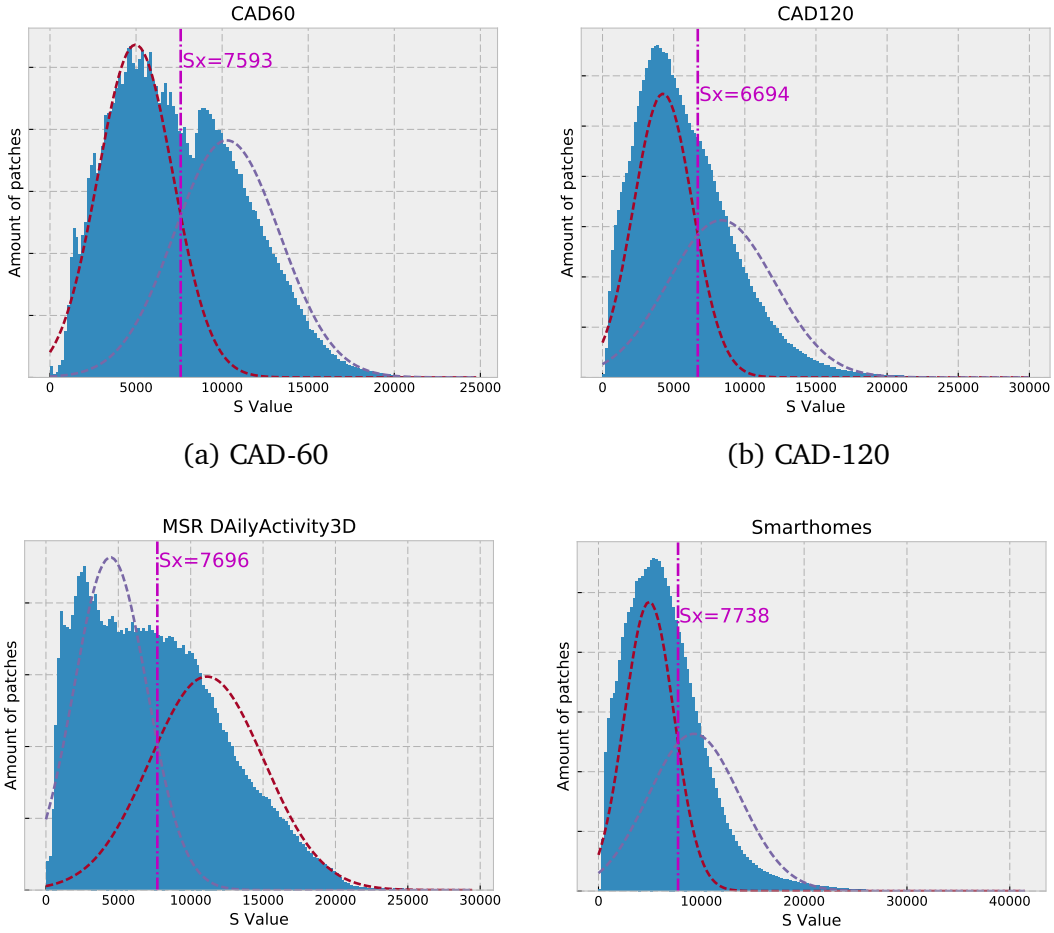
In other words, having bin b_i that represents a particular orientation in a HOG descriptor and number of bins n for each histogram, we express the sum of all bins of a particular patch j as:

$$s_j = \sum_{i=1}^{n} b_i \tag{5.1}$$

Thus we define distribution $S = p(s)$ of s_j values *w.r.t.* their frequency. The fig. 5.2 shows the distribution S for CAD-60 dataset.

5.2.3 Expectation Maximization

Figure 5.2 shows that distribution S is a mixture of several other distributions. In particular, we would like to find a distribution which models texture-less patches and a second



(a) CAD-60 (b) CAD-120

(c) MSRDailyActivity3D (d) Smarthomes

Figure 5.3: Distribution of the sum of bins for randomly sampled patches for each dataset. Modeled distributions are drawn in dashed lines.

distribution which models patches with high amount of texture. Thus, we model S as a mixture of two Gaussian distributions:

$$S = \mathcal{N}_1(\mu_1, \sigma_1) + \mathcal{N}_2(\mu_2, \sigma_2). \tag{5.2}$$

We apply Expectation Maximization (EM) algorithm to find the parameters $(\mu_1, \mu_2, \sigma_1, \sigma_2)$ of $\mathcal{N}_1$ and $\mathcal{N}_2$ that stand for a Gaussian Mixture Model (GMM). In fig. 5.2 – the dashed lines corresponds to Gaussians found by EM algorithm.

If we analyze different EB values and plot corresponding accuracy score (green line in fig. 5.2), we can find the EB $= s_x$ value which leads to best accuracy score on validation set. Surprisingly s_x corresponds to the value where both Gaussians are intersecting, that

is:

$$s_x = \arg \ \big[\mathcal{N}_1(s) = \mathcal{N}_2(s)\big]. \tag{5.3}$$

Based on that we claim that selecting EB value based on intersection of Gaussians is a good strategy and can replace the cross validation schema. Our experiments show that EB based on intersection of Gaussian, outperforms standard HOG representation and in most cases is very close to best accuracy score based on EB obtained via cross-validation. We would like to highlight that in described algorithm we assume that patches in given dataset can be divided into two groups: texture-less and with high amount of texture (for instance fig. 5.2). This is not necessarily, the case in all datasets (fig. 5.3), but described search method via expectation maximization is still able to find good EB values.

5.2.4 Algorithm

Finally once EB value is found. We add the EB value to HOG descriptor for all patches in the dataset. Then, we normalize descriptors using L2-norm and continue with the normal work flow *w.r.t.* given application. As a summary, we present the steps of our proposed algorithm:

Algorithm 1 Proposed algorithm

1: Extract HOG descriptor from randomly sampled patches from the dataset.
2: Generate a distribution of S where $s_j = \sum b_i$.
3: Fit two Gaussian models using EM.
4: Find the intersection s_x of Gaussian curves.
5: Add EB $= s_x$ for each HOG descriptor in the dataset,
6: Normalize descriptors with a L-like norm and continue with the normal work flow.

5.3 Implementation details

Many authors [116, 52] combine appearance descriptors with motion-based descriptors. We also combine proposed method with Dense Trajectories motion features (trajectory shape, HOF, MBH).

For evaluation we extract local spatio-temporal patches from a video sequence. For this task we use Dense Trajectories detector, but it is worth mentioning that any other detector can be used. We follow the same procedure as in [113] and we cropp the patch around each trajectory point. Then for each patch we compute HOG descriptor with standard normalization and with proposed extra bin. Finally we follow standard Fisher Vectors [82] framework, where we encode a video sequence using first and second order statistics of a

distribution of a descriptor feature set $\mathbb{X}$. The details of the action recognition framework are described in chapter 3.

5.4 Experiments

In this section we will report the performance of the proposed normalization method on 4 datasets (the detailed description of datasets is available in section 2.3):

- CAD-60
- CAD-120
- MSRDailyActivity3D
- Smarthomes

We report performance of proposed method with different EB values compared to standard HOG descriptor obtained from Dense Trajectories. Since we use Fisher Vector encoding we provide ablation analysis for different codebook sizes: 16, 32, 64, 128, 256, 512. We also do an extensive search of EB value – we try 8 different EB values: 5000, 10000, 15000, 20000, 25000, 30000, 35000, 40000. Then we compare accuracy of descriptor normalized with EB$= s_x$ value found with proposed expectation maximization algorithm (see section 5.2.4) with descriptor normalized with EB value found through extensive search. Our experiments show that adding extra bin to HOG descriptor improves accuracy. Moreover the proposed algorithm finds EB value which performs as good as EB value found through extensive search.

Finally we compare Dense Trajectories descriptor (DT) with standard HOG with DT;HOGEB descriptor where standard HOG descriptor was replaced with HOG normalized with EB$= s_x$ value.

In this section we use the following abbreviations:

- HOG – Standard HOG descriptor,
- HOG(EB=x) – HOG descriptor normalized with EB value equals to x,
- HOG(EB$=s_x$) – HOG descriptor normalized with EB with proposed algorithm (see section 5.2.4),
- DT – Dense Trajectories descriptor *i.e.* fusion of TSD, HOG, HOF, MBH,
- DT;HOGEB – Dense Trajectories descriptor with HOG(EB$=s_x$) in place of standard HOG descriptor.

5.4.1 CAD-60

Let's start with analysis of influence of EB value the fig. 5.4 and table 5.1 show the performance of HOG descriptor *w.r.t.* to different EB value (for each EB value best performing codebook size was selected). In addition we report also performance of standard HOG descriptor and one normalized with EB$=s_x$ value which in case of CAD60 dataset equals to 7593 and was obtained with algorithm described in section 5.2.4. The results in fig. 5.4 and table 5.1 show that:

- proposed normalization schema improves HOG performance for all selected EB values,
- HOG(EB$=s_x$) descriptor with EB value found with proposed expectation maximization algorithm (see section 5.2.4) outperforms standard HOG descriptor and all HOG descriptors normalized with EB values found with extensive search.

For CAD-60 dataset we compared HOG(EB$=s_x$) with HOG histogram which was normalized with EB$=s_x$, but afterwards the EB$=s_x$ was removed from final histogram. In such case we noticed over 3% drop of accuracy rate comparing to HOG(EB$=s_x$). This result shows that EB$=s_x$ provides useful information about a scale of the histogram.

In table 5.2 we compare HOG(EB$=$5000) – best performing descriptor with EB value found via extensive search, HOG(EB$=s_x$) and standard HOG. The results show that:

- HOG descriptor is consistently outperformed by both: HOG(EB$=$5000) and HOG(EB$=s_x$) *w.r.t.* all codebook sizes,
- HOG(EB$=s_x$) outperforms standard HOG.
- HOG(EB$=s_x$) outperforms HOG(EB$=$5000) by small margin. If we consider standard deviation of the mentioned results the mentioned descriptors achieve comparable results. This shows that the algorithm proposed in section 5.2.4 found as good value as extensive search.

Finally in table 5.3 we show the results of DT;HOGEB descriptor which is fusion of Dense Trajectories descriptors (TSD, HOG, HOF, MBH), where HOG was replaced with HOG(EB$=s_x$). The results shows that further fusion of HOG(EB$=s_x$) descriptor improves performance of standard DT descriptor.

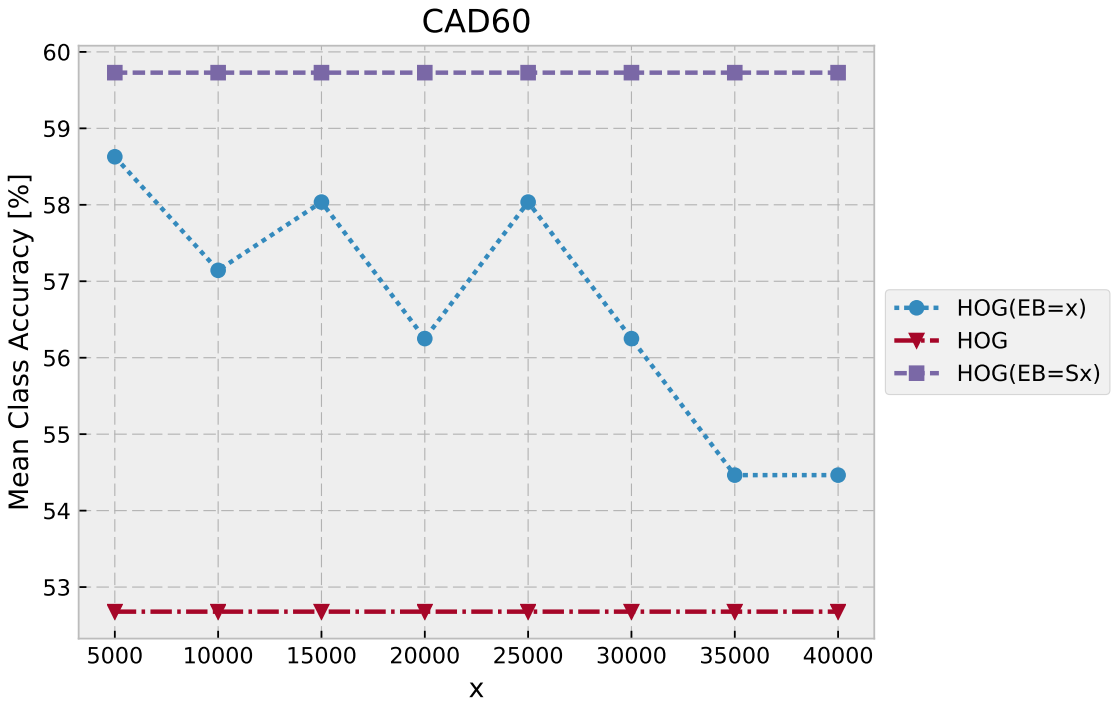
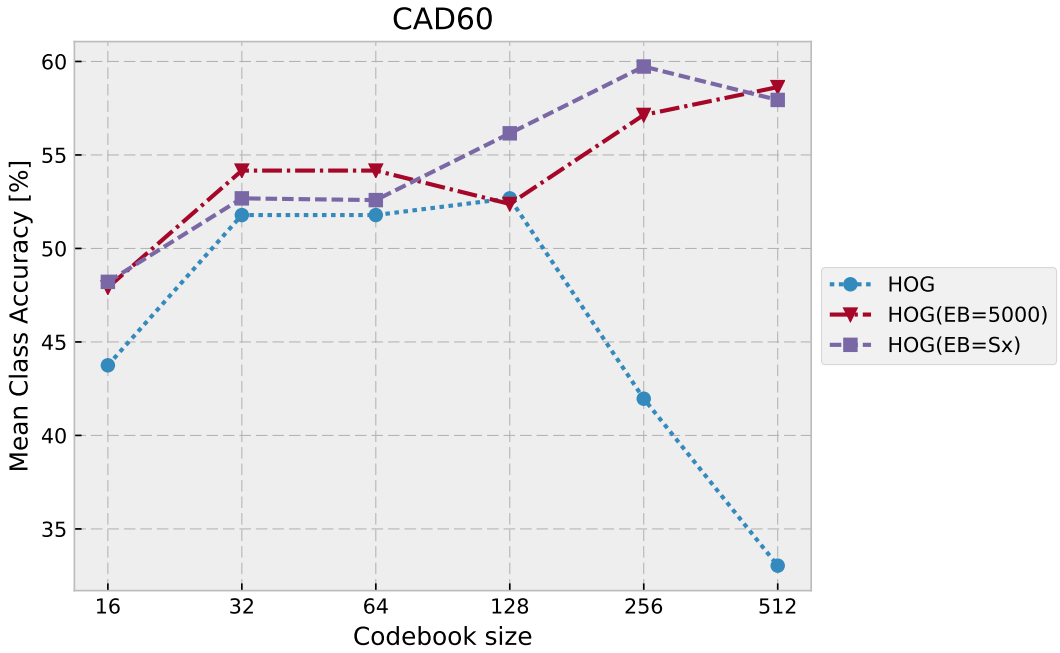


Figure 5.4: Results on CAD-60 dataset. The plot shows Mean Class Accuracy for HOG normalized with different EB values (blue dotted line). The standard HOG performance is showed in red, while EB=s_x is shown in purple. The figure shows that HOG all tested EB values as well as EB=s_x outperform standard HOG descriptor. The EB=s_x for CAD-60 is equal to 7593.

	Acc [%]	σ
HOG(EB=Sx)	59.73	2.53
HOG(EB=5000)	58.63	3.79
HOG(EB=15000)	58.04	3.79
HOG(EB=25000)	58.04	2.53
HOG(EB=10000)	57.14	2.53
HOG(EB=20000)	56.25	1.26
HOG(EB=30000)	56.25	2.53
HOG(EB=35000)	54.46	5.05
HOG(EB=40000)	54.46	2.53
HOG	52.68	3.79

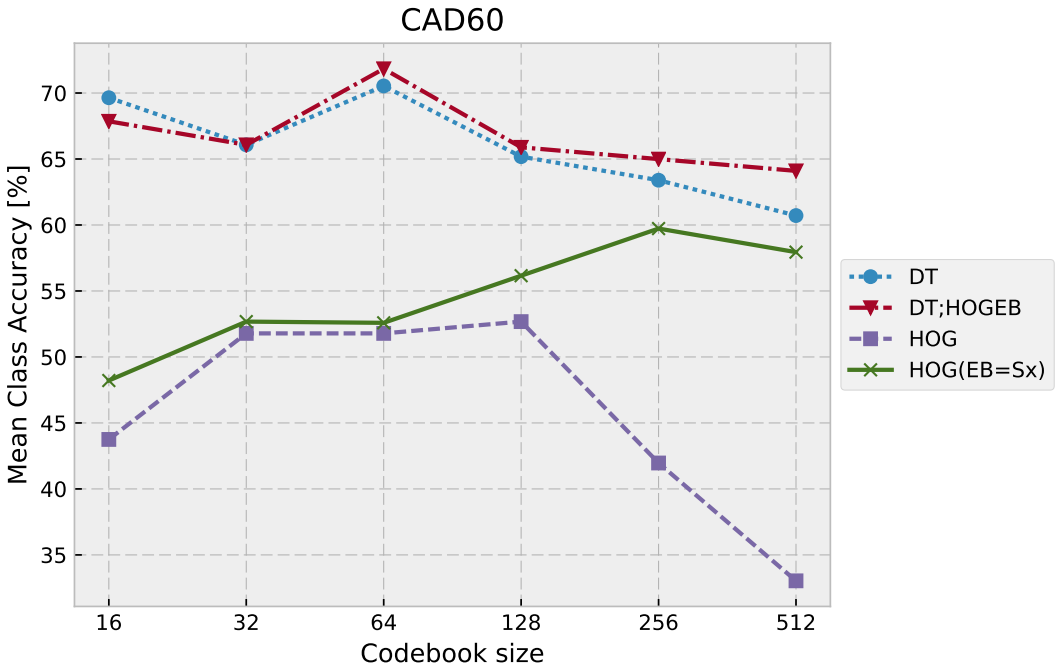
Table 5.1: Results on CAD-60 dataset. The table shows Mean Class Accuracy for HOG normalized with different EB values, standard HOG performance and EB=s_x. The table shows that HOG all tested EB values as well as EB=s_x outperform standard HOG descriptor. The EB=s_x for CAD-60 is equal to 7593.

Table 5.2: Results on CAD-60 dataset, the plot and table show Mean Class Accuracy *w.r.t.* codebook size of standard HOG, normalized with EB value equals to 5000 (best performing value found via cross-validation) and normalized with EB$=s_x$. The EB$=s_x$ for CAD-60 is equal to 7593.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
HOG(EB=Sx)	48.21	3.79	52.68	3.79	52.59	2.53	56.16	3.79	59.73	2.53	57.94	2.53	59.73	2.53
HOG(EB=5000)	47.91	3.79	54.16	8.84	54.16	6.31	52.38	3.79	57.14	3.79	58.63	3.79	58.63	3.79
HOG	43.75	1.26	51.79	0.00	51.79	0.00	52.68	3.79	41.96	6.31	33.04	1.26	52.68	3.79

Table 5.3: Results on CAD-60 dataset, the plot and table show Mean Class Accuracy *w.r.t.* codebook size of standard HOG, standard DT descriptor, HOG normalized with EB=s_x and DT;HOGEB where standard HOG was replaced with HOG(EB=s_x). The EB=s_x for CAD-60 is equal to 7593.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
DT;HOGEB	67.86	1.26	66.07	0.00	71.84	1.26	65.89	1.26	64.99	1.26	64.10	2.53	71.84	1.26
DT	69.64	2.53	66.07	2.53	70.54	1.26	65.18	1.26	63.39	1.26	60.71	2.53	70.54	1.26
HOG(EB=Sx)	48.21	3.79	52.68	3.79	52.59	2.53	56.16	3.79	59.73	2.53	57.94	2.53	59.73	2.53
HOG	43.75	1.26	51.79	0.00	51.79	0.00	52.68	3.79	41.96	6.31	33.04	1.26	52.68	3.79

5.4.2 CAD-120

We follow similar analysis for CAD-120 dataset. The fig. 5.5 and table 5.4 show the performance of HOG descriptor *w.r.t.* to different EB value (for each EB value best performing codebook size was selected). In addition we report also performance of standard HOG descriptor and one normalized with EB$=s_x$ value which in case of CAD-120 dataset equals to 6690 and was obtained with algorithm described in section 5.2.4. The results in fig. 5.5 and table 5.4 show that:

- proposed normalization schema improves HOG performance for EB values smaller than 20000,
- we can observe (fig. 5.5) trend, where bigger EB values leads to decrease in action recognition accuracy,
- HOG(EB$=s_x$) descriptor with EB value found with proposed expectation maximization algorithm (see section 5.2.4) outperforms standard HOG descriptor. The HOG(EB=5000) where EB value found through extensive search leads to slightly better performance than HOG(EB$=s_x$).

In table 5.5 we compare HOG(EB=5000) – best performing descriptor with EB value found via extensive search, HOG(EB$=s_x$) and standard HOG. The results show that:

- HOG descriptor is consistently outperformed by both: HOG(EB=5000) and HOG(EB$=s_x$) *w.r.t.* all codebook sizes,
- HOG(EB$=s_x$) outperforms standard HOG.
- HOG(EB=5000) outperforms HOG(EB$=s_x$) If we consider standard deviation of the mentioned results the mentioned descriptors achieve comparable results. This shows that the algorithm proposed in section 5.2.4 found as good value as extensive search.

Finally in table 5.6 we show the results of MBH;HOGEB descriptor which is fusion of MBH, with HOG(EB$=s_x$). It is worth noting that in case of CAD-120 dataset analysis and fusions are done with MBH dataset, because it was best performing baseline descriptor. The results shows that further fusion of HOG(EB$=s_x$) descriptor improves performance of standard DT descriptor.

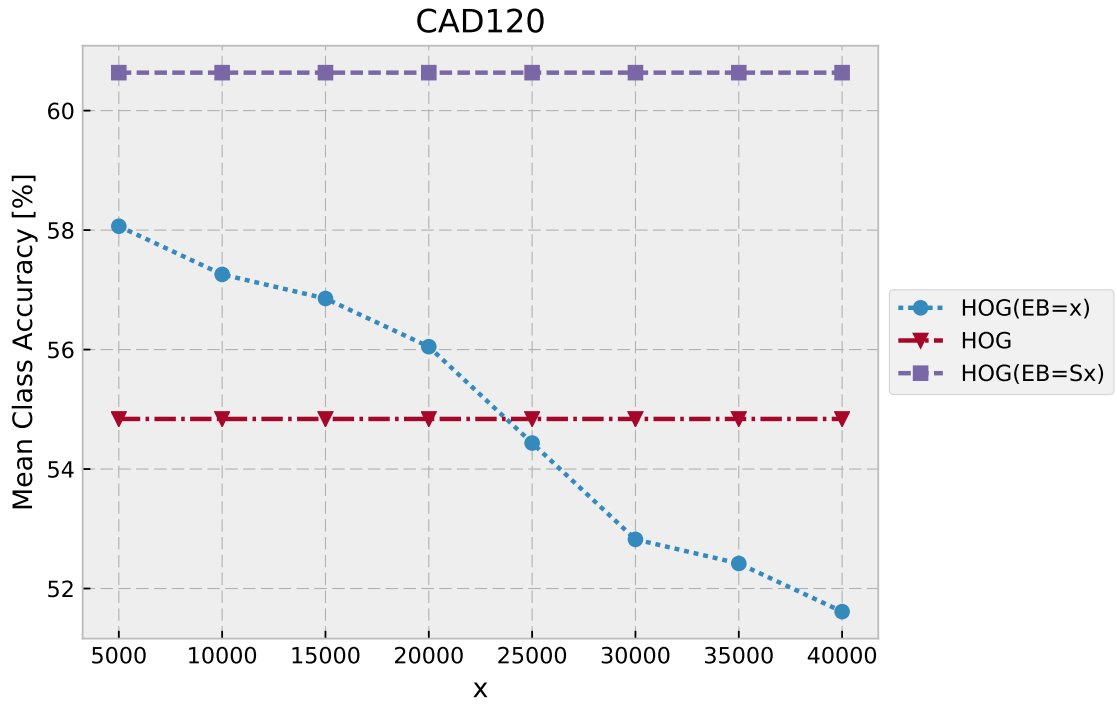
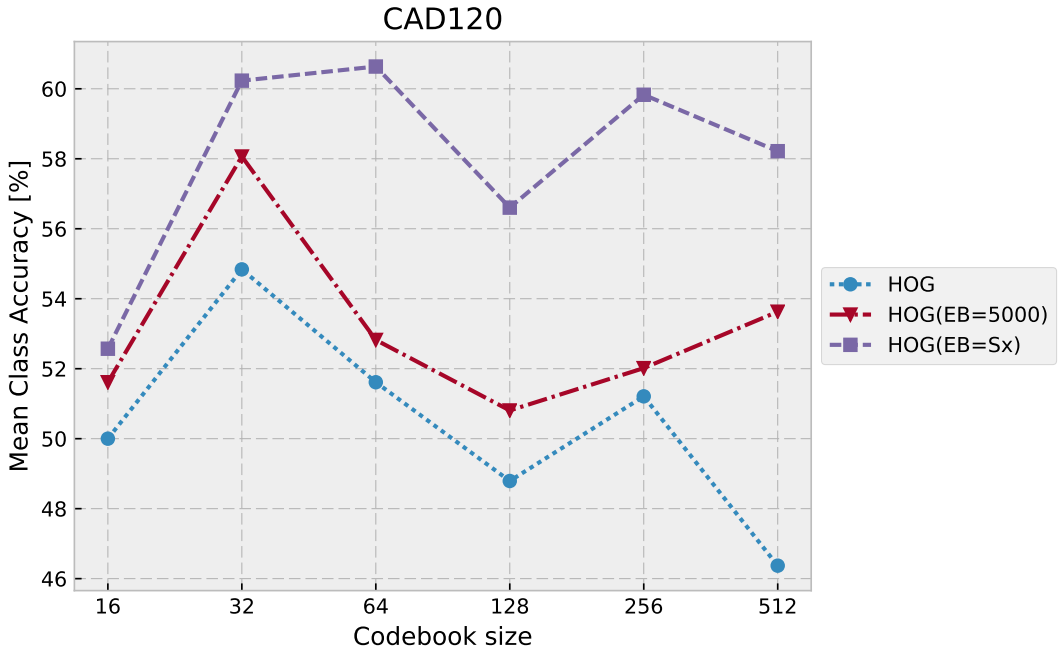


Figure 5.5: Results on CAD-120 dataset. The plot shows Mean Class Accuracy for HOG normalized with different EB values (blue dotted line). The standard HOG performance is showed in red, while EB=s_x is shown in purple. The figure shows that HOG all tested EB values as well as EB=s_x outperform standard HOG descriptor. The EB=s_x for CAD-120 is equal to 6690.

	Acc [%]	σ
HOG(EB=Sx)	60.64	1.71
HOG(EB=5000)	58.06	3.42
HOG(EB=10000)	57.26	2.28
HOG(EB=15000)	56.85	5.13
HOG(EB=20000)	56.05	1.71
HOG	54.84	1.14
HOG(EB=25000)	54.44	1.71
HOG(EB=30000)	52.82	0.57
HOG(EB=35000)	52.42	3.42
HOG(EB=40000)	51.61	1.14

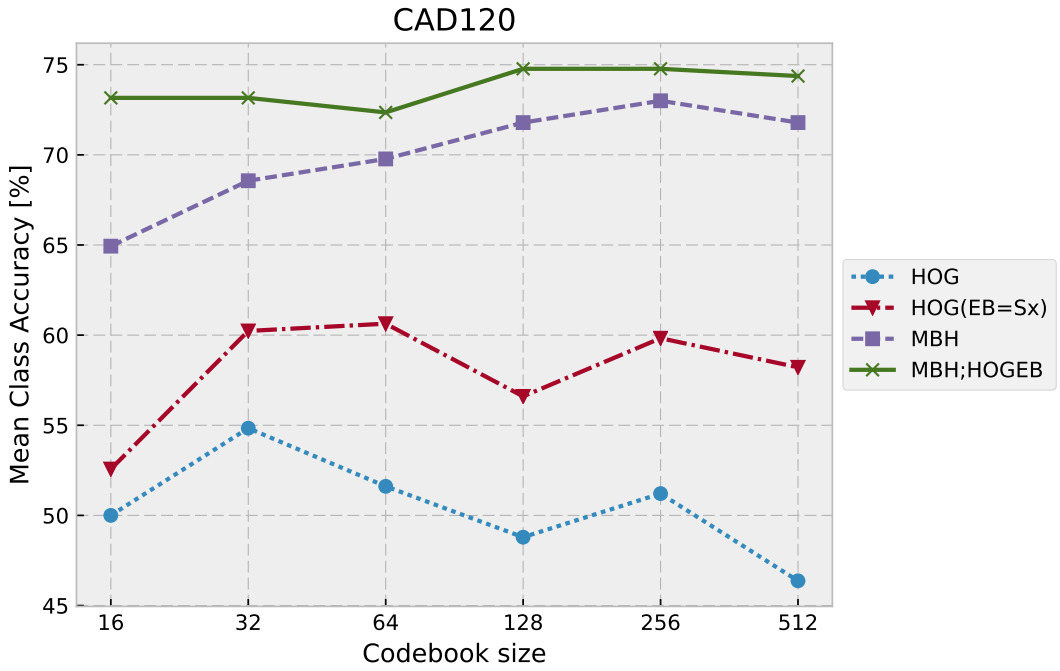
Table 5.4: Results on CAD-120 dataset. The table shows Mean Class Accuracy for HOG normalized with different EB values, standard HOG performance and EB=s_x. The table shows that HOG all tested EB values as well as EB=s_x outperform standard HOG descriptor. The EB=s_x for CAD-120 is equal to 6690.

Table 5.5: Results on CAD-120 dataset, the plot and table show Mean Class Accuracy *w.r.t.* codebook size of standard HOG, HOG normalized with EB value equals to 5000 (best performing value found via cross-validation) and EB=s_x. The EB=s_x for CAD-120 is equal to 6690.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
HOG(EB=Sx)	52.57	3.99	60.23	2.28	60.64	1.71	56.60	4.56	59.83	2.85	58.22	6.27	60.64	1.71
HOG(EB=5000)	51.61	1.14	58.06	3.42	52.82	3.42	50.81	4.56	52.02	1.71	53.63	1.71	58.06	3.42
HOG	50.00	2.28	54.84	1.14	51.61	1.14	48.79	2.85	51.21	5.13	46.37	3.99	54.84	1.14

Table 5.6: Results on CAD-120 dataset, the plot and table show Mean Class Accuracy *w.r.t.* codebook size of standard HOG, standard DT descriptor, HOG normalized with EB=s_x and DT;HOGEB where standard HOG was replaced with HOG(EB=s_x). The EB=s_x for CAD-120 is equal to 6690.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
MBH;HOGEB	73.16	3.42	73.16	3.42	72.35	3.42	74.77	1.14	74.77	2.28	74.37	1.71	74.77	1.14
MBH	64.94	1.14	68.56	1.71	69.77	0.00	71.79	2.85	73.00	1.14	71.79	3.99	73.00	1.14
HOG(EB=Sx)	52.57	3.99	60.23	2.28	60.64	1.71	56.60	4.56	59.83	2.85	58.22	6.27	60.64	1.71
HOG	50.00	2.28	54.84	1.14	51.61	1.14	48.79	2.85	51.21	5.13	46.37	3.99	54.84	1.14

5.4.3 MSRDailyActivity3D dataset

We follow similar analysis for MSRDailyActivity3D dataset. The fig. 5.6 and table 5.7 show the performance of HOG descriptor *w.r.t.* to different EB value (for each EB value best performing codebook size was selected). In addition we report also performance of standard HOG descriptor and one normalized with EB$=s_x$ value which in case of MSRDailyActivity3D dataset equals to 7696 and was obtained with algorithm described in section 5.2.4. The results in fig. 5.6 and table 5.7 show that:

- proposed normalization schema improves HOG performance for all selected EB values,
- HOG(EB$=s_x$) descriptor with EB value found with proposed expectation maximization algorithm (see section 5.2.4) outperforms standard HOG descriptor and all HOG descriptors normalized with EB values found with extensive search.

In table 5.8 we compare HOG(EB$=$20000) – best performing descriptor with EB value found via extensive search, HOG(EB$=s_x$) and standard HOG. The results show that:

- HOG descriptor is consistently outperformed by both: HOG(EB$=$20000) and HOG(EB$=s_x$) *w.r.t.* all codebook sizes,
- HOG(EB$=s_x$) outperforms standard HOG.
- HOG(EB$=s_x$) outperforms HOG(EB$=$20000). If we consider standard deviation of the mentioned results the mentioned descriptors achieve comparable results. This shows that the algorithm proposed in section 5.2.4 found as good value as extensive search.

Finally in table 5.9 we show the results of DT;HOGEB descriptor which is fusion of Dense Trajectories descriptors (TSD, HOG, HOF, MBH), where HOG was replaced with HOG(EB$=s_x$). The results shows that further fusion of HOG(EB$=s_x$) descriptor improves performance of standard DT descriptor.

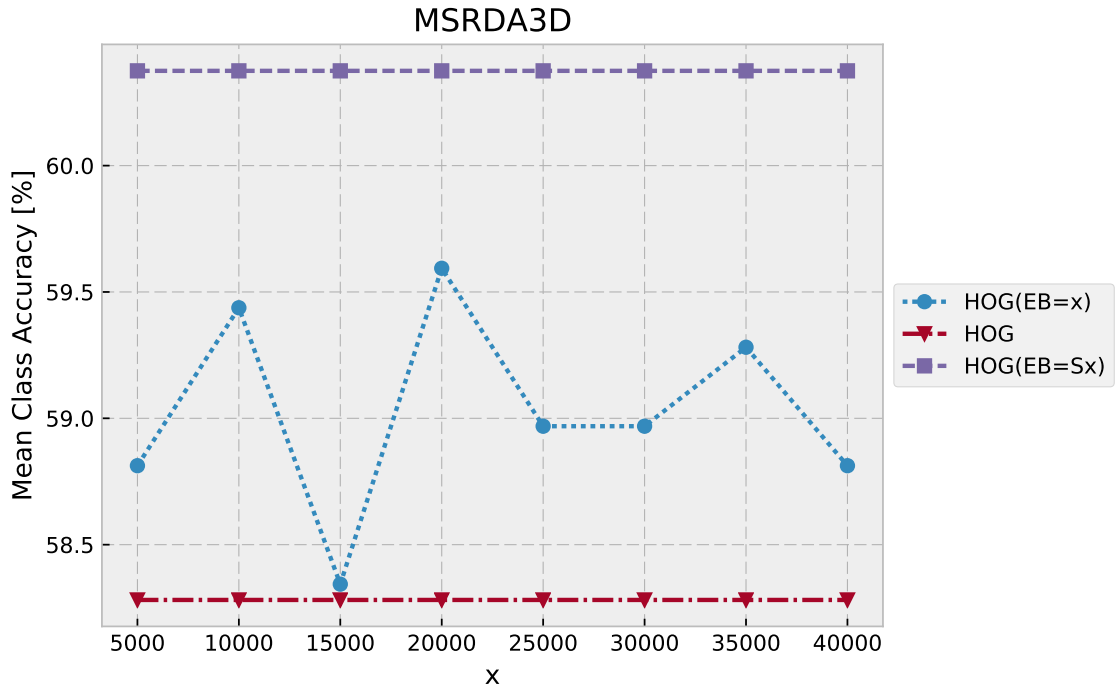
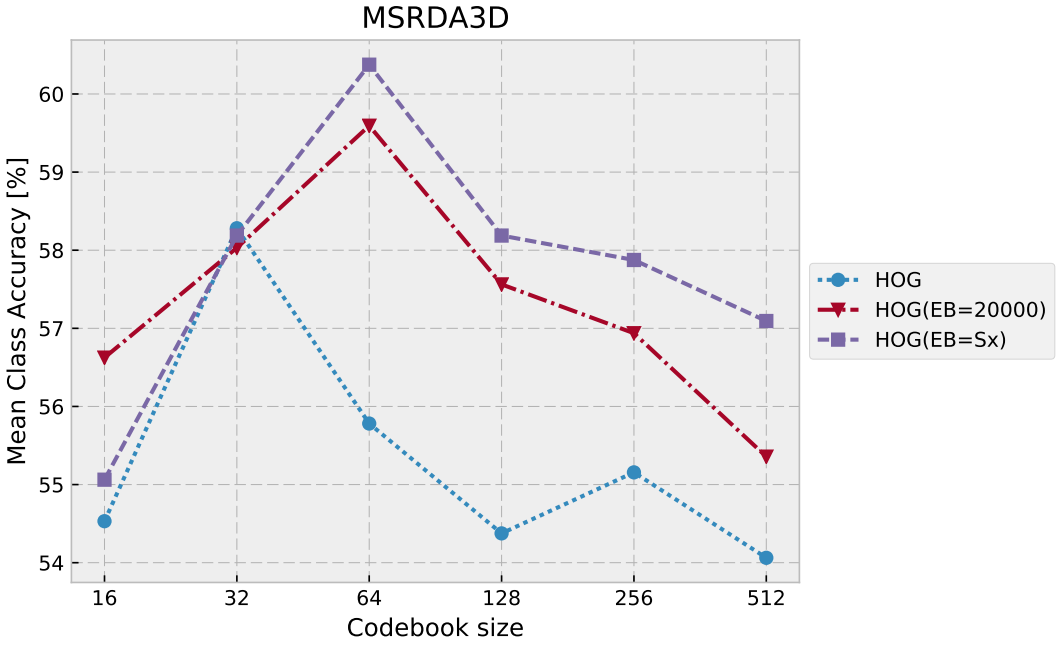


Figure 5.6: Results on MSRDailyActivity3D dataset. The plot shows Mean Class Accuracy for HOG normalized with different EB values (blue dotted line). The standard HOG performance is showed in red, while EB=s_x is shown in purple. The figure shows that HOG all tested EB values as well as EB=s_x outperform standard HOG descriptor. The EB=s_x for MSRDailyActivity3D is equal to 7696.

	Acc [%]	σ
HOG(EB=Sx)	60.38	1.33
HOG(EB=20000)	59.59	1.10
HOG(EB=10000)	59.44	0.88
HOG(EB=35000)	59.28	1.55
HOG(EB=25000)	58.97	0.88
HOG(EB=30000)	58.97	1.33
HOG(EB=40000)	58.81	2.21
HOG(EB=5000)	58.81	0.44
HOG(EB=15000)	58.34	0.66
HOG	58.28	0.66

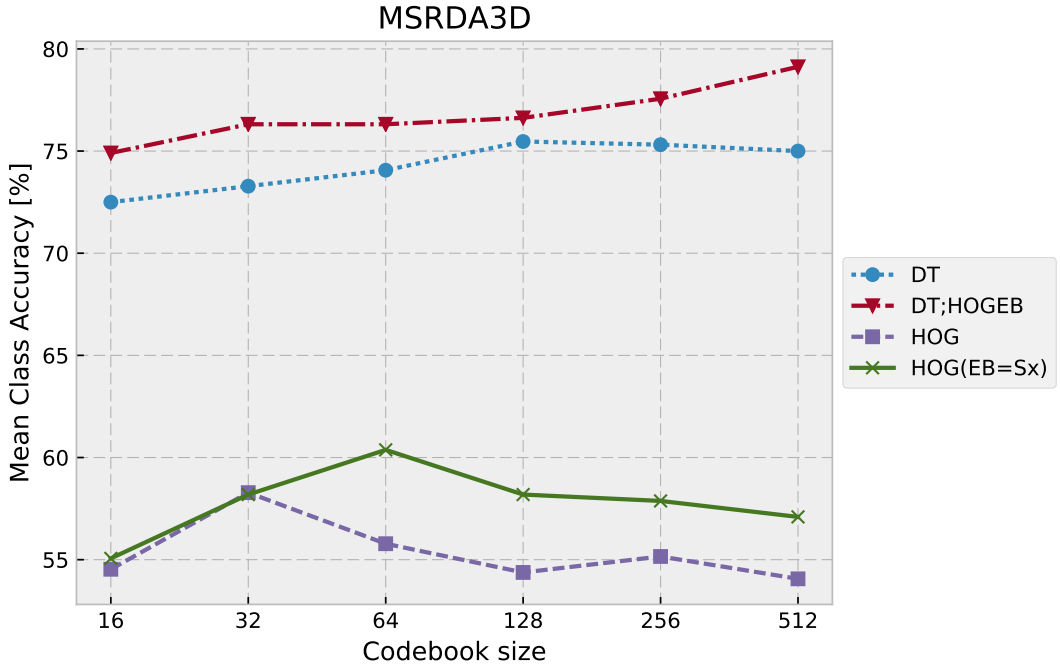
Table 5.7: Results on MSRDailyActivity3D dataset. The table shows Mean Class Accuracy for HOG normalized with different EB values, standard HOG performance and EB=s_x. The table shows that HOG all tested EB values as well as EB=s_x outperform standard HOG descriptor. The EB=s_x for MSRDailyActivity3D is equal to 7696.

Table 5.8: Results on MSRDailyActivity3D dataset, the plot and table show Mean Class Accuracy *w.r.t.* codebook size of standard HOG, HOG normalized with EB value equals to 20000 (best performing value found via cross-validation) and EB$=s_x$. The EB$=s_x$ for MSRDailyActivity3D is equal to 7696.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
HOG(EB=Sx)	55.06	2.65	58.19	2.21	60.38	1.33	58.19	0.00	57.88	0.88	57.09	2.04	60.38	1.33
HOG(EB=20000)	56.62	1.55	58.03	2.43	59.59	1.10	57.56	0.66	56.94	0.88	55.36	2.23	59.59	1.10
HOG	54.53	0.22	58.28	0.66	55.78	0.66	54.37	0.44	55.16	0.22	54.06	0.00	58.28	0.66

Table 5.9: Results on MSRDailyActivity3D dataset, the plot and table show Mean Class Accuracy *w.r.t.* codebook size of standard HOG, standard DT descriptor, HOG normalized with EB=s_x and DT;HOGEB where standard HOG was replaced with HOG(EB=s_x). The EB=s_x for MSRDailyActivity3D is equal to 7696.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
DT;HOGEB	74.91	1.77	76.31	0.44	76.31	1.33	76.63	0.66	77.56	1.33	79.12	0.00	79.12	0.00
DT	72.50	0.88	73.28	0.22	74.06	0.88	75.47	0.22	75.31	1.33	75.00	0.00	75.47	0.22
HOG(EB=Sx)	55.06	2.65	58.19	2.21	60.38	1.33	58.19	0.00	57.88	0.88	57.09	2.04	60.38	1.33
HOG	54.53	0.22	58.28	0.66	55.78	0.66	54.37	0.44	55.16	0.22	54.06	0.00	58.28	0.66

5.4.4 Smarthomes dataset

We follow similar analysis for Smarthomes dataset. The fig. 5.7 and table 5.10 show the performance of HOG descriptor *w.r.t.* to different EB value (for each EB value best performing codebook size was selected). In addition we report also performance of standard HOG descriptor and one normalized with EB$=s_x$ value which in case of Smarthomes dataset equals to 7738 and was obtained with algorithm described in section 5.2.4. The results in fig. 5.7 and table 5.10 show that:

- proposed normalization schema improves HOG performance for all selected EB values,
- HOG(EB$=s_x$) descriptor with EB value found with proposed expectation maximization algorithm (see section 5.2.4) outperforms standard HOG descriptor and all HOG descriptors normalized with EB values found with extensive search.

HOG descriptor normalized with EB$=s_x$ still achieves best results, such result is not fully intuitive because fig. 5.3 shows that in case of Smarthomes dataset distribution of gradient magnitudes forms single Gaussian. In Smarhomes dataset people are much further from the camera than in CAD-60, CAD-120 and MSRDailyActivity3D, because of that HOG descriptor is not able to catch as many details and distribution does not show separation of texture and texture-less Gaussians. Unfortunately we cannot offer explanation why using our Expectation Maximization algorithm is still good strategy to find EB value, even if modeled distribution is unimodal. As shown in table 5.11 HOG descriptor normalized with EB$=s_x$, achieves better results than one found via cross-validation.

Finally in table 5.12 we show the results of DT;HOGEB descriptor which is fusion of Dense Trajectories descriptors (TSD, HOG, HOF, MBH), where HOG was replaced with HOG(EB$=s_x$). The results shows that further fusion of HOG(EB$=s_x$) descriptor improves performance of standard DT descriptor. Recognition of actions with similar motion pattern but different object involved was improved. Recognition of actions like: "use tablet", "eat snack", "cut bread" was improved comparing to standard DT descriptor.

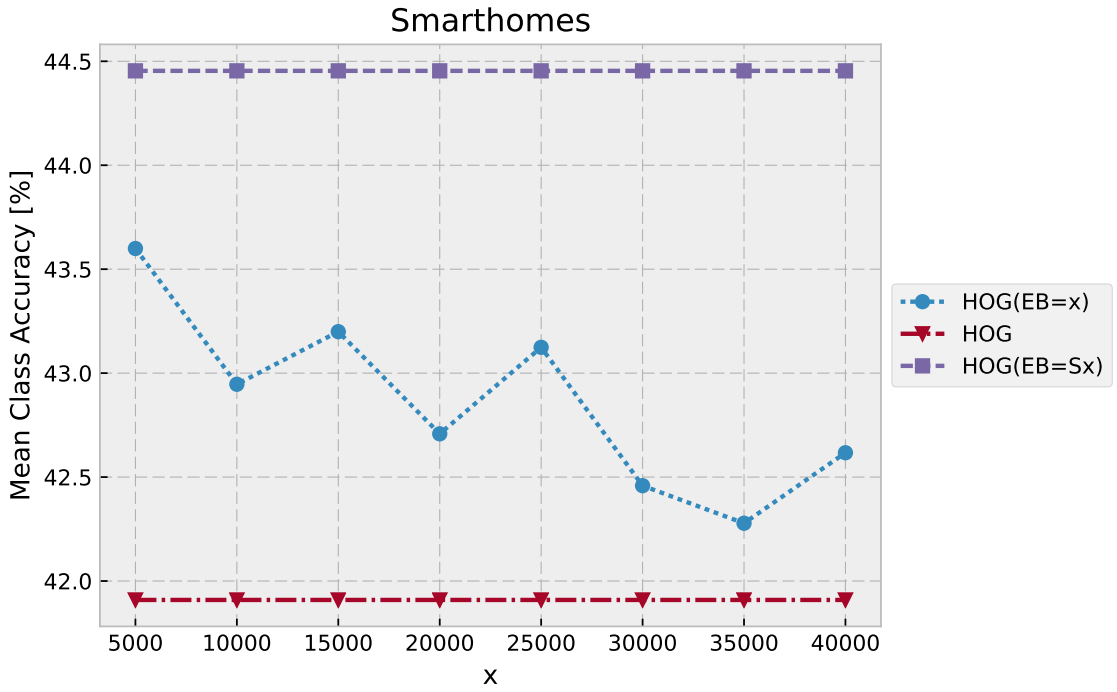
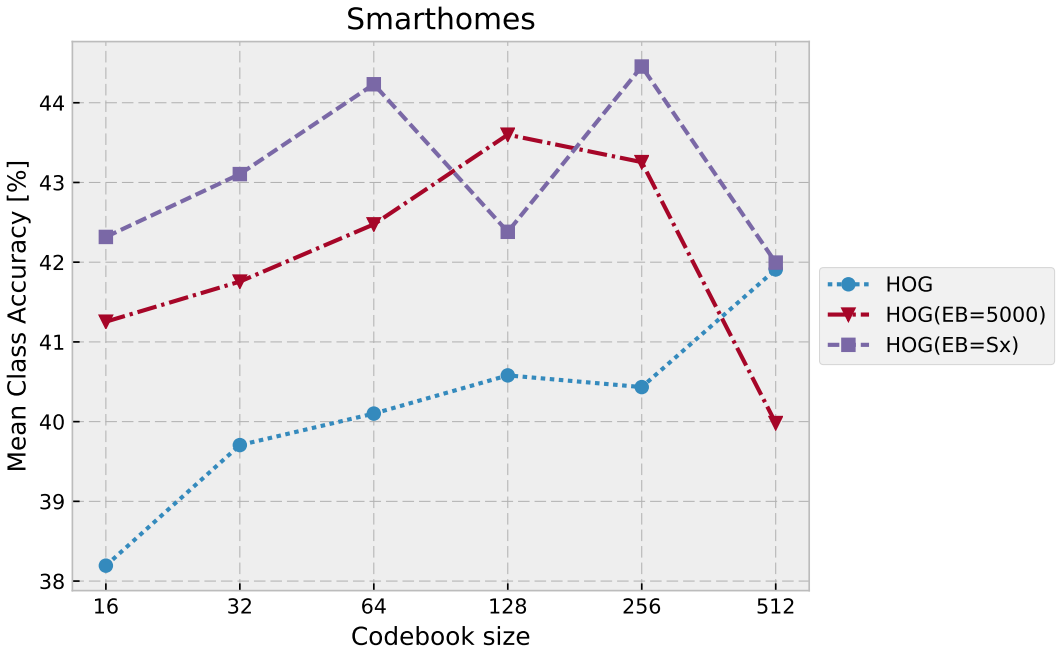


Figure 5.7: Results on Smarthomes dataset. The plot shows Mean Class Accuracy for HOG normalized with different EB values (blue dotted line). The standard HOG performance is showed in red, while EB=s_x is shown in purple. The figure shows that HOG all tested EB values as well as EB=s_x outperform standard HOG descriptor. The EB=s_x for Smarthomes is equal to 7738.

	Acc [%]	σ
HOG(EB=Sx)	44.45	0.87
HOG(EB=5000)	43.60	0.29
HOG(EB=15000)	43.20	2.36
HOG(EB=25000)	43.12	0.07
HOG(EB=10000)	42.95	1.10
HOG(EB=20000)	42.71	1.56
HOG(EB=40000)	42.62	0.26
HOG(EB=30000)	42.46	2.14
HOG(EB=35000)	42.28	0.10
HOG	41.91	0.34

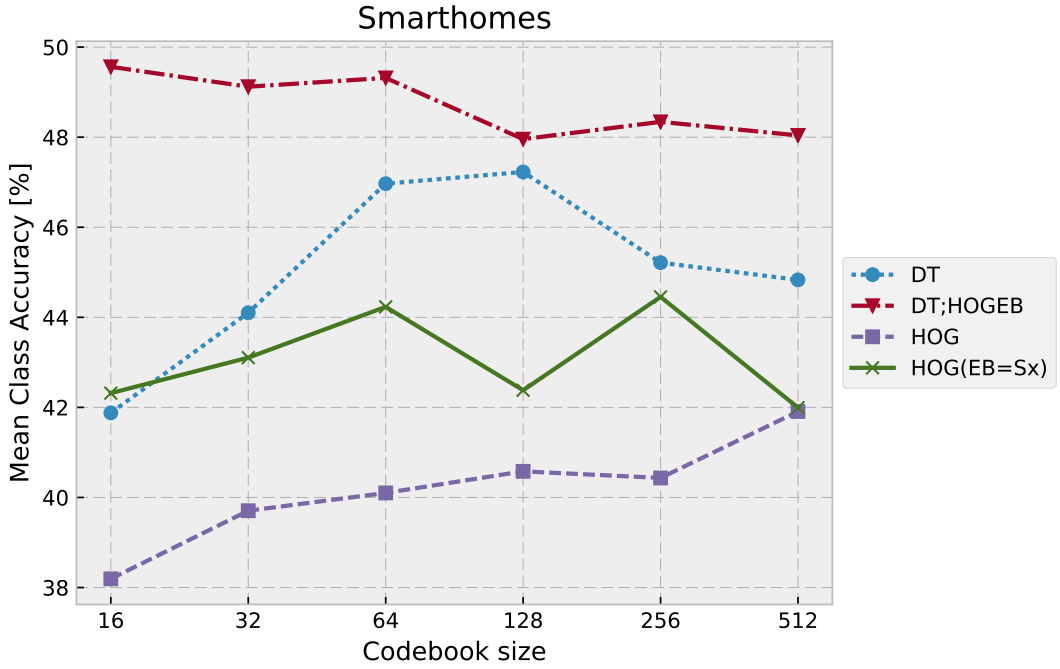
Table 5.10: Results on Smarthomes dataset. The table shows Mean Class Accuracy for HOG normalized with different EB values, standard HOG performance and EB=s_x. The table shows that HOG all tested EB values as well as EB=s_x outperform standard HOG descriptor. The EB=s_x for MSRDailyActivity3D is equal to 7738.

Table 5.11: Results on Smarthomes dataset, the plot and table show Mean Class Accuracy *w.r.t.* codebook size of standard HOG, HOG normalized with EB value equals to 5000 (best performing value found via cross-validation) and EB=s_x. The EB=s_x for Smarthomes is equal to 7738.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
HOG(EB=Sx)	42.32	3.31	43.10	1.67	44.23	1.53	42.38	2.03	44.45	0.87	41.99	0.95	44.45	0.87
HOG(EB=5000)	41.25	0.67	41.76	2.10	42.47	0.94	43.60	0.29	43.26	0.88	39.98	0.18	43.60	0.29
HOG	38.19	0.08	39.71	0.13	40.10	0.60	40.58	0.79	40.43	0.74	41.91	0.34	41.91	0.34

Table 5.12: Results on Smarthomes dataset, the plot and table show Mean Class Accuracy *w.r.t.* codebook size of standard HOG, standard DT descriptor, HOG normalized with EB=s_x and DT;HOGEB where standard HOG was replaced with HOG(EB=s_x). The EB=s_x for Smarthomes is equal to 7738.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
DT;HOGEB	49.56	0.37	49.12	0.61	49.32	0.13	47.96	2.30	48.34	1.69	48.04	0.23	49.56	0.37
DT	41.88	0.07	44.10	0.04	46.97	0.77	47.23	0.20	45.21	2.36	44.83	0.23	47.23	0.20
HOG(EB=Sx)	42.32	3.31	43.10	1.67	44.23	1.53	42.38	2.03	44.45	0.87	41.99	0.95	44.45	0.87
HOG	38.19	0.08	39.71	0.13	40.10	0.60	40.58	0.79	40.43	0.74	41.91	0.34	41.91	0.34

5.5 Conclusion

This chapter presents a new normalization scheme of histogram descriptors that effectively minimizes a hallucination effect. We have demonstrated that while normalizing HOG, the value of EB has impact on the performance. We have also shown that extra bin used in normalization step should be kept in final histogram representation. We model the gradient sum distribution as the Mixture of two Gaussian obtaining a separation of high and low texture patches. The intersection of these distributions gives us the value of the extra bin (EB) that is added to the set of gradient orientations in the HOG descriptor. This is effective strategy to find the value of EB, and works in case where patches can be divided into texture and texture-less groups. In other cases our algorithm still finds good EB value, while we cannot offer exact explanation to that fact, the experiments show that values between 5000 to 15000 seems to be most effective. The differences in performance of HOG normalized with different EB are small for big range of EB values, thus finding an effective EB by cross-validation is also not difficult. Our approach consistently improves accuracy rate in action recognition of HOG descriptor as well as DT descriptor.

Chapter 6

Brownian Covariance

Contents

6.1 Introdction

In image processing, at some point of time a novel trend has emerged that ignores explicit values of given features, focusing instead on their pairwise relations. Such approach is an alternative to descriptors which employs simple statistic (*eg.* histograms) of extracted features (*eg.* HOG, HOF, MBH). If we analyze in details relations between different features, we can find that many of these relationships are non-linear. For instance fig. 6.1 shows

the correlation between intensity and gradient extracted from an image patch. The corresponding intensity and gradient values are plotted together to show their dependency. Based on the plot we can see that the relationship is non-linear and non-monotone.

To model the relationships described above we propose the descriptor based on Brownian covariance [108, 4, 3, 2, 10], which is a result of recent advances in mathematical statistics of Brownian motion. Brownian covariance is able to measure the degree of all kinds of possible relationships – including nonlinear and non-monotone. In addition in [2], authors provide detailed analysis of Brownian manifold and show that it is flat enough, to be treated as Euclidean. In this chapter we propose Brownian descriptor, which is an extension of Brownian covariance for space-time video volumes. Using the proposed descriptor we represent relations between appearance features extracted from a video sequence. We focus particularly on relationships between pixel level features such as an intensity, image gradient and image second derivative. The mentioned pixel level features model appearance rather than motion. While it is a well known fact that motion features play key role in action recognition, it is also important to model the appearance, to be able to distinguish actions with similar motion footprint, but with different objects involved. The HOG descriptor throughout the years was the most popular descriptor to capture appearance in action action recognition. We propose to apply our Brownian descriptor on appearance features, and compare it with HOG. We improve appearance representation, because it is usually performing worse than motion representation, but theirs fusion leads to performance improvements (both representations carry complementary information). Secondly in appearance based tasks like: tracking, object detection and people re-identification, descriptors which model pixel-level relationships have shown good performance [85, 3, 2]. The further studies show that Brownian descriptor and HOG carry complementary information and their fusion leads to significant improvement in recognition accuracy.

6.2 Brownian Covariance

Brownian descriptor is based on mathematical statistics theory, which describes Brownian motion [108]. The descriptor is based on the *distance covariance* statistics that measures the dependence between random vectors in the arbitrary dimension. In the following sections we describe *distance covariance* $\mathcal{V}^2$, *sample distance covariance* $\mathcal{V}_n^2$ and their relations to Brownian covariance $\mathcal{W}$. The mathematical notations and formulas provided in this subsection are in accordance with [108].

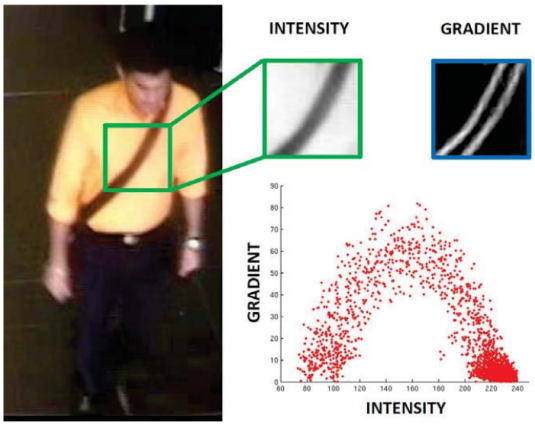


Figure 6.1: Example of non-linear relationship between intensity and gradient. Brownian covariance is able to measure such kind of relationship. The figure courtesy [2].

6.2.1 Distance Covariance $\mathcal{V}^2$

Let $X \in \mathbb{R}^p$ and $Y \in \mathbb{R}^q$ be random vectors, where p and q are natural numbers. f_X and f_Y denote the characteristic functions of X and Y, respectively, and their joint characteristic function is denoted as $f_{X,Y}$. In terms of characteristic functions, X and Y are independent if and only if $f_{X,Y} = f_X f_Y$. Thus, a natural way of measuring the dependence between X and Y is to find a suitable norm to measure the distance between $f_{X,Y}$ and $f_X f_Y$.

Distance covariance $\mathcal{V}^2$ [108] is a new measure of dependence between random vectors and can be defined as:

$$\mathcal{V}^2(X,Y) = ||f_{X,Y}(t,s) - f_X(t)f_Y(s)||^2 \tag{6.1}$$

$$= \frac{1}{c_p c_q} \int_{\mathbb{R}^{p+q}} \frac{|f_{X,Y}(t,s) - f_X(t)f_Y(s)|^2}{|t|_p^{1+p}|s|_q^{1+q}} dtds, \tag{6.2}$$

where c_p and c_q are constants determining norm function in $\mathbb{R}^p \times \mathbb{R}^q$, $t \in X$, $s \in Y$.

This measure is analogous to classical covariance, but with the important property that $\mathcal{V}^2(X,Y) = 0$ if and only if X and Y are independent.

6.2.2 Sample Distance Covariance $\mathcal{V}^2$

In practice, when designing new descriptor we are more interested in modeling relationship between finite distribution (finite number of pixels). In such case we can employ the sample distance covariance. Szekely *et al.* [108] provide us the following definition of a sample distance covariance $\mathcal{V}_n^2$. For a random sample $(\mathbf{X}, \mathbf{Y}) = \{(X_k, Y_k) : k = 1 \dots n\}$ of n i.i.d random vectors (X, Y) from their joint distribution, we compute the Euclidean

distance matrices $(a_{kl}) = (|X_k - X_l|_p)$ and $(b_{kl}) = (|Y_k - Y_l|_q)$. Define:

$$A_{kl} = a_{kl} - \bar{a}_{k\cdot} - \bar{a}_{\cdot l} + \bar{a}_{\cdot\cdot}, \quad k, l = 1, \ldots, n, \tag{6.3}$$

where:

$$\bar{a}_{k\cdot} = \frac{1}{n}\sum_{l=1}^{n} a_{kl}, \quad \bar{a}_{\cdot l} = \frac{1}{n}\sum_{k=1}^{n} a_{kl}, \quad \bar{a}_{\cdot\cdot} = \frac{1}{n^2}\sum_{k,l=1}^{n} a_{kl}. \tag{6.4}$$

Similarly, we define $B_{kl} = b_{kl} - \bar{b}_{k\cdot} - \bar{b}_{\cdot l} + \bar{b}_{\cdot\cdot}$. Having these simple linear functions of the pairwise distances between sample elements of X and Y distributions, we define distance covariance as:

$$\mathcal{V}_n^2(X, Y) = \frac{1}{n^2}\sum_{k,l=1}^{n} A_{kl} B_{kl}. \tag{6.5}$$

6.2.3 Standardization

Similarly to covariance, which has its standardized counterpart ρ, $\mathcal{V}_n^2$ has its standardized version referred to as *distance correlation* $\mathcal{R}_n^2$, defined as:

$$\mathcal{R}_n^2(X, Y) = \begin{cases} \frac{\mathcal{V}_n^2(X,Y)}{\sqrt{\mathcal{V}_n^2(X)\mathcal{V}_n^2(Y)}}, & \mathcal{V}_n^2(X)\mathcal{V}_n^2(Y) > 0; \\ 0, & \mathcal{V}_n^2(X)\mathcal{V}_n^2(Y) = 0, \end{cases} \tag{6.6}$$

where:

$$\mathcal{V}_n^2(X) = \mathcal{V}_n^2(X, X) = \frac{1}{n^2}\sum_{k,l=1}^{n} A_{kl}^2. \tag{6.7}$$

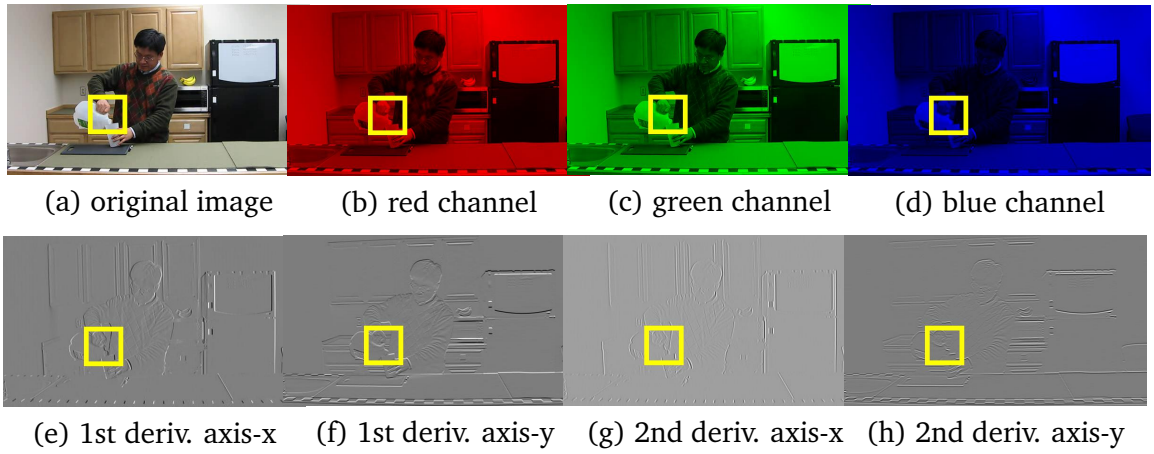
6.2.4 Brownian covariance $\mathcal{W}$

Brownian motion is a stochastic process which models random movements of particles in fluids. The interactions between particles can be expressed by Brownian covariance. Let's define Brownian covariance as $\mathcal{W}$. According to Szekely *et al.* [108] $\mathcal{W}$ measures all kinds of possible relationships between random particles (variables). This implies that $\mathcal{W} = 0$ if and only if X and Y are independent. For arbitrary $X \in \mathbb{R}^p$ and $Y \in \mathbb{R}^q$ with finite second moments.

$$\mathcal{W}(X, Y) = \mathcal{V}(X, Y) \tag{6.8}$$

The proof is available in Theorem 8 in [108]. Further Theorem 2 from [108] states that: if $E|X|_p < \infty \wedge E|Y|_q < \infty$, *then almost surely*:

$$\lim_{n\to\infty} \mathcal{V}_n(X, Y) = \mathcal{V}(X, Y) \tag{6.9}$$



(a) original image (b) red channel (c) green channel (d) blue channel

(e) 1st deriv. axis-x (f) 1st deriv. axis-y (g) 2nd deriv. axis-x (h) 2nd deriv. axis-y

Figure 6.2: Low-level appearance features extracted in a video frame. Yellow rectangle indicates a sample patch.

From eq. (6.8) and eq. (6.9), we can see that:

$$\mathcal{W}(X,Y) = \lim_{n\to\infty} \mathcal{V}_n(X,Y) \propto \mathcal{R}_n^2(X,Y) \tag{6.10}$$

Thus based on above if $\mathcal{R}_n^2(X,Y) = 0$, we will assume that there is no dependence between variables.

6.3 Brownian Covariance Descriptor for Action Recognition

We propose to model a relationship of each patch of video volume of size $S \times S \times t$, where S is size of patch in pixels and t is number of frames. The patch is obtained by cropping $S \times S$ pixels around detected Local Point of Interest. Please note that at this point we do not assume usage of any particular detector, but in our experiments we use Dense Trajectories.

6.3.1 Low-Level Appearance Features

Brownian(7)

In this paragraph we describe the Brownian descriptor which models relationship between seven low-level features. For each patch of a video volume, we compute seven low-level appearance features. For every pixel we extract intensities in red, green, and blue channels, first and second order derivatives of grey scale intensity image along x and y axis. Thus, every pixel at time t can be expressed in the following vector form:

$$f_t = \left[\boldsymbol{R}, \boldsymbol{G}, \boldsymbol{B}, \frac{\partial \boldsymbol{I}}{\partial x}, \frac{\partial \boldsymbol{I}}{\partial y}, \frac{\partial^2 \boldsymbol{I}}{\partial x^2}, \frac{\partial^2 \boldsymbol{I}}{\partial y^2}\right], \tag{6.11}$$

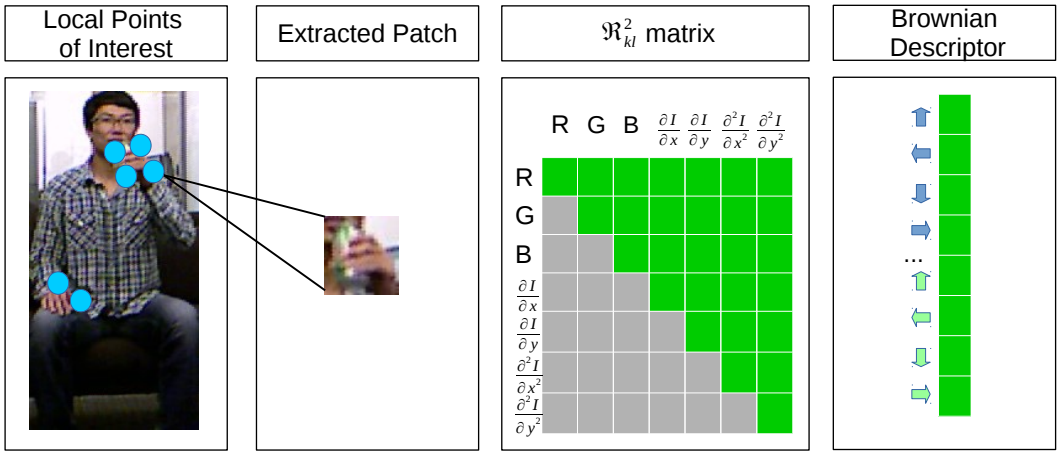


Figure 6.3: Brownian Descriptor is created by taking upper half of *distance correlation* $\mathcal{R}^2$.

where I is a gray scale intensity image. The examples of the extracted low-level appearance features are presented in Figure 6.2.

Brownian(9)

In this paragraph we describe the Brownian descriptor which models relationship between nine low-level features. In this case we use all low-level features used in **Brownian(7)** and we add spatial location (x, y) of each pixel. Thus **Brownian(9)** descriptor can be defined as:

$$f_t = \left[\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{R}, \boldsymbol{G}, \boldsymbol{B}, \frac{\partial \boldsymbol{I}}{\partial x}, \frac{\partial \boldsymbol{I}}{\partial y}, \frac{\partial^2 \boldsymbol{I}}{\partial x^2}, \frac{\partial^2 \boldsymbol{I}}{\partial y^2} \right], \tag{6.12}$$

6.3.2 Descriptor Definition

For each video volume patch we compute Brownian covariance $R^2_n(X, Y)$, between all pairs of mentioned above low-level features. Because we use either 7 or 9 low-level features the R^2_n is 7×7 or 9×9 matrix. To form Brownian descriptor d_i for given patch we take upper half of the R^2_n matrix as shown in fig. 6.3.

The d_i describes only single patch of volume of length t. To describe the whole volume we average the descriptors from each frame. Thus the final Brownian descriptor is defined as follows:

$$d = \frac{1}{t} \sum_{i=1}^{t} d_i, \tag{6.13}$$

6.3.3 Features Extraction Detalis

We use dense trajectories [113] to extract local spatio-temporal patches. The dense trajectories were selected based on their use in the recent literature. However, our approach can be used together with any other algorithm extracting local spatio-temporal patches. By extracting dense trajectories, we provide a good coverage of a video and we ensure extraction of meaningful features. We limit the length of trajectories to $t = 15$ frames. Short trajectories are more robust than long trajectories, in particular in the presence of fast irregular motions and when the trajectories are drifting.

Similarly to [113], we extract a space-time volume (*i.e.* a patch) of size 32×32 pixels and 15 frames around each trajectory. The volume is subdivided into 3 temporal cells of $l = 5$ frames. For each cell we compute a descriptor, and we concatenate the descriptor of each cell to create a final trajectory descriptor.

6.4 Experiments

In this section we will report the performance of the proposed normalization method on 4 datasets:

- CAD-60
- CAD-120
- MSRDailyActivity3D
- Smarthomes

Since we use Fisher Vector encoding we provide ablation analysis for different codebook sizes: 16, 32, 64, 128, 256, 512. We compare performance of Brownian descriptor with standard HOG, then we analyze fusion of these two descriptors. Finally we compare Dense Trajectories descriptor (DT) with standard HOG to DT+Brownian descriptor where we add Brownian descriptor to DT descriptors.

In this section we use the following abbreviations:

- HOG – standard HOG descriptor,
- Brownian(7) – Brownian descriptor proposed in this chapter, which models 7 pixel-level features,
- Brownian(9) – Brownian descriptor proposed in this chapter, which models 9 pixel-level features,
- HOG+Brownian(7) – fusion HOG descriptor and Brownian(7) descriptor ,

- HOG+Brownian(9) – fusion HOG descriptor and Brownian(9) descriptor ,
- DT – Dense Trajectories descriptor *i.e.* fusion of TSD, HOG, HOF, MBH,
- DT+Brownian(7) – Dense Trajectories descriptor fused together with Brownian(7) descriptor.
- DT+Brownian(9) – Dense Trajectories descriptor fused together with Brownian(9) descriptor.

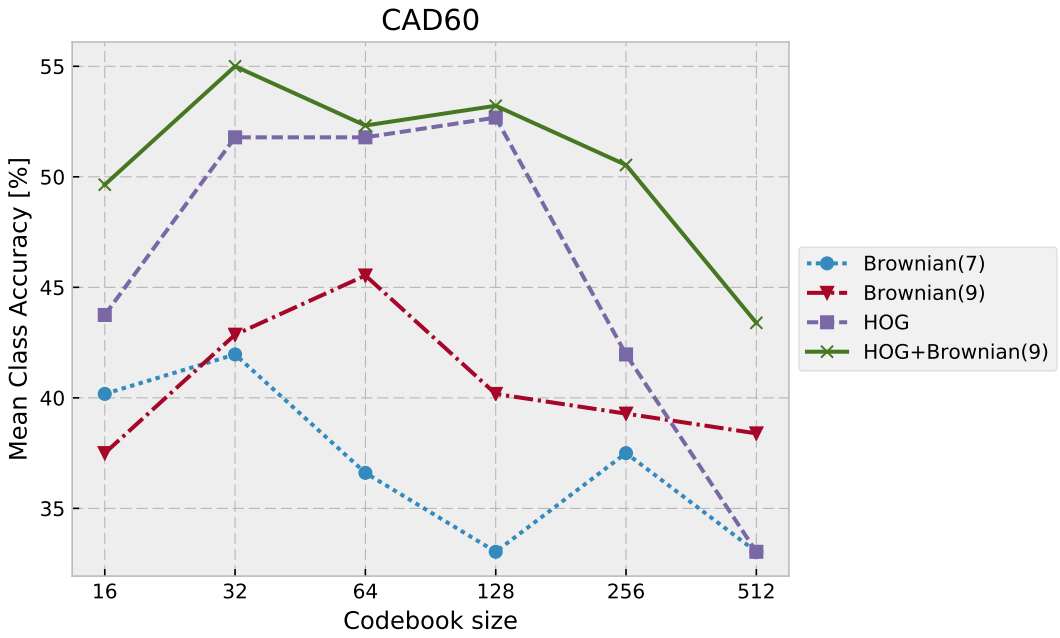
6.4.1 CAD-60

In table 6.1 we compare standard HOG descriptor to Brownian descriptor and to descriptor which is fusion of these two (HOG+Brownian). Based on the experimental results on CAD-60 dataset we can find the following conclusions:

- Brownian(7) and Brownian(9) as a standalone descriptors are outperformed by standard HOG descriptor by big margin.
- Brownian carries complementary information as fusion of HOG and Brownian leads to improvement in recognition accuracy.
- Brownian(9) which models 9 pixel-level features performs better thatn Brownian(7)
- Brownian(9) descriptor carries complementary information as fusion with HOG leads to improvement in recognition accuracy.
- HOG+Brownian(9) descriptor outperforms HOG.

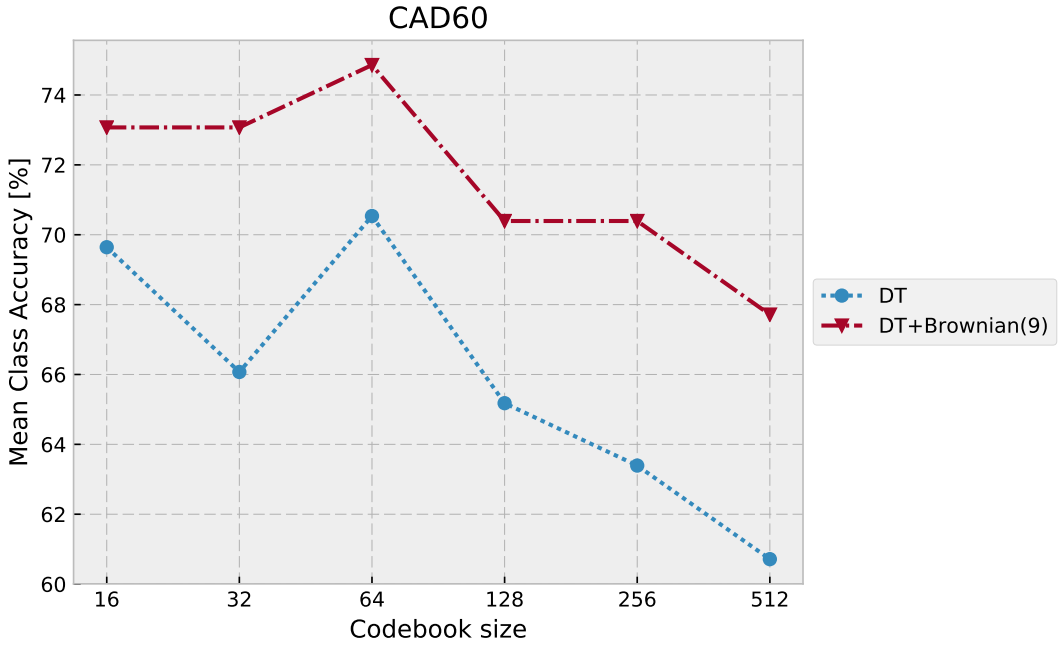
Finally in table 6.2 we show the results of DT+Brownian(9) descriptor which is a fusion of Dense Trajectories descriptors (TSD, HOG, HOF, MBH) and Brownian(9) descriptor. The results shows that further fusion of Brownian(9) descriptor improves performance of standard DT descriptor. The detailed analysis shows that DT+Brownian(9) descriptor improved for instance: "opening pill container" action which was confused with "drinking". These actions share similar motion pattern but are different in terms of appearance (different objects are used). Thus Brownian descriptor proposed in this chapter, which models appearance is able to improve the recognition accuracy.

Table 6.1: Results on CAD-60 dataset, the plot and table show Mean Class Accuracy *w.r.t.* codebook size of standard HOG descriptor, Brownian descriptor and HOG+Brownian descriptor, which is a fusion of HOG and Brownian.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
HOG+Brownian(9)	49.64	5.05	55.00	1.03	52.32	1.26	53.21	1.26	50.54	1.26	43.39	3.79	55.00	1.03
HOG	43.75	1.26	51.79	0.00	51.79	0.00	52.68	3.79	41.96	6.31	33.04	1.26	52.68	3.79
Brownian(9)	37.50	2.53	42.86	0.00	45.54	3.79	40.18	3.79	39.29	2.53	38.39	1.26	45.54	3.79
Brownian(7)	40.18	3.79	41.96	1.26	36.61	6.31	33.04	3.79	37.50	2.53	33.04	1.26	41.96	1.26

Table 6.2: Results on CAD-60 dataset, the plot and table show Mean Class Accuracy *w.r.t.* codebook size of standard DT descriptor (TSD, HOG, HOF, MBH) and DT+Brownian which is a fusion of DT descriptors and Brownian.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
DT+Brownian(9)	73.07	2.53	73.07	1.26	74.86	1.26	70.39	3.79	70.39	2.53	67.71	2.53	74.86	1.26
DT	69.64	2.53	66.07	2.53	70.54	1.26	65.18	1.26	63.39	1.26	60.71	2.53	70.54	1.26

6.4.2 CAD-120

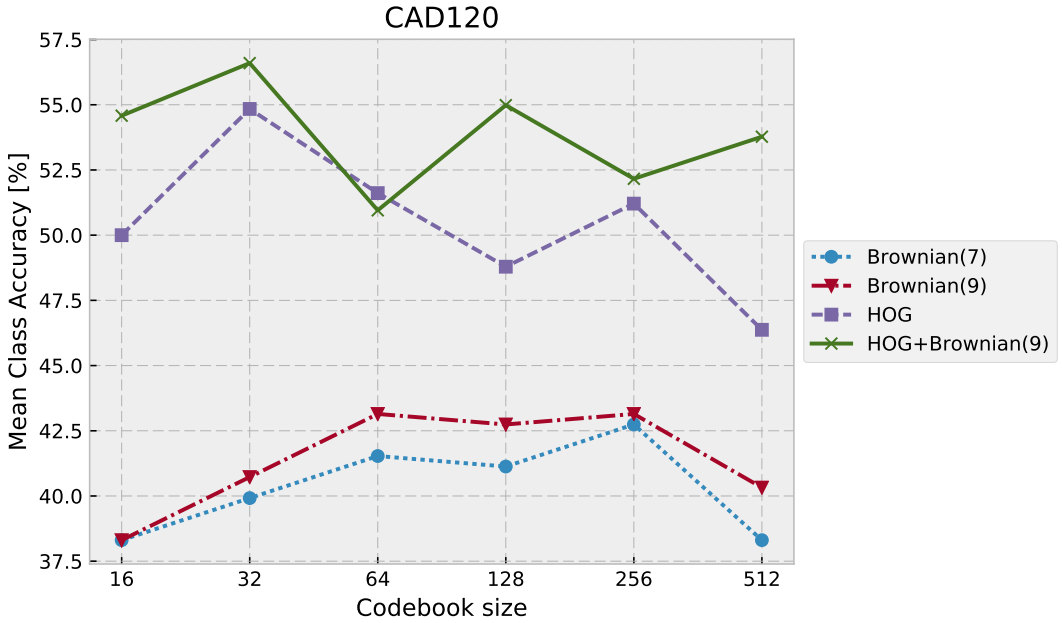
In table 6.3 we compare standard HOG descriptor to Brownian descriptor and to descriptor which is fusion of those two (HOG+Brownian). Based on the experimental results on CAD-120 dataset we can see that conclusions are similar to CAD-60 dataset as:

- Brownian(7) and Brownian(9) as standalone descriptors are outperformed by standard HOG descriptor by big margin, similarly to CAD-60 dataset.
- Brownian(9) which models 9 pixel-level features performs better that Brownian(7)
- Brownian(9) descriptor carries complementary information as fusion with HOG leads to improvement in recognition accuracy.
- The above conclusions are coherent with results on CAD-60 dataset.

Finally in table 6.4 we show the results of MBH+Brownian descriptor which is a fusion of MBH descriptor and Brownian descriptor. It is worth noting that in case of CAD-120 dataset analysis and fusions are done with MBH dataset, because it was best performing baseline descriptor.

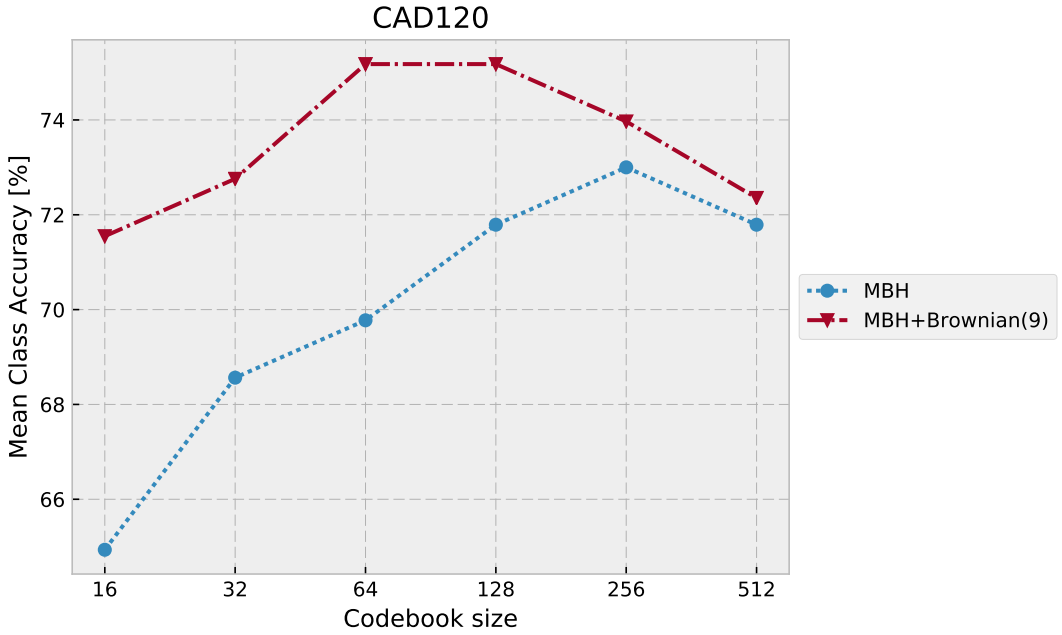
The results shows that further fusion of Brownian descriptor improves performance of standard MBH descriptor.

Table 6.3: Results on CAD-120 dataset, the plot and table show Mean Class Accuracy *w.r.t.* codebook size of standard HOG descriptor, Brownian descriptor and HOG+Brownian descriptor, which is a fusion of HOG and Brownian.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
HOG+Brownian(9)	54.58	5.70	56.60	1.35	50.95	1.71	54.98	6.27	52.16	7.79	53.77	6.08	56.60	1.35
HOG	50.00	2.28	54.84	1.14	51.61	1.14	48.79	2.85	51.21	5.13	46.37	3.99	54.84	1.14
Brownian(9)	38.31	5.13	40.73	3.99	43.15	0.57	42.74	1.71	43.15	1.14	40.32	3.42	43.15	0.57
Brownian(7)	38.31	1.71	39.92	1.71	41.53	0.57	41.13	1.71	42.74	2.28	38.31	1.14	42.74	2.28

Table 6.4: Results on CAD-120 dataset, the plot and table show Mean Class Accuracy *w.r.t.* codebook size of standard DT descriptor (TSD, HOG, HOF, MBH) and DT+Brownian which is a fusion of DT descriptors and Brownian.



	16 Acc [%]	σ	32 Acc [%]	σ	64 Acc [%]	σ	128 Acc [%]	σ	256 Acc [%]	σ	512 Acc [%]	σ	Best Acc [%]	σ
MBH+Brownian(9)	71.55	3.99	72.76	1.71	75.18	1.71	75.18	2.85	73.97	11.02	72.35	4.18	75.18	1.71
MBH	64.94	1.14	68.56	1.71	69.77	0.00	71.79	2.85	73.00	1.14	71.79	3.99	73.00	1.14

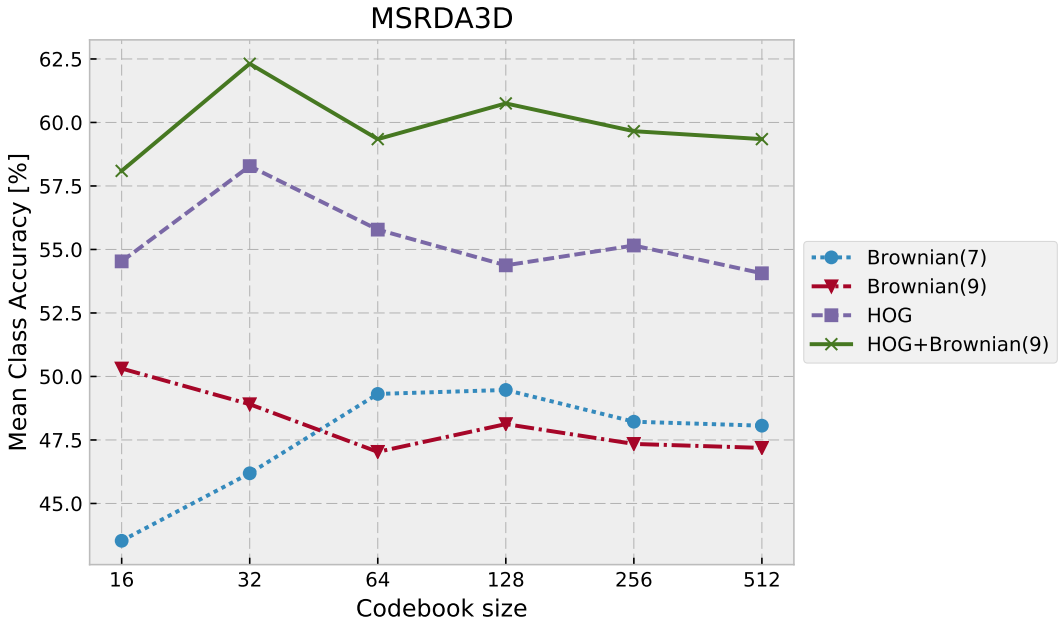
6.4.3 MSRDailyActivity3D

In table 6.5 we compare standard HOG descriptor to Brownian descriptor and to descriptor which is fusion of those two (HOG+Brownian(9)). Based on the experimental results on MSRDailyActivity3D dataset we can find the following conclusions:

- Brownian(7) and Brownian(9) as standalone descriptors are outperformed by standard HOG descriptor by big margin, similarly to the results obtained on previous datasets.
- Brownian(9) carries complementary information as fusion with HOG leads to recognition accuracy improvement.

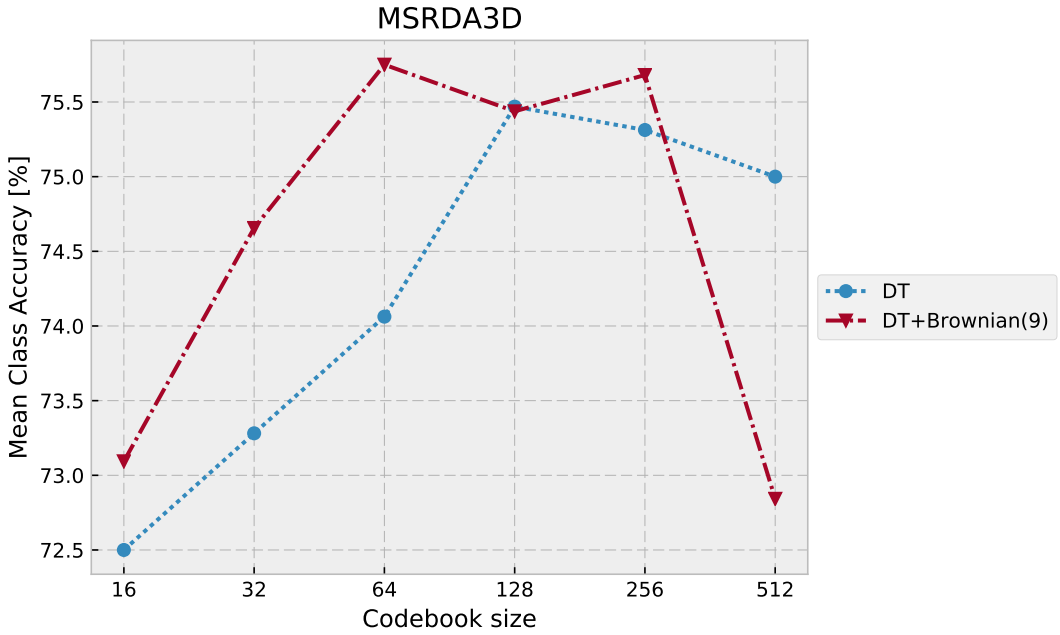
Finally in table 6.6 we show the results of DT+Brownian(9) descriptor which is a fusion of Dense Trajectories descriptors (TSD, HOG, HOF, MBH) and Brownian(9) descriptor. The results shows that further fusion of Brownian descriptor improves performance of standard DT descriptor, by a small margin, if we consider standard deviation of the results we can see that improvement of DT+Brownian(9) is not significant. The results show that beside the fact that HOG+Brownian(9) descriptor leads to improvement over HOG, unfortunately further fusion with DT does not lead to significant improvements. Brownian achieves its top results with small codebook size *eg.* 32, while DT performs better on bigger codebook size *eg.* 128. In our fusion strategy we combine representation with same codebook size, which in this case turned out to be not effective.

Table 6.5: Results on MSRDailyActivity3D dataset, the plot and table show Mean Class Accuracy *w.r.t.* codebook size of standard HOG descriptor, Brownian descriptor and HOG+Brownian descriptor, which is a fusion of HOG and Brownian.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
HOG+Brownian(9)	58.09	1.77	62.31	1.55	59.34	0.66	60.75	0.88	59.66	0.66	59.34	8.29	62.31	1.55
HOG	54.53	0.22	58.28	0.66	55.78	0.66	54.37	0.44	55.16	0.22	54.06	0.00	58.28	0.66
Brownian(9)	50.31	0.44	48.91	2.43	47.03	1.10	48.12	1.33	47.34	1.55	47.19	3.77	50.31	0.44
Brownian(7)	43.53	1.10	46.19	1.33	49.31	1.33	49.47	1.10	48.22	1.10	48.06	2.70	49.47	1.10

Table 6.6: Results on MSRDailyActivity3D dataset, the plot and table show Mean Class Accuracy *w.r.t.* codebook size of standard DT descriptor (TSD, HOG, HOF, MBH) and DT+Brownian which is a fusion of DT descriptors and Brownian.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
DT+Brownian(9)	73.09	1.55	74.66	0.66	75.75	1.10	75.44	0.66	75.68	0.35	72.84	0.2	75.75	1.10
DT	72.50	0.88	73.28	0.22	74.06	0.88	75.47	0.22	75.31	1.33	75.00	0.0	75.47	0.22

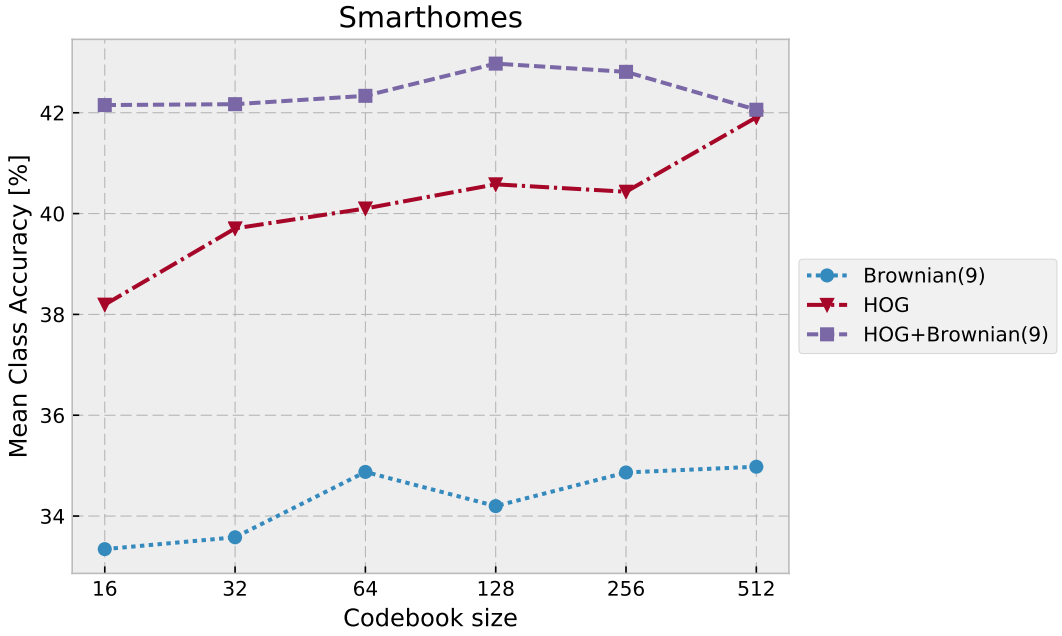
6.4.4 Smarthomes

In table 6.7 we compare standard HOG descriptor to Brownian(9) descriptor and to descriptor which is fusion of those two (HOG+Brownian(9)). The conclusion from experiments are similar to ones from other datasets:

- Brownian(9) as standalone descriptor is outperformed by standard HOG descriptor by big margin.
- Brownian(9) carries complementary information as fusion with HOG leads to recognition accuracy improvement.

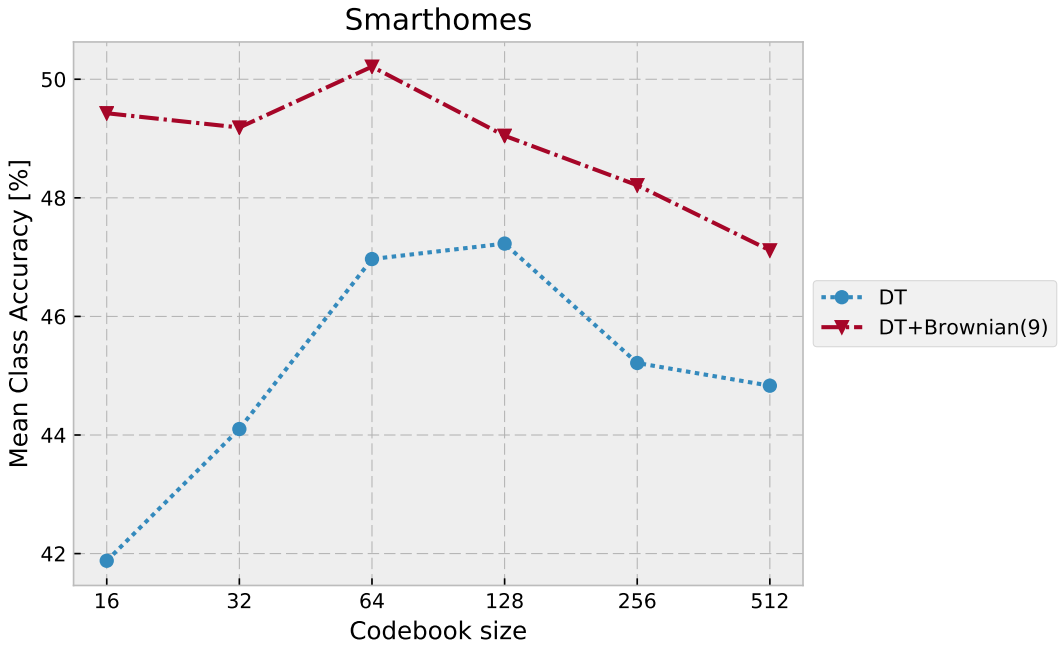
Finally in table 6.8 we show the results of DT+Brownian(9) descriptor which is a fusion of Dense Trajectories descriptors (TSD, HOG, HOF, MBH) and Brownian descriptor. The results shows that further fusion of Brownian descriptor improves performance of standard DT descriptor, by a bigger margin than for instance in MSRDailyActivity3D dataset. Fusion with Brownian(9) descriptor improved recognition of action like: "eat snack", "take pills", which were confused with "drinking" by standard DT descriptor. Brownian improves appearance model thus helps with recognition of actions with similar motion pattern.

Table 6.7: Results on Smarthomes dataset, the plot and table show Mean Class Accuracy *w.r.t.* codebook size of standard HOG descriptor, Brownian descriptor and HOG+Brownian descriptor, which is a fusion of HOG and Brownian.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
HOG+Brownian(9)	42.15	1.18	42.17	0.76	42.34	0.72	42.97	0.45	42.81	1.07	42.06	1.01	42.97	0.45
HOG	38.19	0.08	39.71	0.13	40.10	0.60	40.58	0.79	40.43	0.74	41.91	0.34	41.91	0.34
Brownian(9)	33.34	0.43	33.58	1.37	34.88	1.29	34.20	1.05	34.86	0.91	34.98	0.62	34.98	0.62

Table 6.8: Results on Smarthomes dataset, the plot and table show Mean Class Accuracy *w.r.t.* codebook size of standard DT descriptor (TSD, HOG, HOF, MBH) and DT+Brownian which is a fusion of DT descriptors and Brownian.



	16		32		64		128		256		512		Best	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
DT+Brownian(9)	49.43	0.11	49.19	0.06	50.21	1.46	49.05	1.22	48.21	0.06	47.11	1.94	50.21	1.46
DT	41.88	0.07	44.10	0.04	46.97	0.77	47.23	0.20	45.21	2.36	44.83	0.23	47.23	0.20

6.5 Conclusions

We presented a novel appearance-based descriptor for action recognition, that carries complementary information to the HOG descriptor. In contrast to the HOG (which directly models values of given features), the Brownian descriptor focuses on pairwise relations between features. The fusion of both descriptors gives an increase in performance. Our novel descriptor can be applied to all action recognition methods, which combine appearance-based and motion-based descriptors. Our experiments also shown that fusion strategy based on Fisher Vector concatenation is not always optimal, as fusion of Brownian descriptor with DT on MSRDailyActivity3D dataset did not lead to significant improvement, although fusion of HOG+Brownian significantly outperformed HOG descriptor. Apart from MSRDailyActivity3D dataset fusion of Brownian with DT descriptor achieved accuracy improvement on all other datasets.

Chapter 7

Modeling spatial layout with Fisher Vector and people detection

Contents

7.1 Introduction

If we analyze top performing methods on RGB-D and RGB videos, we can notice clear division. On RGB-D the most successful [116, 78] methods usually use skeleton information such as skeleton joint positions as an input, requiring quite sophisticated skeleton detection. The mentioned methods introduce implicit assumption that skeleton detection is

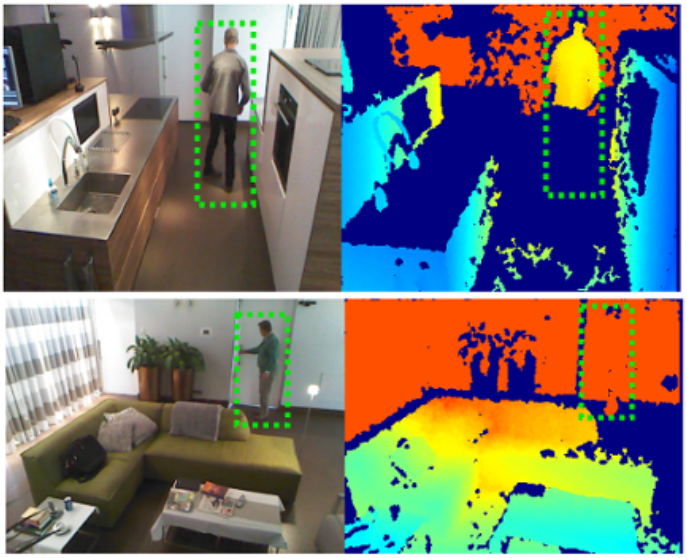


Figure 7.1: In this figure we present two examples where skeleton detection methods fail. Pictures on the left show RGB frame, pictures on the right show depth map (dark blue indicates missing depth information).

available and correct. We claim that in real-world scenario (*eg.* Smarthomes dataset) this assumption cannot be fulfilled (see fig. 7.1). Local features methods, which are successful on RGB [52, 113] are on the other side of extreme. They do not require any segmentation, relying on Points Of Interest detection based on low level features.

In this chapter we propose solution which meets in half way both mentioned above approaches. We propose to model spatial layout of local features *w.r.t.* person bounding box. Thanks to that we require much less error prone people detection, while being able to achieve comparable or even better recognition performance than methods which require skeleton detection.

7.2 Features spatial-layout encoding

In this section we describe in details 3 proposed different methods that encode spatial layout of local features. We assume that some Point of Interest (POI) detector is available and that we are cropping sub-region around each detected POI. Then we compute feature vector $\boldsymbol{f}$ for cropped sub-region. It can be any type of descriptor either hand-crafted *eg.* MBH, HOG (section 3.2) or deep-learning based: TDD features (section 3.3). Such combination of descriptors was widely used by many authors [52, 113], often with different encoding *eg.* Bag of Visual Words or Fisher Vectors. But the main drawback of such approaches is that spatial location of the feature $\boldsymbol{f}$ is lost. In the following sections we

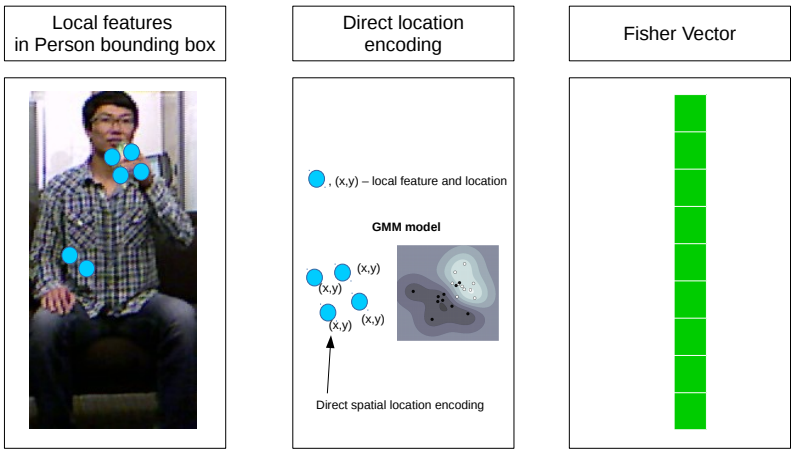


Figure 7.2: Direct spatial layout encoding. In this method we merge together descriptor computed in local neighbourhood of detected points of interest (blue points) with coordinates of point of interest *w.r.t.* to top-left corner of person bounding box. Such representation is then encoded with Fisher Vector.

propose 3 different methods which encodes the spatial information into Fisher Vector.

7.2.1 Direct spatial layout encoding

In this section we propose a method which directly encodes spatial location into feature vector. Let's define vector $\boldsymbol{l} = (x, y)$ which represents a location of detected POI *w.r.t.* top-left corner of person bounding box. To normalize vector $\boldsymbol{l}$ we divide it by width and height of the bounding box respectively. Thus if $\boldsymbol{l} = (0, 0)$ that means that feature location is in top left corner of bounding box, if $\boldsymbol{l} = (1, 1)$ the feature is in bottom right corner. If any $\boldsymbol{l}$ coordinate is either negative or bigger than one – that means that feature is outside of bounding box and is discarded.

Let's define $\boldsymbol{f_l}$ as a feature vector obtained from region around $\boldsymbol{l}$. To preserve spatial location information we propose to concatenate vectors $\boldsymbol{l}$ and $\boldsymbol{f_l}$ obtaining new feature vector $\boldsymbol{d} = (\boldsymbol{l}, \boldsymbol{f_l})$. Please note that we extended vector $\boldsymbol{f_l}$ by two dimensions. This is an advantage of propose encoding, because in most cases vector d is later used as an input to either clustering algorithms or classification algorithms which very often do not scale well with number of dimensions (curse of dimensionality). For instance let's assume that $\boldsymbol{f_l} \in \mathbb{R}^D$ and $\boldsymbol{l} \in \mathbb{R}^2$, then resulting Fisher Vector size for $\boldsymbol{f_l}$ equals to $2KD$, while Fisher Vector size for $\boldsymbol{d}$ equals $2K(D+2) = 2KD+4K$, where K is codebook size. The overview of proposed method is available in fig. 7.2.

The proposed concatenation method has also a downside – since spatial position $\boldsymbol{l}$ is

added to vector $\boldsymbol{f}$, it becomes indistinguishable in further processing. Intuitively our goal is to obtain homogeneous clusters or find the separation plane in terms of both spatial location and features values. But because spatial location and feature values are indistinguishable in $\boldsymbol{d}$, then information about different nature of encoded spatial location is lost.

Let's look at the k-means clustering example. Let's define k-means objective function as:

$$S^* = arg\min_S \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in S_i} \|\boldsymbol{x} - \boldsymbol{\mu_i}\|^2 \tag{7.1}$$

where $\boldsymbol{x}$ is a observation vector, S_i represents i-th cluster and $\boldsymbol{\mu_i}$ is a mean vector of observations $\boldsymbol{x}$ assigned to S_i. Now we can define cluster assignment for spatial location as:

$$S_l^* = arg\min_S \sum_{i=1}^{k} \sum_{\boldsymbol{l} \in S_i} \|\boldsymbol{l} - \boldsymbol{\mu_i^l}\|^2 \tag{7.2}$$

And cluster assignment for features:

$$S_f^* = arg\min_S \sum_{i=1}^{k} \sum_{\boldsymbol{f} \in S_i} \|\boldsymbol{f} - \boldsymbol{\mu_i^f}\|^2 \tag{7.3}$$

Note that each assignment finds homogeneous clusters in terms of spatial location and feature values respectively. Now if we define cluster assignment for proposed spatial location encoding $\boldsymbol{d}$ as:

$$S_d^* = arg\min_S \sum_{i=1}^{k} \sum_{\boldsymbol{d} \in S_i} \|\boldsymbol{d} - \boldsymbol{\mu_i^d}\|^2 \tag{7.4}$$

we will end-up with with cluster assignment S_d^* which is a compromise between assignments S_l^* and S_f^*, because:

$$\|\boldsymbol{d} - \boldsymbol{\mu_i^d}\|^2 \neq \|\boldsymbol{l} - \boldsymbol{\mu_i^l}\|^2 + \|\boldsymbol{f} - \boldsymbol{\mu_i^f}\|^2 \tag{7.5}$$

To solve the described limitation we need encoding method which would treat location information in a different way than feature information. In the following sections 7.2.2 and 7.2.4 we propose two methods which address the mentioned problem.

7.2.2 Grid Fisher Vector (GridFV)

In this section we propose to partition person bounding box into $r \times c$ spatial cells. And then represent each cell as Fisher Vector of it's features. It can be seen that method performs spatial clustering based on location $\boldsymbol{l}$ where clusters where selected arbitrarily (grid layout). After the clustering step we assign each feature $\boldsymbol{f_l}$ to grid cell (n, m), based on it's location $\boldsymbol{l}$. Then for each grid cell we compute Fisher Vector representation. Thus finally we will obtain $r * c$ Fisher Vectors – one Fisher Vector per grid cell. Final representation is concatenation of the Fisher Vectors, thus final Fisher Vector size is $(r * c) * (2KD)$.

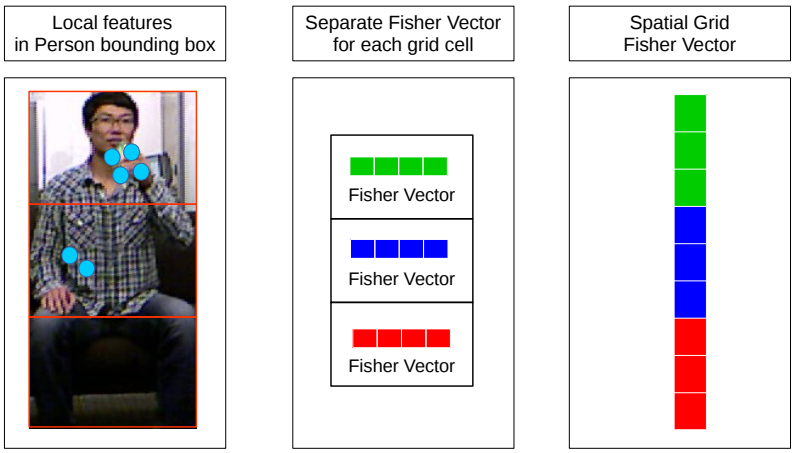


Figure 7.3: Grid Fisher Vector (GridFV). In first step person bounding box is divided int $r \times c$ grids (3×1 in this example). Then Fisher Vector representation is computed for each grid cell separately. Final GridFV representation is obtained by concatenating Fisher Vectors obtained in each grid cell.

All Fisher Vector representations are computed based on common GMM codebook computed on all features in bounding box. We chosen not to compute separate GMM codebook per grid cell to minimize the computational effort. In addition our experiments showed that computing separate GMM codebook per gird cell does not lead to performance improvements. Figure 7.3 shows overview of GridFV method.

The proposed method solves issues of direct encoding method proposed in previous section (7.2.1), as it treats spatial location information and feature information independently. On the other hand resulting Fisher Vector size is much bigger than in direct method. In next section we propose method which also clusters spatial location information and features information separately, but produces much more compact representation.

7.2.3 Fisher Vectors representation and it‘s relation to Mixture of Gaussians

In this section we first remind how to obtain Fisher Vector (FV) representation. The FV is very popular representation in image recognition and action recognition. More details can be found in chapter 3 and [81, 82]. The information provided in this section will be useful in understanding spatial-layout encoding method proposed in next sections.

The Fisher Vector representation is computed based on Gaussian Mixture Model (GMM), which consists of K Guassians where each Gaussian is described by three parameters $\alpha_k, \mu_k, \Sigma_k$ – mixture weight, mean and covariance (diagonal). The parameters

of GMM can be learned using Expectation Maximization (EM) algorithm. Let's assume that we have feature vector $\boldsymbol{f}$. We also define w as quantization index and k indicates k-th Gaussian. Then we can model:

$$p(w = k) = \pi_k \tag{7.6}$$

$$p(\boldsymbol{f}) = \sum_{k=1}^{K} \pi_k p(\boldsymbol{f}|w = k) \tag{7.7}$$

$$p(\boldsymbol{f}|w = k) = \mathcal{N}(\boldsymbol{f}; \mu_k, \Sigma_k), \tag{7.8}$$

where π_k is normalized mixing weight of k-th Gaussian using softmax: $\pi_k = \exp \alpha_k / \sum_j \exp \alpha_j$. Now we can define:

$$q_{nk} = p(w_n = k|\boldsymbol{f_n}) = \frac{p(\boldsymbol{f}|w = k)p(w = k)}{p(\boldsymbol{f})} \tag{7.9}$$

$$f_{nk} = x_n - \boldsymbol{\mu_{nk}} \tag{7.10}$$

where q_{nk} denotes a posterior. Based on definitions above we can define gradients of of log-likelihood for single feature f_n *w.r.t.* GMM model parameters as:

$$\frac{\partial \ln p(f_n)}{\partial \alpha_k} = q_{nk} - \pi_k, \tag{7.11}$$

$$\frac{\partial \ln p(f_n)}{\partial \boldsymbol{\mu_k}} = q_{nk} \boldsymbol{\Sigma}_k^{-1} f_{nk}, \tag{7.12}$$

$$\frac{\partial \ln p(f_n)}{\partial \boldsymbol{\Sigma}_k^{-1}} = \frac{q_{nk}(\boldsymbol{\Sigma}_k^{-1} - f_{nk}^2)}{2}, \tag{7.13}$$

To obtain Fisher Vector representation we normalize gradients by $\sqrt{\boldsymbol{F}}$, where $\boldsymbol{F} = \mathbb{E}[\boldsymbol{g(f)g(f)}^T]$ is Fisher information matrix. Where $\boldsymbol{g(f)}$ is gradient vector.

7.2.4 Spatial Mixture of Gaussians

In method proposed in this section we will model location of visual world to which feature $\boldsymbol{f}$ was assigned. We use formulation of joint probability for location encoding in Fisher Vector proposed by [47]. To encode spatial location of features we use relative coordinates *w.r.t.* top left corner of the bounding box. First let's define visual word w as cluster (Gaussian) id to which feature $\boldsymbol{f}$ was assigned (see 7.8).

$$w = arg \max_{k \in K} p(\boldsymbol{f}|w = k) \tag{7.14}$$

Note that at this point we perform hard assignment of feature $\boldsymbol{f}$ to visual word w, although we used GMM model. Now we can define a tuple $\boldsymbol{u} = (w, \boldsymbol{l})$, which bounds visual word of feature $\boldsymbol{f}$ with it's location $\boldsymbol{l} = (x, y)$.

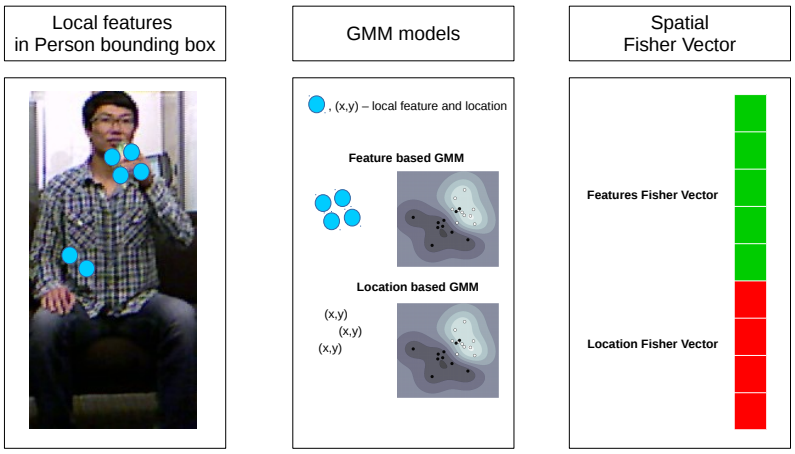


Figure 7.4: Spatial Mixture of Gaussians (SpatialMoG). In the first step local points of interest are detected within person bounding box. Separate Gaussian Mixture Models are computed for local descriptors and for location of local point of interest. Two Fisher Vector are computed: standard one based on local descriptors and second one which is based on features location.

7.2.4.1 Single Gaussian for spatial location encoding

In this paragraph we will model location $\boldsymbol{l}$ for each visual word $w = k, k = 1, 2, .., K$ with single Gaussian. Thus we will obtain K Gaussians for spatial location encoding, where each of them will model location of visual words assigned to k-th Gaussian. The Gaussians which model location can be trivially learned by computing the mean and variance of $\boldsymbol{l}$ assigned to k-th visual word. Now we define generative model over (visual word/location) tuple:

$$p(w = k) = \pi_k \tag{7.15}$$

$$p(\boldsymbol{l}|w = k) = \mathcal{N}(\boldsymbol{l}; \boldsymbol{m}_k, \boldsymbol{S}_k), \tag{7.16}$$

Based on definitions above we can define gradients of of log-likelihood for single feature f_n *w.r.t.* joint GMM parameters on $\boldsymbol{f}$ and single Gaussian based on $\boldsymbol{l}$:

$$\frac{\partial \ln p(u_n)}{\partial \alpha_k} = q_{nk} - \pi_k, \tag{7.17}$$

$$\frac{\partial \ln p(u_n)}{\partial \boldsymbol{m_k}} = q_{nk} \boldsymbol{S}_k^{-1} l_{nk}, \tag{7.18}$$

$$\frac{\partial \ln p(u_n)}{\partial \boldsymbol{S}_k^{-1}} = \frac{q_{nk}(\boldsymbol{S}_k^{-1} - l_{nk}^2)}{2}, \tag{7.19}$$

where $q_{nk} = 1$ if $w_n = k$ and $q_{nk} = 0$ otherwise. Based on gradients obtained above we can create Fisher Vector which encodes both feature and spatial location.

7.2.4.2 Mixture of Gaussian for spatial location encoding

In this paragraph we will show how to extend previously defined method, to model location with Mixture of C Gaussians. Please note that here we will deal with two separate GMM models - one computed based on features $\boldsymbol{f}$, second computed on tuple $\boldsymbol{u} = (w, \boldsymbol{l})$, which models spatial layout. The generative model over location, now can be redefined as:

$$p(\boldsymbol{u}) = \sum_{k=1}^{K} p(w = k) p(\boldsymbol{l}|w = k) \tag{7.20}$$

$$p(w = k) = \pi_k \tag{7.21}$$

$$p(\boldsymbol{l}|w = k) = \sum_{c=1}^{C} \theta_{kc} \mathcal{N}(\boldsymbol{l}; \boldsymbol{m}_{kc}, \boldsymbol{S}_{kc}), \tag{7.22}$$

Where $\theta_{kc} = exp\beta_{kc} / \sum_j exp\beta_{kj}$ is normalized mixing weight for k-th Gaussian in spatial model. Now we can define:

$$r_{nkc} = p(c|\boldsymbol{l_n}, w_n = k) = \frac{\theta_{kc} \mathcal{N}(\boldsymbol{l_n}; \boldsymbol{m}_{kc}, \boldsymbol{S}_{kc})}{p(\boldsymbol{l_n}|w_n = k)} \tag{7.23}$$

$$\boldsymbol{l_{nkc}} = \boldsymbol{l_n} - \boldsymbol{m_{kc}} \tag{7.24}$$

Based on definitions above we can define gradients of of log-likelihood for single feature f_n *w.r.t.* joint GMM parameters on $\boldsymbol{f}$ and GMM based on $\boldsymbol{l}$:

$$\frac{\partial \ln p(u_n)}{\partial \beta_{kc}} = q_{nk}(r_{nkc} - \theta_{kc}), \tag{7.25}$$

$$\frac{\partial \ln p(u_n)}{\partial \boldsymbol{m_k c}} = q_{nk} r_{knc} \boldsymbol{S}_{kc}^{-1} l_{nkc}, \tag{7.26}$$

$$\frac{\partial \ln p(u_n)}{\partial \boldsymbol{S}_{kc}^{-1}} = \frac{q_{nk} r_{knc} (\boldsymbol{S}_{kc}^{-1} - l_{nkc}^2)}{2}, \tag{7.27}$$

Based on gradients obtained above we can create Fisher Vector which encodes both feature and spatial location.

7.2.4.3 Final representation

The final representation is formed by concatenation of Fisher Vector which encodes only feature vector $\boldsymbol{f}$ (Eq. (7.12) - (7.13) with Fisher Vector which it's spatial location - (Eq.(7.26) - (7.27).

7.3 Extension to spatio-temporal layout modeling

So far in this chapter we described how to model spatial layout of local features. But our methods can be easily extended to model spatio-temporal layout of features.

Thus to extend "Direct spatial layout encoding" (section 7.2.1) to spatio-temporal case,

we simply redefine location vector $\boldsymbol{l}$, to $\boldsymbol{l} = (x, y, t)$, where t is a frame number in which given point of interest was detected.

In case of GridFV method (section 7.2.2) we first stack all detected person bounding boxes over the time, so that we form person bounding box volume. Then we partition person bounding box volume into $r \times c \times v$ spatio-temporal cells. We again redefine location vector to $\boldsymbol{l} = (x, y, t)$. And follow same Fisher Vector encoding strategy.

Similarly "Mixture of Gaussians spatial model" (section 7.2.4), can be extended to spatio-temporal case. In this case we again need to redefine location vector $\boldsymbol{l}$, to $\boldsymbol{l} = (x, y, t)$. Then the GMM for spatial encoding will become 3 dimensional.

7.4 Implementation details

7.4.1 Feature descriptors

The proposed above spatial layout encoding methods can be used together with different descriptors. We used Dense Trajectories to detect and compute descriptors. We decided to model spatio-temporal layout of HOG and MBH descriptors, because they are good representatives of motion and appearance features. For CNN features we selected TDD Spatial and TDD Temporal descriptors for spatio-temporal layout modeling. As we can see our method is generic and can be used with any features detector and any descriptor type.

7.4.2 People detection

In the proposed method we use people detector proposed by [104]. The mentioned detector use RGB and depth information. Such combination is beneficial since depth data is robust with respect to illumination changes, but sensitive to noise and low depth resolution far from sensor. RGB data on the other hand provides color and texture, but detector often fails under non-ideal illumination. In case of Smarthomes dataset we used Pose Machine method [59].

It is worth to mention, that our method can work with any people detector: either RBG-based [59, 90, 32]. Or depth-based [104]. All people detection methods mentioned above have their pros and cons, but we claim that they are more robust than skeleton detectors.

7.5 Experiments

In this section we will compare proposed three spatial-layout encoding methods on 4 datasets:

- CAD-60
- CAD-120
- MSRDailyActivity3D
- Smarthomes

The proposed spatial-layout encoding methods introduce hyper-parameters such as: number of rows r, columns c and temporal volumes v in spatio-temporal grid or numbers of Gaussians C in Mixture of Gaussian method. We provide experimental analysis for different values of hyper-parameters.

In our experiments we apply proposed method to appearance and motion features separately. We use two different appearance features: HOG (section 3.2) and "TDD Appearance" (section 3.3) descriptor. We also use two different motion features: MBH section 3.2 and "TDD Temporal" (section 3.3). All mentioned descriptors were computed in local neighbourhood of points detected with Dense Trajectories method (see section 3.2). We analyze also fusion of appearance and motion features representation.

In this section we use the following abbreviations:

- HOG($r \times c$) – SpatialFV (see section 7.2.2) computed on HOG descriptor.
- HOG($r \times c \times v$) – Spatio-TemporalFV (see section 7.2.2) computed on HOG descriptor.
- HOG($C = c$) – Spatial MoG modeling (c – number of Gaussians, see section 7.2.4) computed on HOG descriptor.
- HOG(DS) – direct spatial encoding (see section 7.2.1) on HOG descriptor.
- HOG(DST) – direct spatio-temporal encoding (see section 7.2.1) on HOG descriptor.
- MBH($r \times c$) – SpatialFV (see section 7.2.2) computed on MBH descriptor.
- MBH($r \times c \times v$) – Spatio-TemporalFV (see section 7.2.2) computed on MBH descriptor.
- MBH($C = c$) – Spatial MoG modeling (c – number of Gaussians, see section 7.2.4) computed on MBH descriptor.
- MBH(DS) – direct spatial encoding (see section 7.2.1) on MBH descriptor.
- MBH(DST) – direct spatio-temporal encoding (see section 7.2.1) on MBH descriptor.
- TDD Appearance($r \times c$) – SpatialFV (see section 7.2.2) computed on TDD Appearance descriptor.

- TDD Appearance($r \times c \times v$) – Spatio-TemporalFV (see section 7.2.2) computed on TDD Appearance descriptor.
- TDD Appearance($C = c$) – Spatial MoG modeling (c – number of Gaussians, see section 7.2.4) computed on TDD Appearance descriptor.
- TDD Appearance(DS) – direct spatial encoding (see section 7.2.1) on TDD Appearance descriptor.
- TDD Appearance(DST) – direct spatio-temporal encoding (see section 7.2.1) on TDD Appearance descriptor.
- TDD Temporal($r \times c$) – SpatialFV (see section 7.2.2) computed on TDD Temporal descriptor.
- TDD Temporal($r \times c \times v$) – Spatio-TemporalFV (see section 7.2.2) computed on TDD Temporal descriptor.
- TDD Temporal($C = c$) – Spatial MoG modeling (c – number of Gaussians, see section 7.2.4) computed on TDD Temporal descriptor.
- TDD Temporal(DS) – direct spatial encoding (see section 7.2.1) on TDD Temporal descriptor.
- TDD Temporal(DST) – direct spatio-temporal encoding (see section 7.2.1) on TDD Temporal descriptor.

In the following section we will discuss impact of spatio-temporal layout modeling on appearance features (HOG and TDD Appearance), motion features (MBH and TDD Temporal) and fusion of proposed methods. At the end in section 7.5.4 we will provide general conclusions based on all experiments.

7.5.1 Appearance descriptors

In this section we discuss impact of spatio-temporal layout modeling on appearance features. In tables from 7.1 to 7.4 and figures from 7.5 to 7.8 we provide results on HOG descriptor. In tables from 7.5 to 7.8 and figures from 7.9 to 7.12 we provide results of "TDD Appearance" descriptor.

The results show that modeling spatial layout and spatio-temporal layout of appearance features leads to significant improvement over standard descriptors. In most cases all proposed techniques outperform standard descriptors, while the best performance is obtained with GridFV method. The best performing method was GridFV with 3×1 layout on CAD-60, CAD-120 and Smarthomes dataset for both HOG and TDD Appearance

descriptors. 3×1 grid it is the most effective because it takes advantage of implicit semantic information (top grid belongs to head, bottom grid belongs to legs). 3×2 grid can encode information if action was done with right or left hand, but this information does not improve accuracy in most cases as same action can be done with either hand. 3×2 grid turned out to be the most effective for MSRDailyActivity3D dataset, but reported high standard deviation of the result. Spatio-temporal grids achieved worse results than spatial grids on all datasets (we explain the reason behind it in section 7.5.4).

Direct encoding of features was second best performing method on smaller dataset such as: CAD-60, CAD-120, MSRDailyActivity3D, while on Smarthomes dataset they managed only to slightly outperform baseline descriptor. This conclusion was expected, because as we explained in section 7.2.1 direct encoding method is limited in terms of features clustering.

Spatial MoG is underperforming on CAD-60 dataset, while on CAD-120 and MSRDailyActivity3D MoG with $C = 6$ and $C = 3$ were able to beat the baselines. Interestingly Spatial MoG achieves good results on Smarthomes dataset. Analysis of learning process shows that Spatial MoG tends to overfit on small dataset, even reducing number C of Gaussians does not help to solve that problem. The experiments show that more data helps to find better Spatial MoG model, but further analysis is required to find exact reason of overfitting problem on small datasets.

7.5.2 Motion descriptors

In this section we discuss impact of spatio-temporal layout modeling on motion features. In tables from 7.9 to 7.12 and figures from 7.13 to 7.16 we provide results on MBH descriptor. In tables from 7.13 to 7.16 and figures from 7.17 to 7.20 we provide results of "TDD Temporal" descriptor.

The results are pretty aligned with ones for appearance features, but because motion features play key role in action recognition, encoding their spatial layout to Fisher Vector leads to even better accuracy improvement. Again GridFV was the best performing method, 3×1 layout achieved the best results on CAD-60, CAD-120 and Smarthomes dataset. Same layout was the most effective for appearance descriptors. And similarly on MSRDailyActivity3D the best performing layout was 3×2 exactly as in appearance descriptors. Spatio-temporal volumes were not effective also for motion features – we explain this fact in details in section 7.5.4.

Also similarly to appearance descriptors direct encoding was second best method, while Spatial MoG performed worse on small datasets: CAD-60 dataset, CAD-120 and MSRDailyActivity3D. For Smarthomes dataset Spatial MoG $C = 6$ performed almost as good SpatialFV. It is worth to notice that results of Spatial MoG methods report slightly

higher standard deviation on Smarthomes dataset. We think that it is due to the fact that in this case there are two GMM models, which result depends on random initialization.

7.5.3 Descriptors fusion

In this section we report result of fusion of appearance and motion features with propose GridFV encoding. For CAD60 we fuse DT descriptor with GridFV descriptors HOG(3×1) and MBH(3×1), the results in table 7.17 and fig. 7.21 show that the proposed fusion outperforms standard DT descriptor. Also fusion of TDD descriptor with GridFV encoding leads to further accuracy improvement.

On CAD120 we fuse standard MBH descriptor with GridFV descriptors HOG(3×1) and MBH(3×1), instead of DT because MBH was the best the performing baseline. Results in table 7.18 and fig. 7.22 show that the proposed fusion outperforms standard MBH descriptor. Also fusion of TDD descriptor with GridFV encoding leads to significant accuracy improvement.

For MSRDailyActivity3D dataset we fuse standrd DT descriptor with HOG(3×2) and MBH(3×2), the result of fusion are available in table 7.19 and fig. 7.23, the results of TDD descriptors fusion are available in table 7.23 and fig. 7.27. In both cases fusion improved recognition accuracy.

For Smarthomes data set fusions with DT descriptor showed significant improvement (fig. 7.24 and fig. 7.28), especially if we consider size of the dataset and number of classes.

Finally we conclude that feature fusion is effective in all cases and lead to accuracy improvements.

7.5.4 Experiments Summary

The experiments show that proposed methods of modeling spatio-temporal layout of features outperform with the big margin standard descriptors. The best strategy is to use spatial grid with either 3×1 or 3×2 layout. Spatio-temporal grids (*eg.* $3 \times 2 \times 2$) report smaller performance gain. We claim that spatial grids implicitly encodes information about person posture (top grids belongs to head, bottom grids belongs to legs). This is true in most cases, as long as person does not perform action upside down or in lying position. This property allows spatial grids to achieve best performance. In case of spatio-temporal grids further division into 2 or 3 temporal volumes does not lead to further accuracy improvement. The temporal domain is less structured than spatial layout, thus actions cannot be easily decomposed into 2 or 3 phases.

Modeling spatio-temporal layout with MoG performs worse than grid methods. MoG method can be seen as extension of grid method with irregular grids, which are inferred from data. The analysis of training process shows that Spatial MoG is overfitting on small

datasets. On small datasets Spatial MoG is able to find detailed fine grained relationships between codewords and their location, which does not generalize to unseen data. In case of Smarthomes dataset data itself works as regularization method, thus results obtained on Smarthomes dataset achieve much better accuracy.

Finally direct location encoding showed better performance on small datasest and in most cases managed to outperform baseline descriptor. It is worth to notice, that direct encoding method does not introduce additional parameters, which is an advantage.

Table 7.1: Modeling spatio-temporal layout of appearance features: results on HOG descriptor and CAD-60 dataset. Modeling spatial layout with grid: 3 columns by 1 rows, leads to significant improvement over standard HOG.

	Acc [%]	σ
HOG(3x1)	66.96	3.79
HOG(3x1x2)	62.50	3.79
HOG(3x1x3)	61.61	3.79
HOG(DST)	61.61	2.53
HOG(3x2)	59.82	3.79
HOG(DS)	58.93	2.53
HOG(3x2x2)	58.93	2.53
HOG(3x2x3)	53.57	2.53
Standard HOG	52.68	3.79
HOG(C=6)	41.96	1.26
HOG(C=3)	41.96	1.26
HOG(C=1)	29.76	5.05

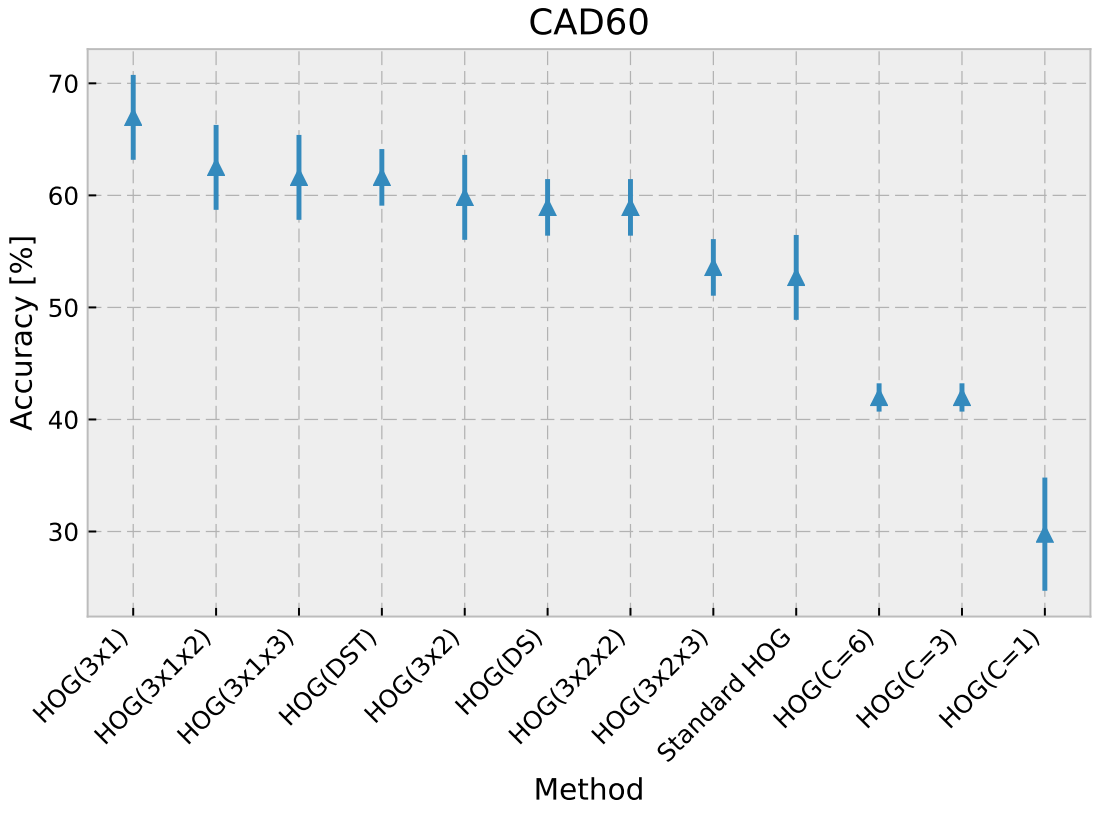


Figure 7.5: Comparison of proposed methods which model spatio-temporal layout on HOG descriptor and CAD-60 dataset. Length of lines indicate standard deviation.

Table 7.2: Modeling spatio-temporal layout of appearance features: results on HOG descriptor and CAD-120 dataset. Modeling spatial layout with grid: 3 columns by 2 rows, leads to significant improvement over standard HOG.

	Acc [%]	σ
HOG(3x2)	70.05	1.14
HOG(3x1)	69.24	0.57
HOG(3x2x2)	66.35	2.85
HOG(3x1x2)	66.13	1.71
HOG(3x2x3)	65.95	1.71
HOG(3x1x3)	65.32	1.14
HOG(C=6)	64.78	2.66
HOG(DST)	61.69	2.47
HOG(DS)	59.68	3.42
Standard HOG	54.84	1.14
HOG(C=3)	52.82	0.57
HOG(C=1)	52.02	3.99

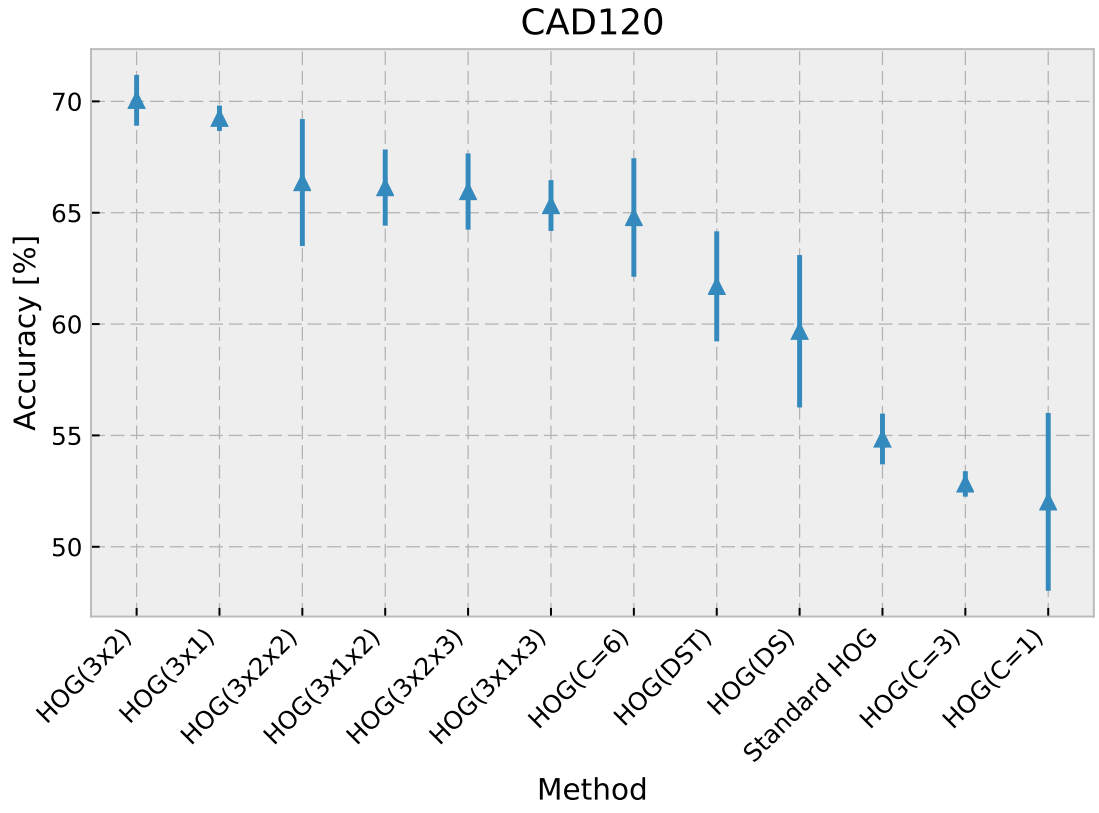


Figure 7.6: Comparison of proposed methods which model spatio-temporal layout on HOG descriptor and CAD-120 dataset. Length of lines indicate standard deviation.

Table 7.3: Modeling spatio-temporal layout of appearance features: results on HOG descriptor and MSRDailyActivity3D dataset. Modeling spatial layout with grid: 3 columns by 2 rows, leads to significant improvement over standard HOG.

	Acc [%]	σ
HOG(3x2)	75.78	6.08
HOG(3x1)	72.12	0.00
HOG(3x1x3)	71.88	4.42
HOG(3x1x2)	70.90	1.93
HOG(3x2x3)	70.12	5.89
HOG(3x2x2)	67.19	0.88
HOG(DST)	61.72	1.55
HOG(DS)	61.56	0.88
HOG(C=3)	59.22	0.77
Standard HOG	58.28	0.66
HOG(C=6)	57.97	0.22
HOG(C=1)	54.37	0.44

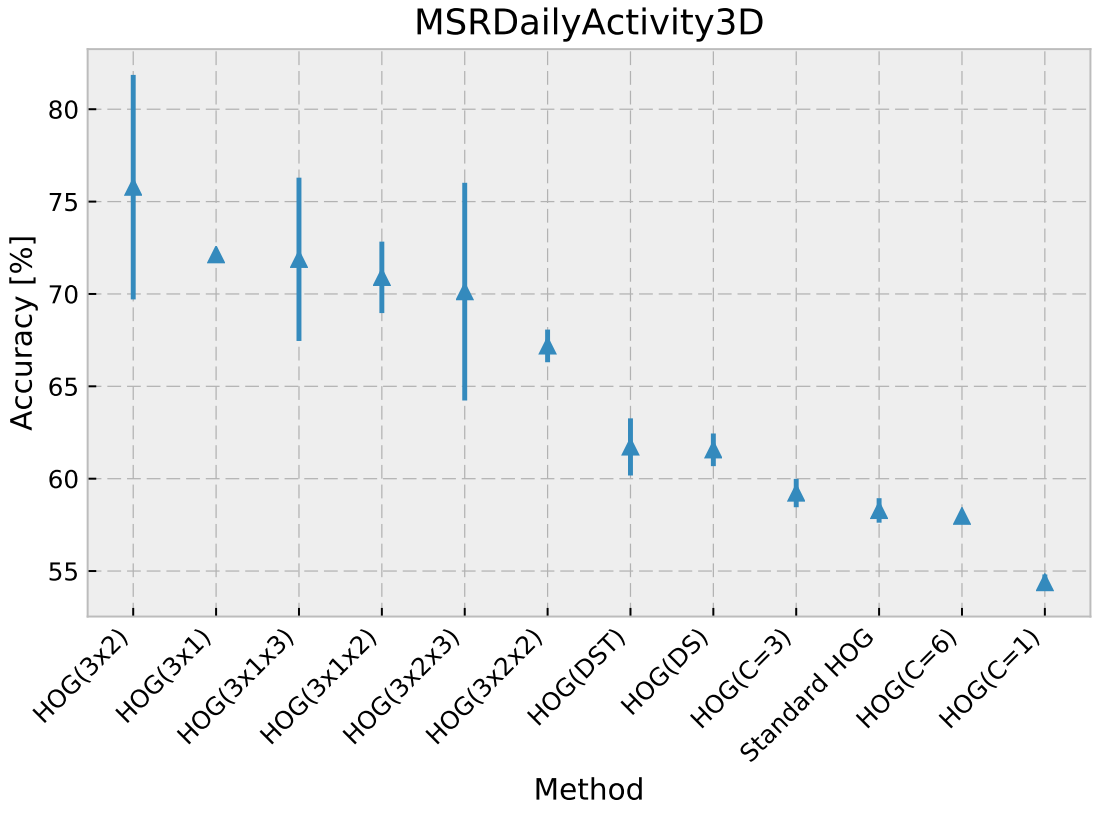


Figure 7.7: Comparison of proposed methods which model spatio-temporal layout on HOG descriptor and MSRDailyActivity3D dataset. Length of lines indicate standard deviation.

Table 7.4: Modeling spatio-temporal layout of appearance features: results on HOG descriptor and Smarthomes dataset. Modeling spatial layout with grid: 3 columns by 1 rows, leads to significant improvement over standard HOG.

	Acc [%]	σ
HOG(3x1)	43.99	0.05
HOG(3x2)	43.52	1.88
HOG(C=6)	43.16	1.63
HOG(C=3)	43.10	1.37
HOG(C=1)	42.89	0.99
HOG(3x1x2)	42.82	0.03
HOG(3x2x2)	42.05	0.27
HOG(DS)	41.92	0.40
Standard HOG	41.91	0.34
HOG(DST)	40.77	0.86
HOG(3x1x3)	39.05	0.01
HOG(3x2x3)	37.16	0.02

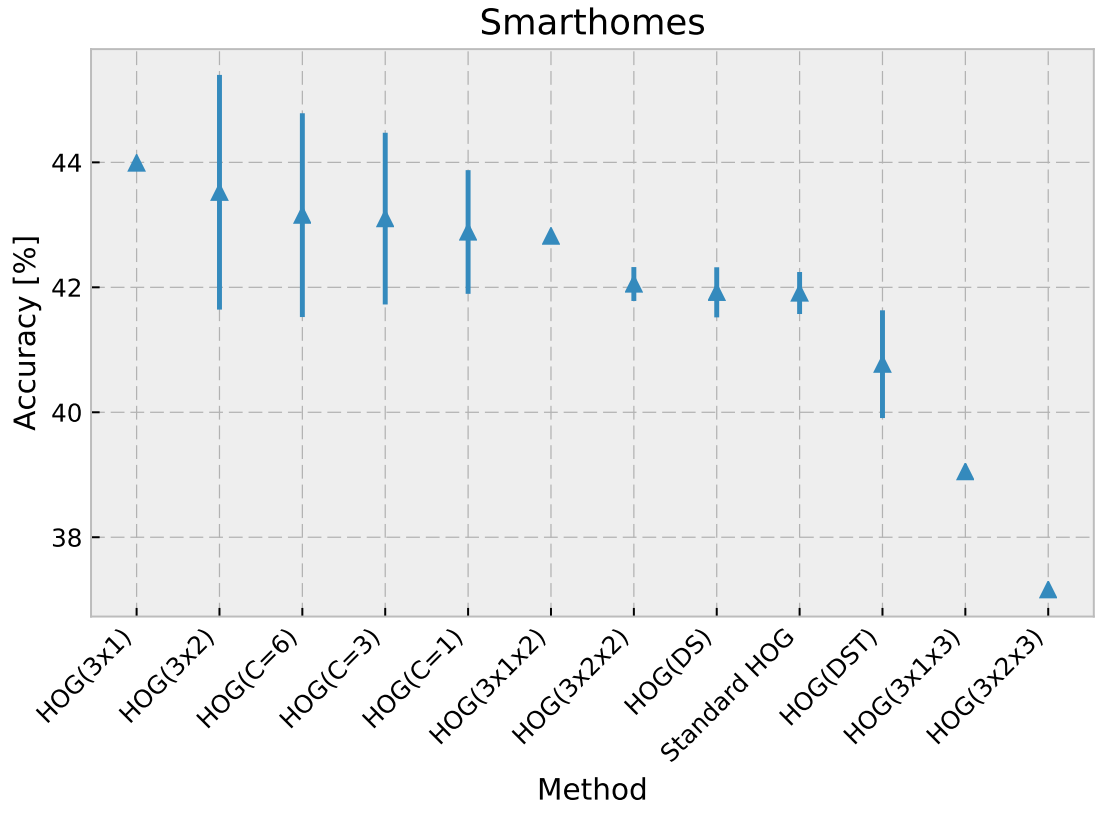


Figure 7.8: Comparison of proposed methods which model spatio-temporal layout on HOG descriptor and Smarthomes dataset. Length of lines indicate standard deviation.

Table 7.5: Modeling spatio-temporal layout of appearance features: results on "TDD Appearance" descriptor and CAD-60 dataset. Modeling spatial layout with grid: 3 columns by 1 rows, leads to significant improvement over standard "TDD Appearance".

	Acc [%]	σ
TDD Appearance(3x1)	70.54	3.79
TDD Appearance(3x1x2)	67.86	0.00
TDD Appearance(3x1x3)	66.07	0.00
TDD Appearance(DST)	64.29	0.00
TDD Appearance(DS)	64.29	2.53
TDD Appearance(3x2)	62.50	0.00
TDD Appearance(3x2x2)	60.29	1.26
TDD Appearance(C=6)	59.82	1.26
TDD Appearance	56.25	3.79
TDD Appearance(C=3)	56.25	1.26
TDD Appearance(3x2x3)	51.79	0.00
TDD Appearance(C=1)	41.67	5.05

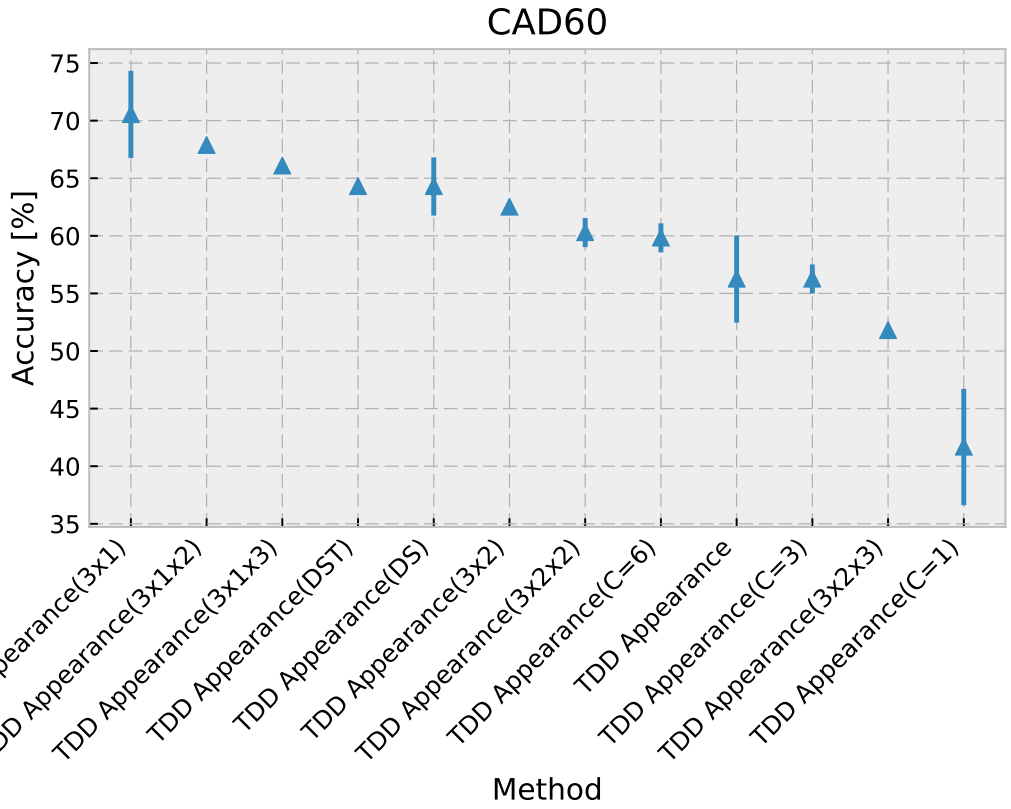


Figure 7.9: Comparison of proposed methods which model spatio-temporal layout on "TDD Appearance" descriptor and CAD-60 dataset. Length of lines indicate standard deviation.

Table 7.6: Modeling spatio-temporal layout of appearance features: results on "TDD Appearance" descriptor and CAD-120 dataset. Modeling spatial layout with grid: 3 columns by 1 rows, leads to significant improvement over standard "TDD Appearance".

	Acc [%]	σ
TDD Appearance(3x1)	87.77	2.09
TDD Appearance(3x2)	83.87	1.14
TDD Appearance(3x1x3)	83.02	3.04
TDD Appearance(3x1x2)	82.89	0.57
TDD Appearance(3x2x3)	81.85	1.71
TDD Appearance(3x2x2)	81.68	0.00
TDD Appearance(DS)	74.60	1.71
TDD Appearance	69.76	3.99
TDD Appearance(DST)	68.55	4.56
TDD Appearance(C=6)	67.61	1.33
TDD Appearance(C=3)	66.94	1.14
TDD Appearance(C=1)	64.11	0.57

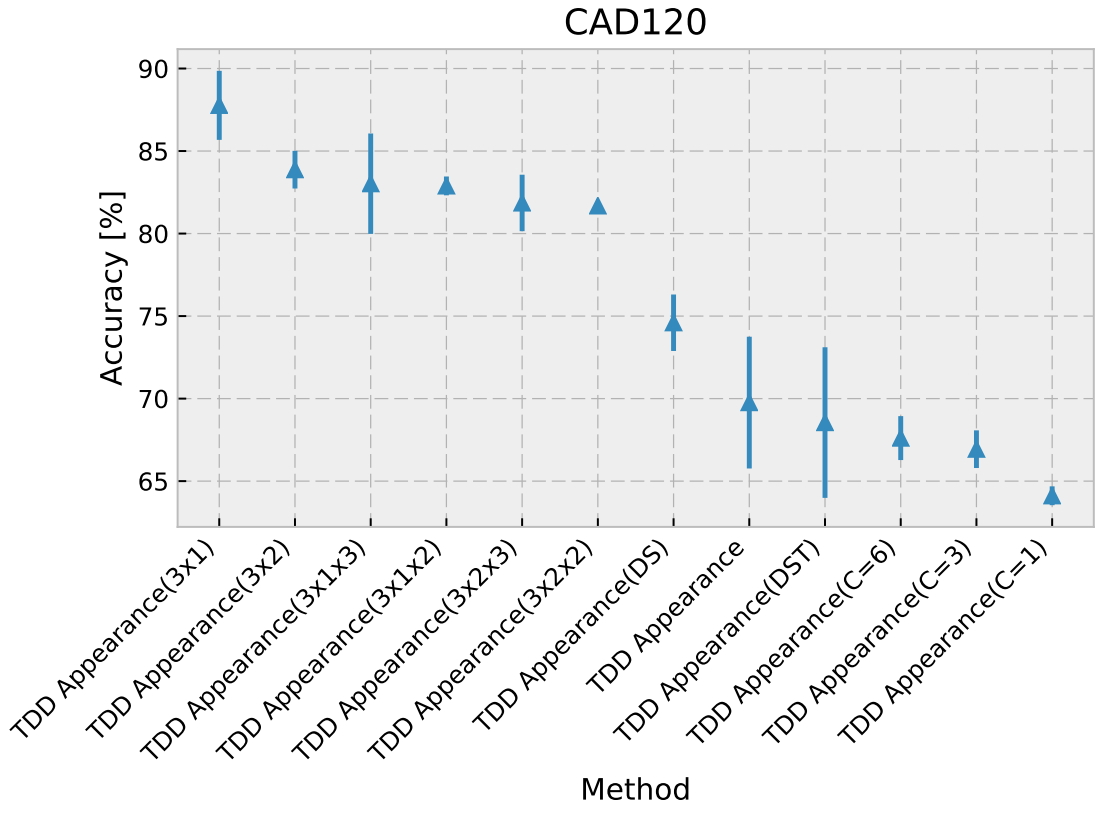


Figure 7.10: Comparison of proposed methods which model spatio-temporal layout on "TDD Appearance" descriptor and CAD-120 dataset. Length of lines indicate standard deviation.

Table 7.7: Modeling spatio-temporal layout of appearance features: results on "TDD Appearance" descriptor and MSRDailyActivity3D dataset. Modeling spatial layout with grid: 3 columns by 2 rows, leads to significant improvement over standard "TDD Appearance".

	Acc [%]	σ
TDD Appearance(3x2)	82.81	6.63
TDD Appearance(3x1x2)	82.29	1.47
TDD Appearance(3x2x2)	82.03	1.10
TDD Appearance(3x1x3)	81.25	3.31
TDD Appearance(3x2x3)	80.75	0.23
TDD Appearance(3x1)	78.40	1.50
TDD Appearance(C=3)	64.95	1.50
TDD Appearance(DS)	64.82	0.48
TDD Appearance	64.38	0.44
TDD Appearance(C=6)	62.97	0.66
TDD Appearance(C=1)	62.81	3.09
TDD Appearance(DST)	62.21	0.44

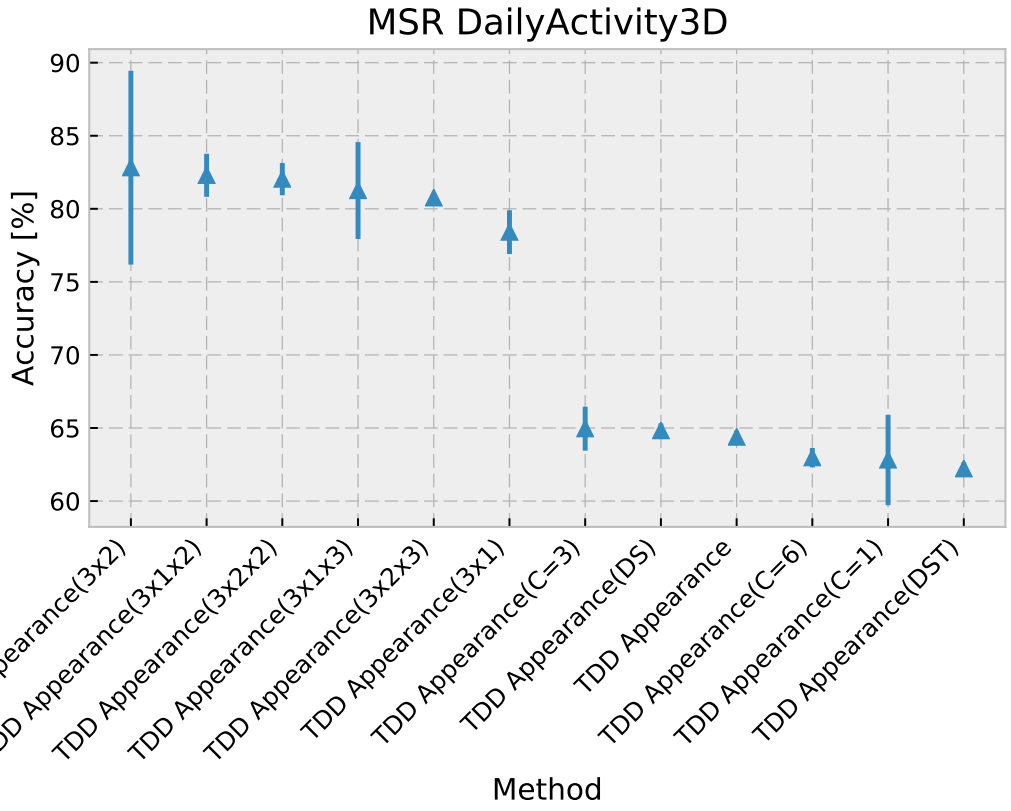


Figure 7.11: Comparison of proposed methods which model spatio-temporal layout on "TDD Appearance" descriptor and MSRDailyActivity3D dataset. Length of lines indicate standard deviation.

Table 7.8: Modeling spatio-temporal layout of appearance features: results on "TDD Appearance" descriptor and Smarthomes dataset. Modeling spatial layout with grid: 3 columns by 1 rows, leads to significant improvement over standard "TDD Appearance".

	Acc [%]	σ
TDD Appearance(3x1)	33.18	0.77
TDD Appearance(C=3)	31.91	0.32
TDD Appearance(3x2)	31.45	0.17
TDD Appearance(C=6)	31.42	0.60
TDD Appearance(DS)	30.58	0.77
TDD Appearance	29.16	0.52
TDD Appearance(3x1x2)	28.12	0.96
TDD Appearance(3x2x2)	27.32	0.06
TDD Appearance(C=1)	26.83	1.87
TDD Appearance(DST)	26.31	0.77
TDD Appearance(3x1x3)	26.19	0.35
TDD Appearance(3x2x3)	25.17	0.38

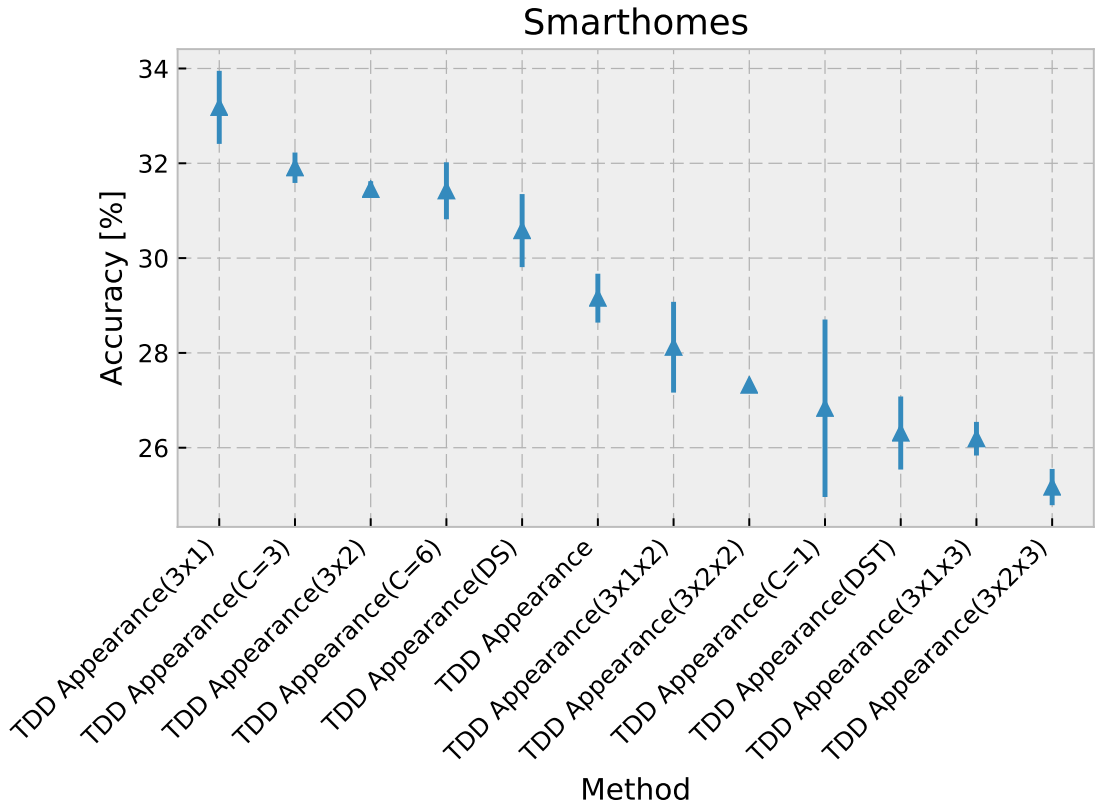


Figure 7.12: Comparison of proposed methods which model spatio-temporal layout on "TDD Appearance" descriptor and Smarthomes dataset. Length of lines indicate standard deviation.

Table 7.9: Modeling spatio-temporal layout of motion features: results on MBH descriptor and CAD-60 dataset. Modeling spatial layout with grid: 3 columns by 1 rows, leads to significant improvement over standard MBH.

	Acc [%]	σ
MBH(3x1)	72.18	3.79
MBH(DST)	71.43	2.53
MBH(DS)	70.71	2.53
MBH(3x2)	68.82	2.53
MBH(3x1x2)	66.96	1.26
MBH(C=6)	62.82	1.26
Standard MBH	61.61	1.26
MBH(C=3)	61.36	0.00
MBH(3x1x3)	58.04	3.79
MBH(3x2x2)	56.25	1.26
MBH(C=1)	42.67	1.68
MBH(3x2x3)	41.07	2.53

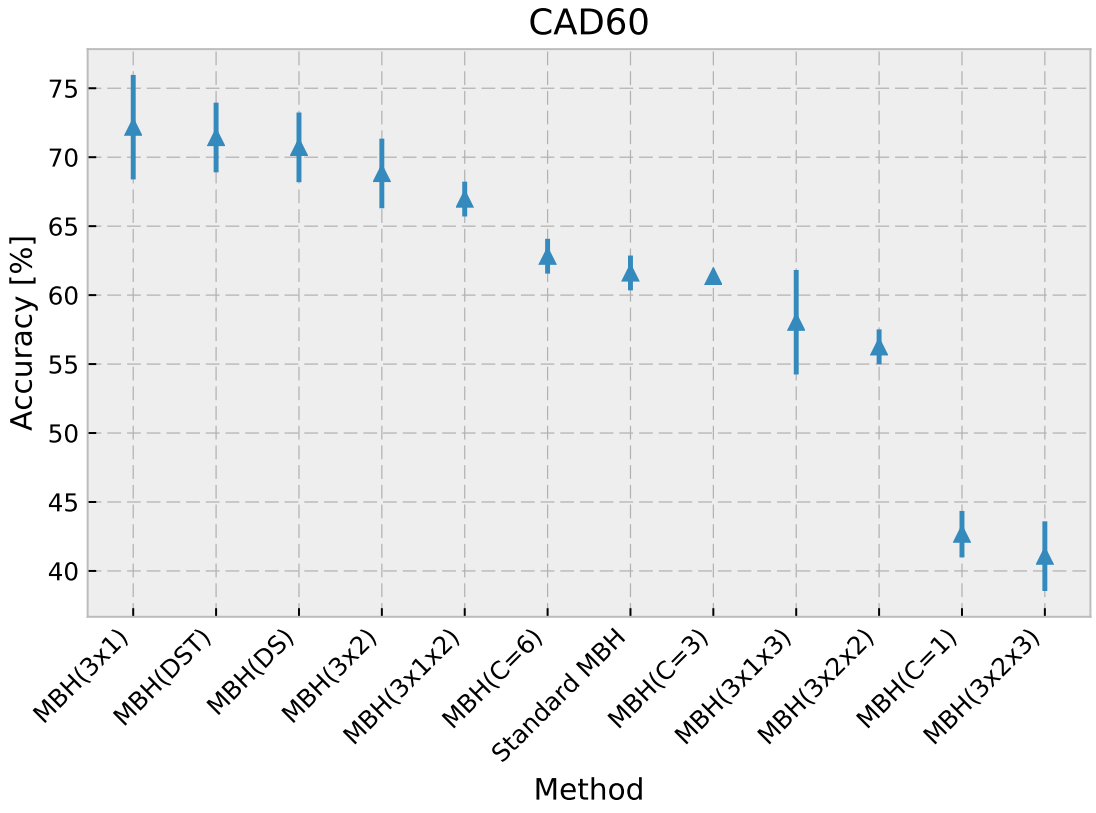


Figure 7.13: Comparison of proposed methods which model spatio-temporal layout on MBH descriptor and CAD-60 dataset. Length of lines indicate standard deviation.

Table 7.10: Modeling spatio-temporal layout of motion features: results on MBH descriptor and CAD-120 dataset. Modeling spatial layout with grid: 3 columns by 1 rows, leads to significant improvement over standard MBH.

	Acc [%]	σ
MBH(3x1)	82.26	1.14
MBH(3x2)	79.44	1.71
MBH(DST)	78.23	1.71
MBH(3x1x3)	77.02	2.28
MBH(DS)	76.79	1.71
MBH(3x1x2)	76.61	2.28
MBH(C=6)	73.79	0.57
MBH(3x2x2)	73.39	2.28
Standard MBH	73.00	1.14
MBH(3x2x3)	71.77	1.14
MBH(C=1)	66.69	1.71
MBH(C=3)	66.31	2.85

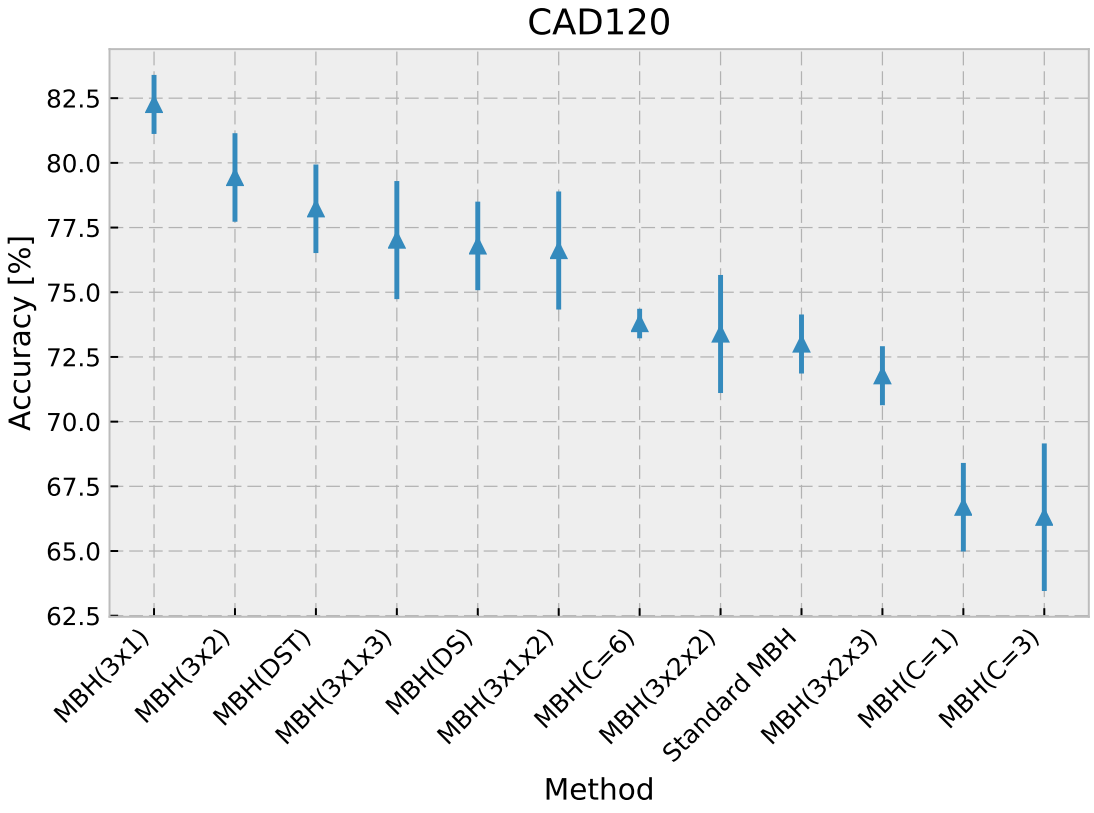


Figure 7.14: Comparison of proposed methods which model spatio-temporal layout on MBH descriptor and CAD-120 dataset. Length of lines indicate standard deviation.

Table 7.11: Modeling spatio-temporal layout of motion features: results on MBH descriptor and MSRDailyActivity3D dataset. Modeling spatial layout with grid: 3 columns by 2 rows, leads to significant improvement over standard MBH.

	Acc [%]	σ
MBH(3x2)	81.09	0.66
MBH(3x2x3)	80.59	1.10
MBH(3x1)	80.31	0.44
MBH(3x2x2)	80.00	0.00
MBH(3x1x3)	80.00	3.87
MBH(3x1x2)	79.84	2.58
MBH(DS)	77.73	0.55
MBH(DST)	77.03	0.22
MBH(C=6)	76.72	0.22
MBH(C=3)	75.72	0.15
MBH(C=1)	74.81	0.44
Standard MBH	74.38	0.00

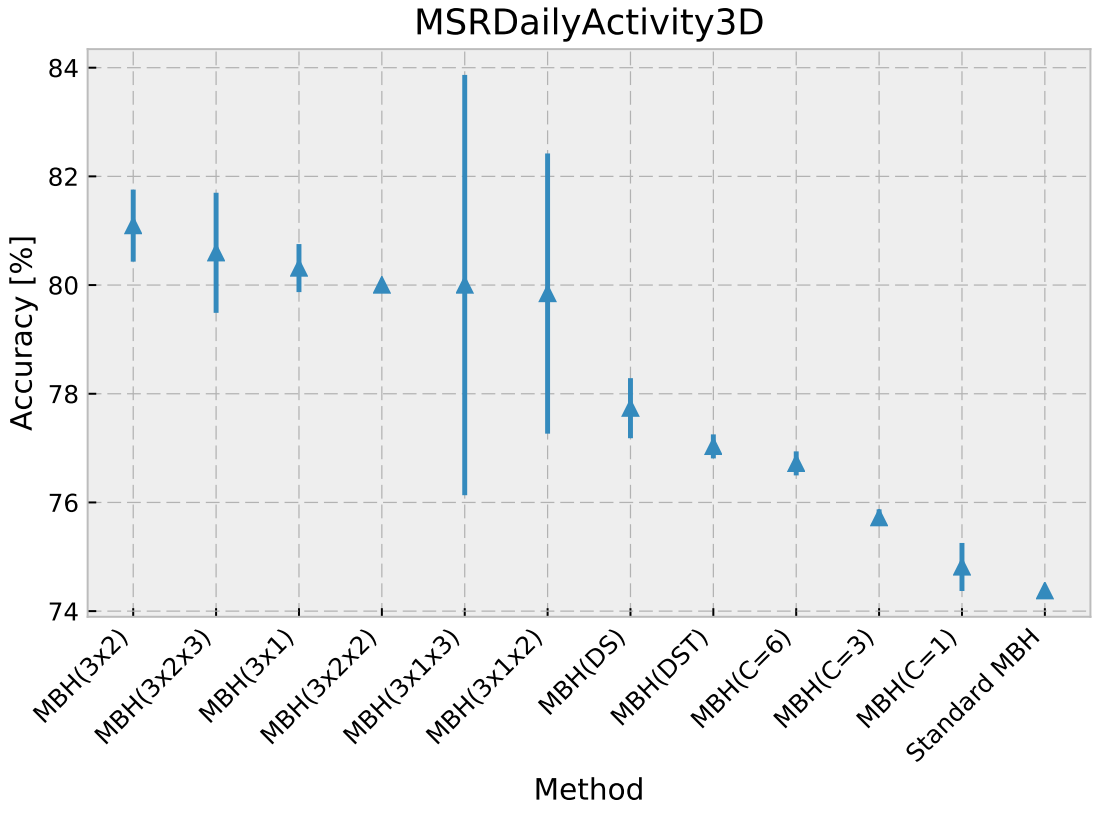


Figure 7.15: Comparison of proposed methods which model spatio-temporal layout on MBH descriptor and MSRDailyActivity3D dataset. Length of lines indicate standard deviation.

Table 7.12: Modeling spatio-temporal layout of motion features: results on MBH descriptor and Smarthomes dataset. Modeling spatial layout with grid: 3 columns by 1 rows, leads to significant improvement over standard MBH.

	Acc [%]	σ
MBH(3x1)	47.44	0.19
MBH(3x2)	46.30	1.14
MBH(C=6)	45.90	0.66
MBH(C=3)	45.65	0.87
MBH(C=1)	45.61	0.10
MBH(DS)	45.37	0.83
Standard MBH	44.96	0.07
MBH(DST)	44.23	0.20
MBH(3x1x2)	43.99	0.26
MBH(3x2x2)	43.87	0.35
MBH(3x1x3)	41.21	0.59
MBH(3x2x3)	40.03	0.08

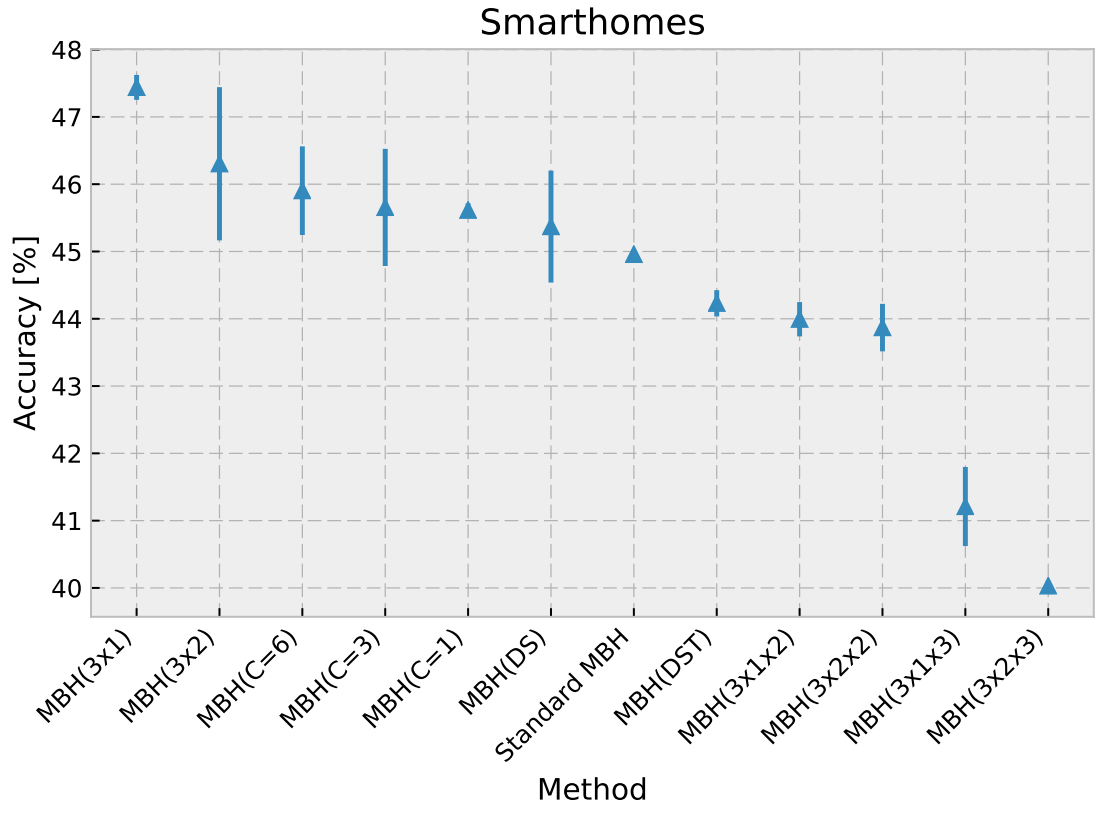


Figure 7.16: Comparison of proposed methods which model spatio-temporal layout on MBH descriptor and Smarthomes dataset. Length of lines indicate standard deviation.

Table 7.13: Modeling spatio-temporal layout of motion features: results on "TDD Temporal" descriptor and CAD-60 dataset. Modeling spatial layout with grid: 3 columns by 1 rows, leads to significant improvement over standard "TDD Temporal".

	Acc [%]	σ
TDD Temporal(3x1)	75.89	1.26
TDD Temporal(3x1x2)	68.75	1.26
TDD Temporal(3x1x3)	66.54	1.26
TDD Temporal(DST)	66.07	2.53
TDD Temporal(3x2)	66.07	0.00
TDD Temporal(DS)	65.18	3.79
TDD Temporal	63.39	1.26
TDD Temporal(3x2x2)	62.50	2.53
TDD Temporal(3x2x3)	61.61	1.26
TDD Temporal(C=6)	60.71	0.00
TDD Temporal(C=3)	57.14	2.53
TDD Temporal(C=1)	48.81	1.68

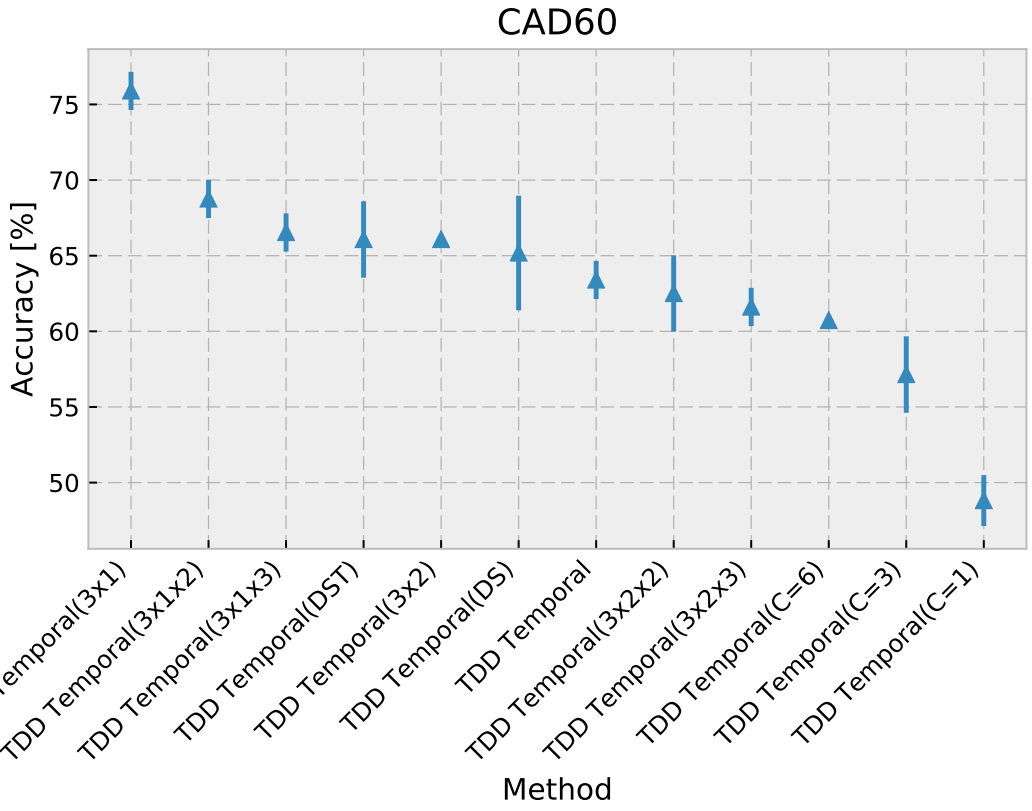


Figure 7.17: Comparison of proposed methods which model spatio-temporal layout on "TDD Temporal" descriptor and CAD-60 dataset. Length of lines indicate standard deviation.

Table 7.14: Modeling spatio-temporal layout of motion features: results on "TDD Temporal" descriptor and CAD-120 dataset. Modeling spatial layout with grid: 3 columns by 1 rows, leads to significant improvement over standard "TDD Temporal".

	Acc [%]	σ
TDD Temporal(3x1)	81.85	0.57
TDD Temporal(3x2)	80.38	1.90
TDD Temporal(DS)	79.84	1.14
TDD Temporal(DST)	79.66	0.57
TDD Temporal(3x1x3)	79.03	2.28
TDD Temporal(3x1x2)	78.49	1.52
TDD Temporal(3x2x2)	73.79	2.85
TDD Temporal(3x2x3)	73.12	3.04
TDD Temporal	72.98	1.71
TDD Temporal(C=6)	72.98	1.71
TDD Temporal(C=3)	70.16	1.14
TDD Temporal(C=1)	65.73	0.57

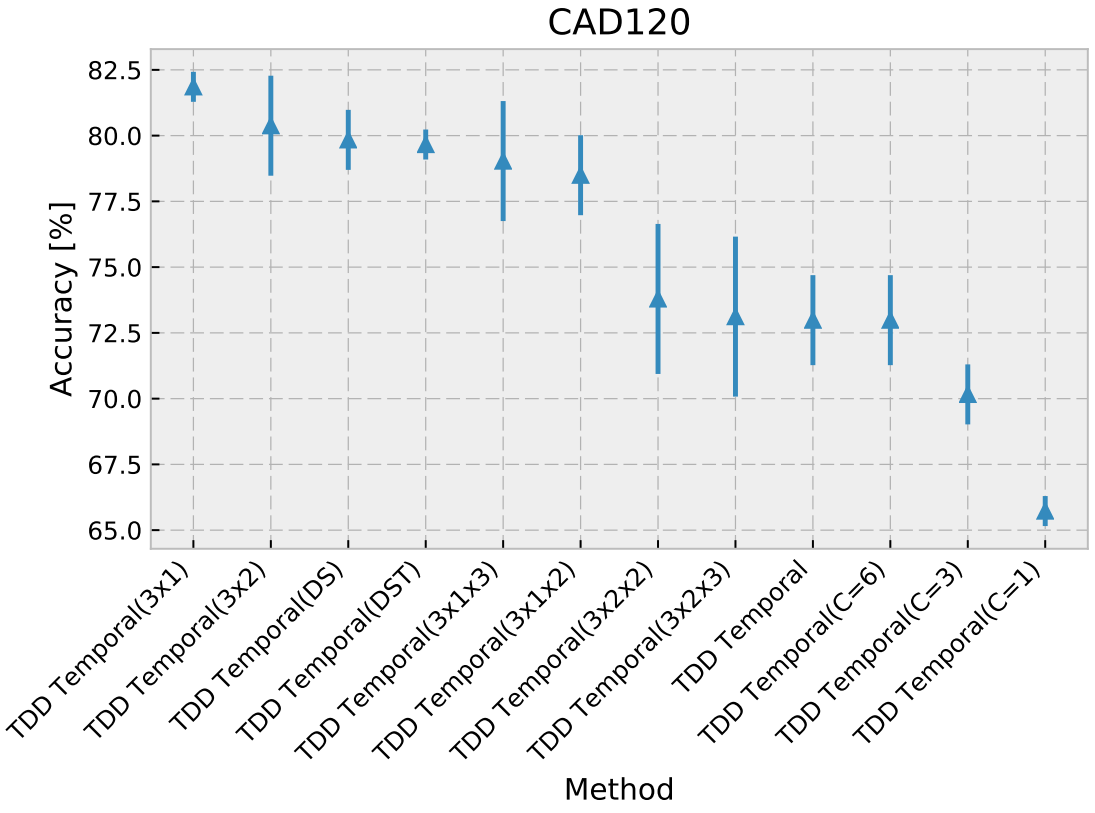


Figure 7.18: Comparison of proposed methods which model spatio-temporal layout on "TDD Temporal" descriptor and CAD-60 dataset. Length of lines indicate standard deviation.

Table 7.15: Modeling spatio-temporal layout of motion features: results on "TDD Temporal" descriptor and MSRDailyActivity3D dataset. Modeling spatial layout with grid: 3 columns by 2 rows, leads to significant improvement over standard "TDD Temporal".

	Acc [%]	σ
TDD Temporal(3x2)	81.52	1.66
TDD Temporal(3x1)	79.91	0.63
TDD Temporal(3x1x2)	79.84	0.53
TDD Temporal(3x2x2)	78.39	4.05
TDD Temporal(C=6)	69.84	0.66
TDD Temporal(C=3)	69.49	0.06
TDD Temporal(3x1x3)	68.25	0.23
TDD Temporal	68.12	0.44
TDD Temporal(C=1)	67.66	0.22
TDD Temporal(DST)	66.23	0.32
TDD Temporal(DS)	62.81	0.43
TDD Temporal(3x2x3)	62.13	3.13

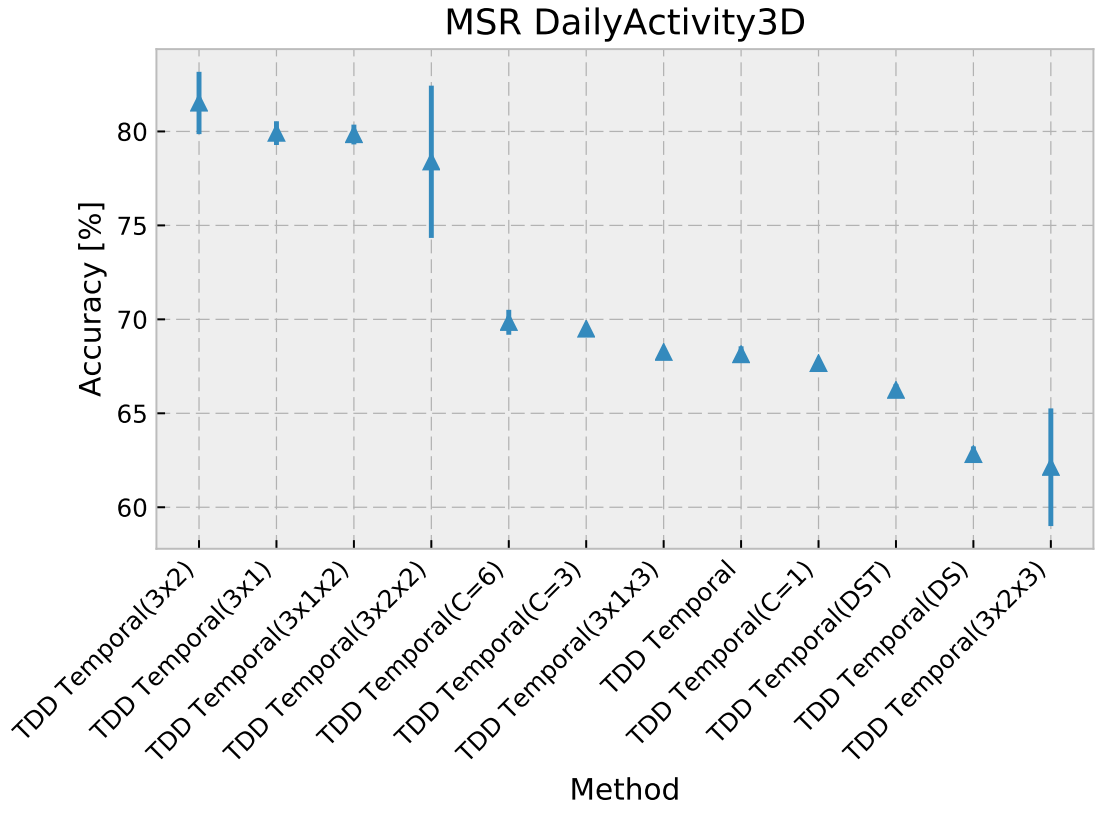


Figure 7.19: Comparison of proposed methods which model spatio-temporal layout on "TDD Temporal" descriptor and MSRDailyActivity3D dataset. Length of lines indicate standard deviation.

Table 7.16: Modeling spatio-temporal layout of motion features: results on "TDD Temporal" descriptor and Smarthomes dataset. Modeling spatial layout with grid: 3 columns by 1 rows, leads to significant improvement over standard "TDD Temporal".

	Acc [%]	σ
TDD Temporal(3x1)	48.72	0.41
TDD Temporal(3x2)	46.08	0.54
TDD Temporal(C=6)	45.35	1.31
TDD Temporal(C=1)	45.29	0.63
TDD Temporal(C=3)	45.08	1.41
TDD Temporal(DS)	44.18	0.16
TDD Temporal	43.15	0.18
TDD Temporal(DST)	42.11	0.02
TDD Temporal(3x1x2)	41.78	0.09
TDD Temporal(3x2x2)	41.02	0.66
TDD Temporal(3x1x3)	39.99	0.94
TDD Temporal(3x2x3)	38.18	0.44

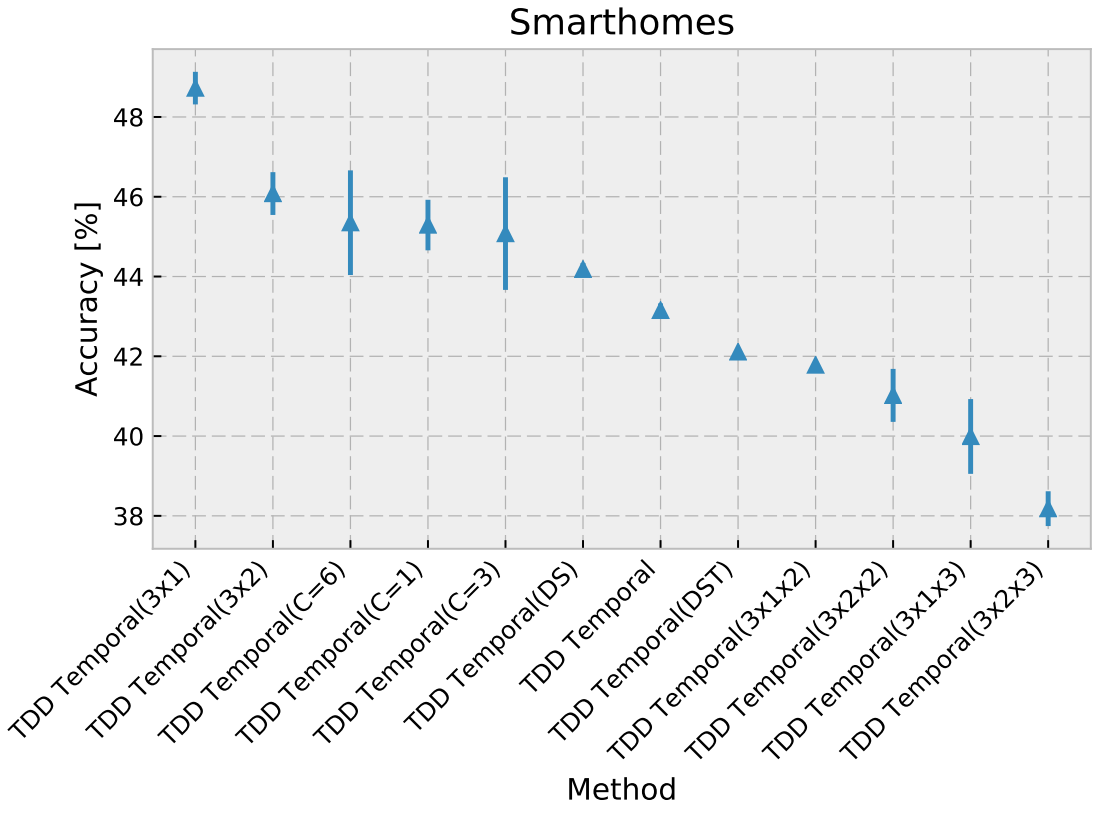


Figure 7.20: Comparison of proposed methods which model spatio-temporal layout on "TDD Temporal" descriptor and Smarthomes dataset. Length of lines indicate standard deviation.

Table 7.17: Modeling spatio-temporal layout: fusion with DT on CAD-60 dataset. Fusion of DT with HOG and MBH with grid (3 columns by 1 rows). The fusion leads to significant improvement over standard standard DT descriptor.

	Acc [%]	σ
DT+MBH(3x1)+HOG(3x1)	75.89	1.26
DT	70.54	1.26

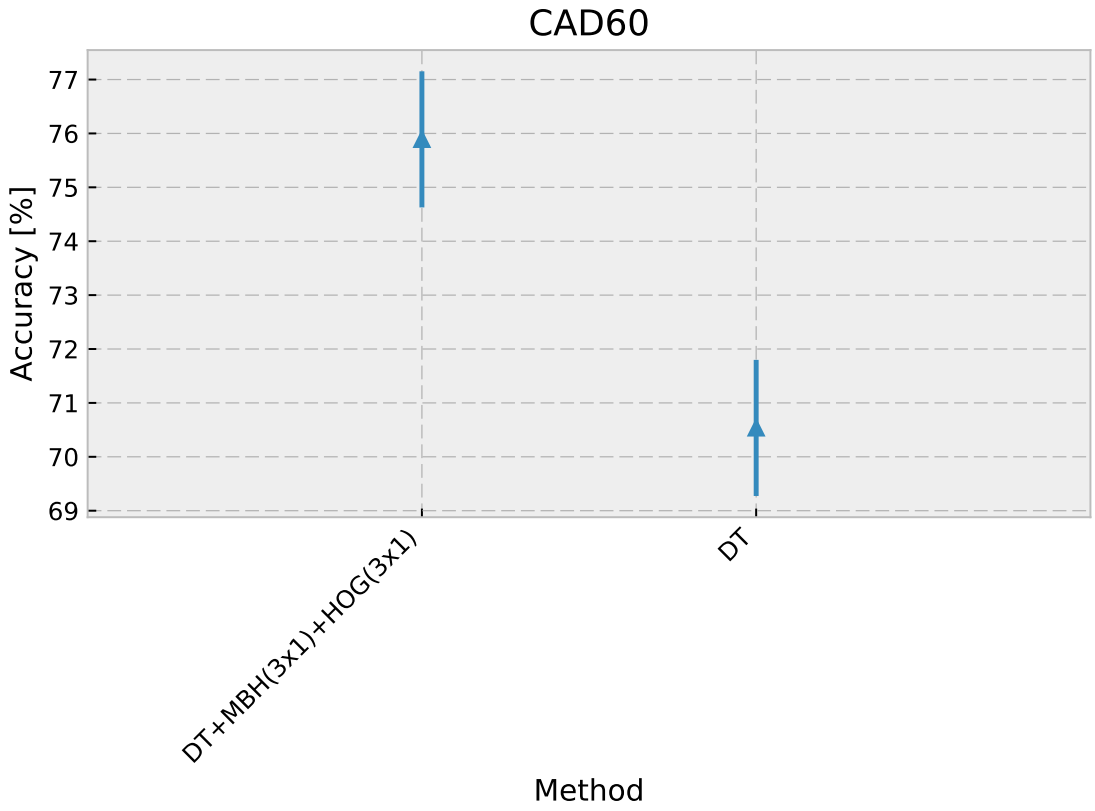


Figure 7.21: Fusion with DT features on CAD-60 dataset. Length of lines indicate standard deviation.

Table 7.18: Modeling spatio-temporal layout: fusion with MBH on CAD-120 dataset. Fusion of MBH with HOG and MBH with grid (3 columns by 2 rows). The fusion leads to significant improvement over standard standard DT descriptor.

	Acc [%]	σ
MBH+MBH(3x1)+HOG(3x1)	83.03	1.14
MBH	73.00	1.14

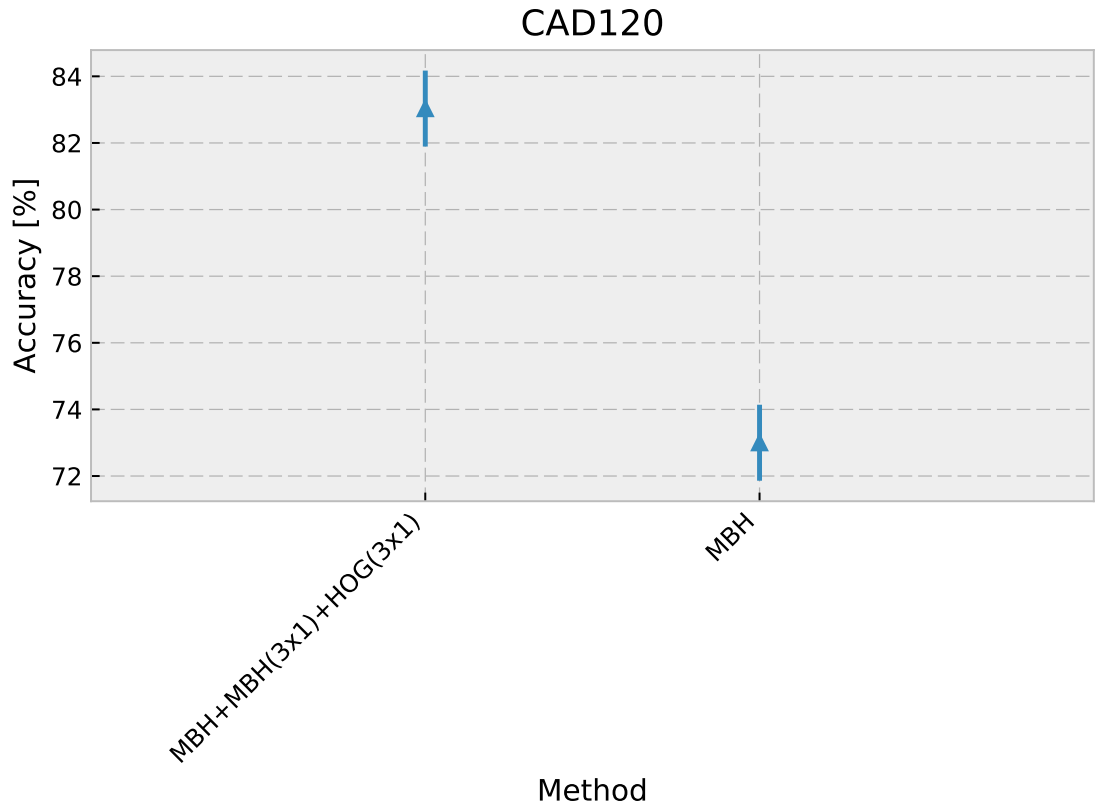


Figure 7.22: Fusion with DT features on CAD-120 dataset. Length of lines indicate standard deviation.

Table 7.19: Modeling spatio-temporal layout: fusion with DT on MSRDailyActivity3D dataset. Fusion of DT with HOG and MBH with grid (3 columns by 2 rows). The fusion leads to significant improvement over standard standard DT descriptor.

	Acc [%]	σ
DT+MBH(3x2)+HOG(3x2)	82.62	0.22
DT	75.47	0.22

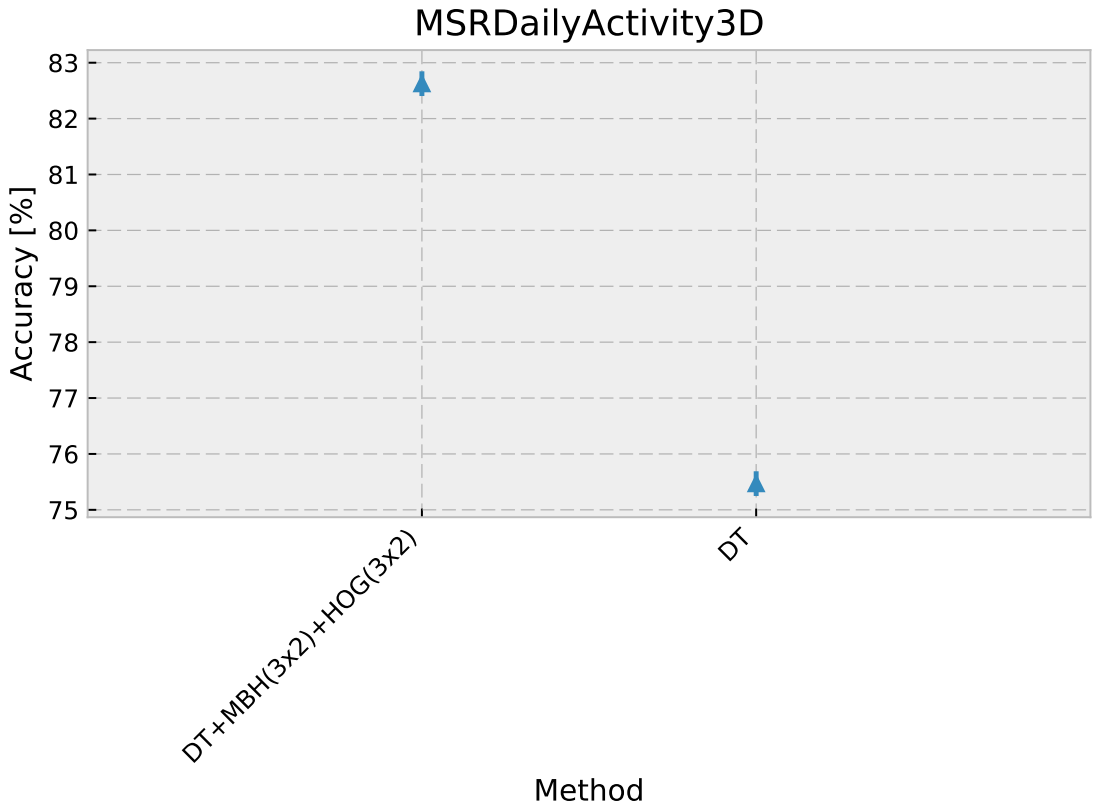


Figure 7.23: Fusion with DT features on MSRDailyActivity3D dataset. Length of lines indicate standard deviation.

Table 7.20: Modeling spatio-temporal layout: fusion with DT on Smarthomes dataset. Fusion of DT with HOG and MBH with grid (3 columns by 1 rows). The fusion leads to significant improvement over standard standard DT descriptor.

	Acc [%]	σ
DT+MBH(3x1)+HOG(3x1)	52.83	0.92
DT	47.23	0.20

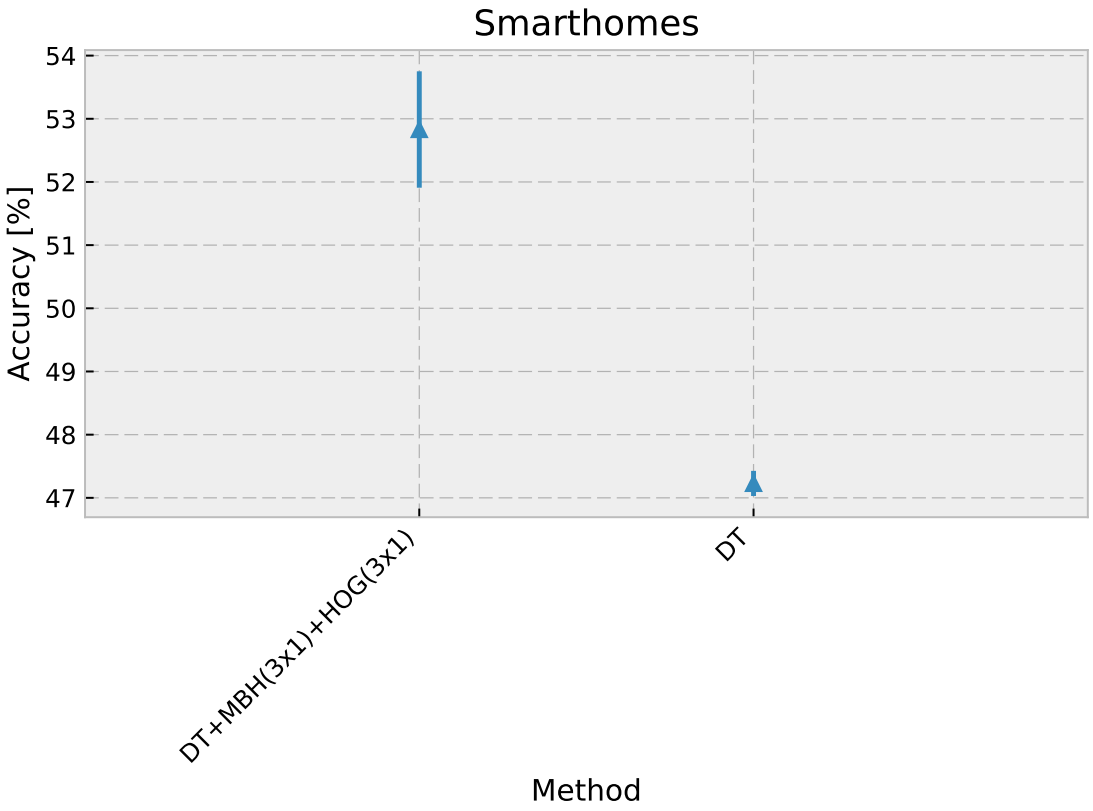


Figure 7.24: Fusion with DT features on Smarthomes dataset. Length of lines indicate standard deviation.

Table 7.21: Modeling spatio-temporal layout: fusion with TDD on CAD-60 dataset. Fusion of TDD with "TDD Appearance" and "TDD Temporal" with grid (3 columns by 1 rows). The fusion leads to significant improvement over standard standard TDD descriptor.

	Acc [%]	σ
TDD+TDD Temporal(3x1)+TDD Appearance(3x1)	75.00	2.53
TDD	65.18	1.26

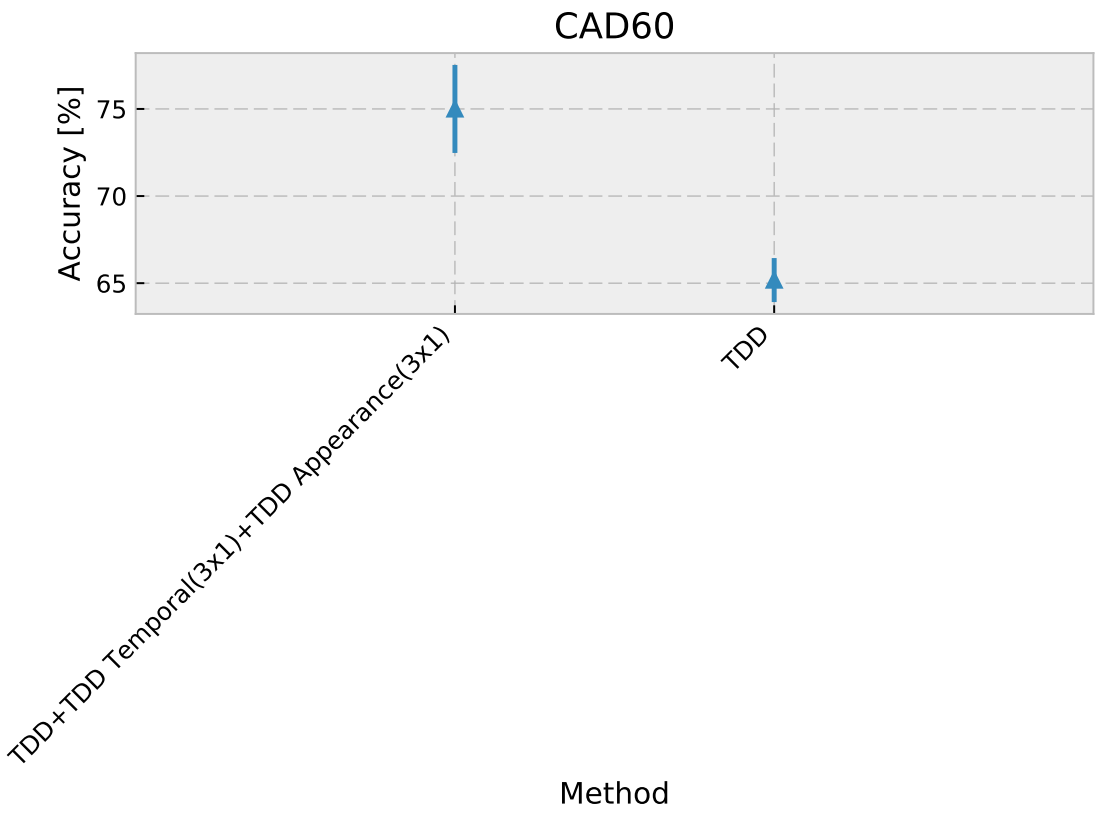


Figure 7.25: Fusion with TDD on CAD-60 dataset. Length of lines indicate standard deviation.

Table 7.22: Modeling spatio-temporal layout: fusion with TDD on CAD-120 dataset. Fusion of TDD with "TDD Appearance" and "TDD Temporal" with grid (3 columns by 1 rows). The fusion leads to significant improvement over standard standard TDD descriptor

	Acc [%]	σ
TDD+TDD Temporal(3x1)+TDD Appearance(3x1)	90.29	1.14
TDD	80.38	1.90

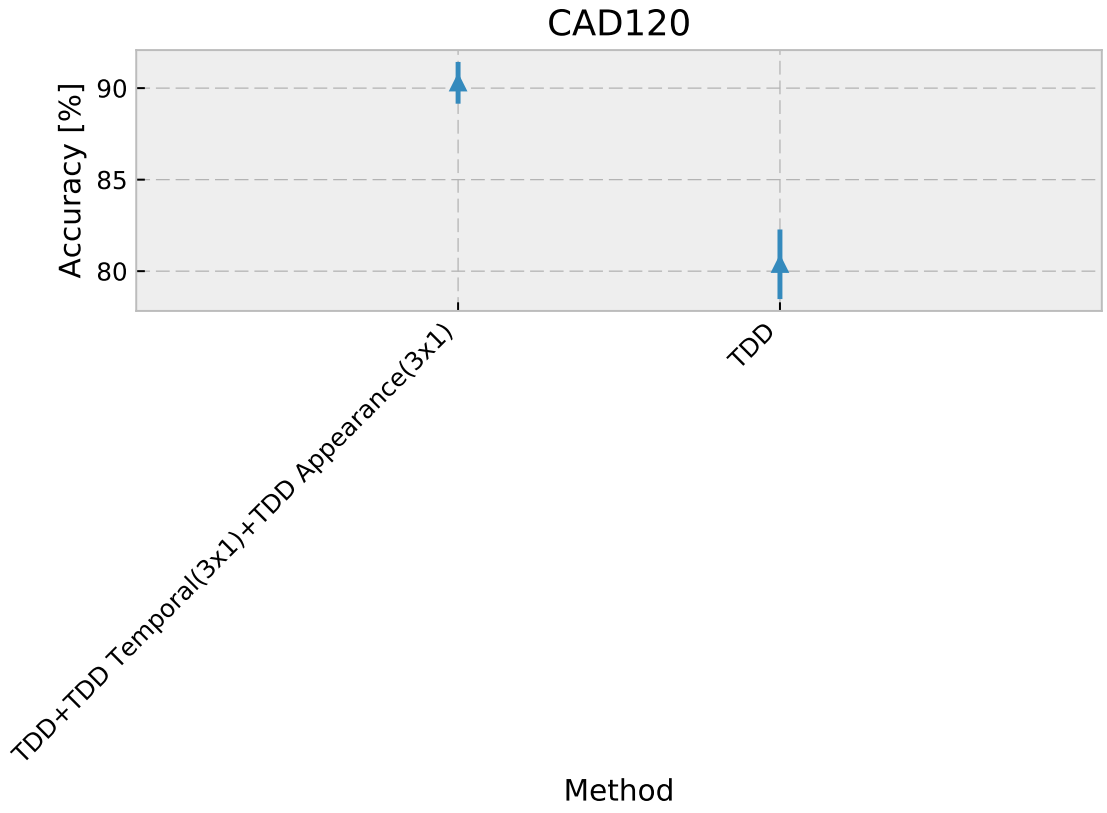


Figure 7.26: Fusion with TDD on CAD-120 dataset. Length of lines indicate standard deviation.

Table 7.23: Modeling spatio-temporal layout: fusion with TDD on MSRDailyActivity3D dataset. Fusion of TDD with "TDD Appearance" and "TDD Temporal" with grid (3 columns by 2 rows). The fusion leads to significant improvement over standard standard TDD descriptor

	Acc [%]	σ
TDD+TDD Temporal(3x1)+TDD Appearance(3x1)	80.73	0.71
TDD	73.28	0.22

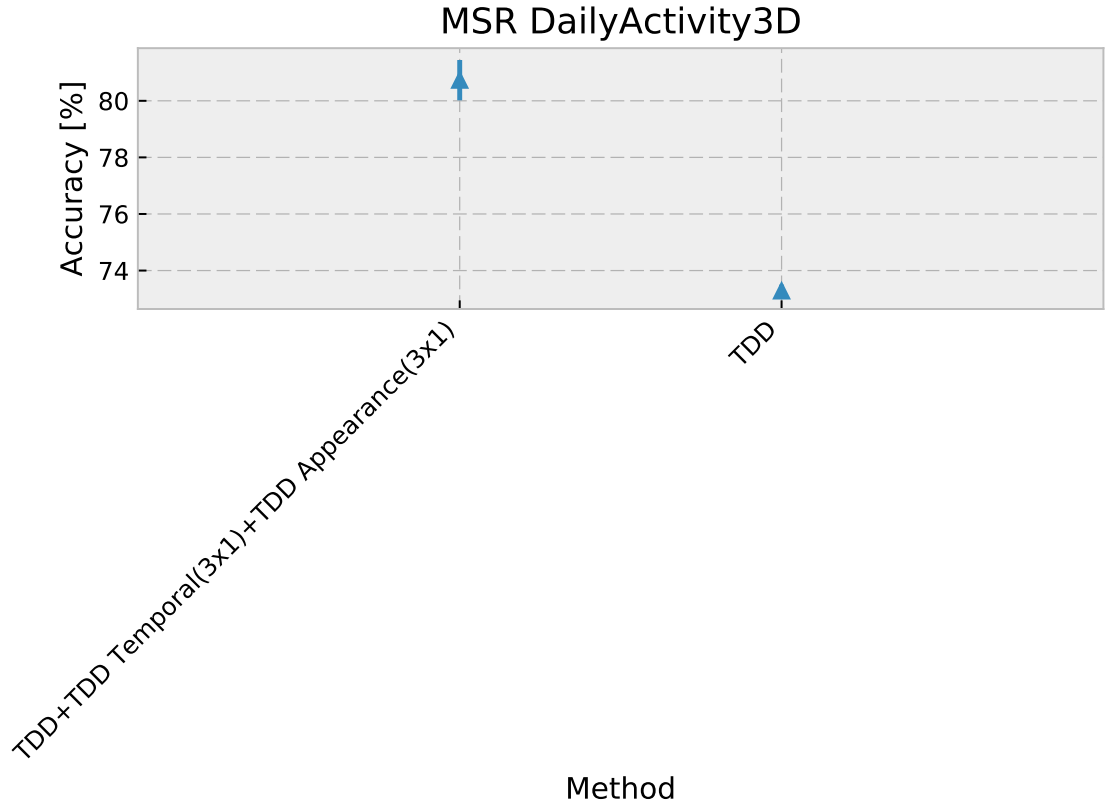


Figure 7.27: Fusion with TDD on MSRDailyActivity3D dataset. Length of lines indicate standard deviation.

Table 7.24: Modeling spatio-temporal layout: fusion with TDD on Smarthomes dataset. Fusion of TDD with "TDD Appearance" and "TDD Temporal" with grid (3 columns by 1 rows). The fusion leads to significant improvement over standard standard TDD descriptor

	Acc [%]	σ
TDD+TDD Temporal(3x1)+TDD Appearance(3x1)	47.02	0.87
TDD	45.93	0.16

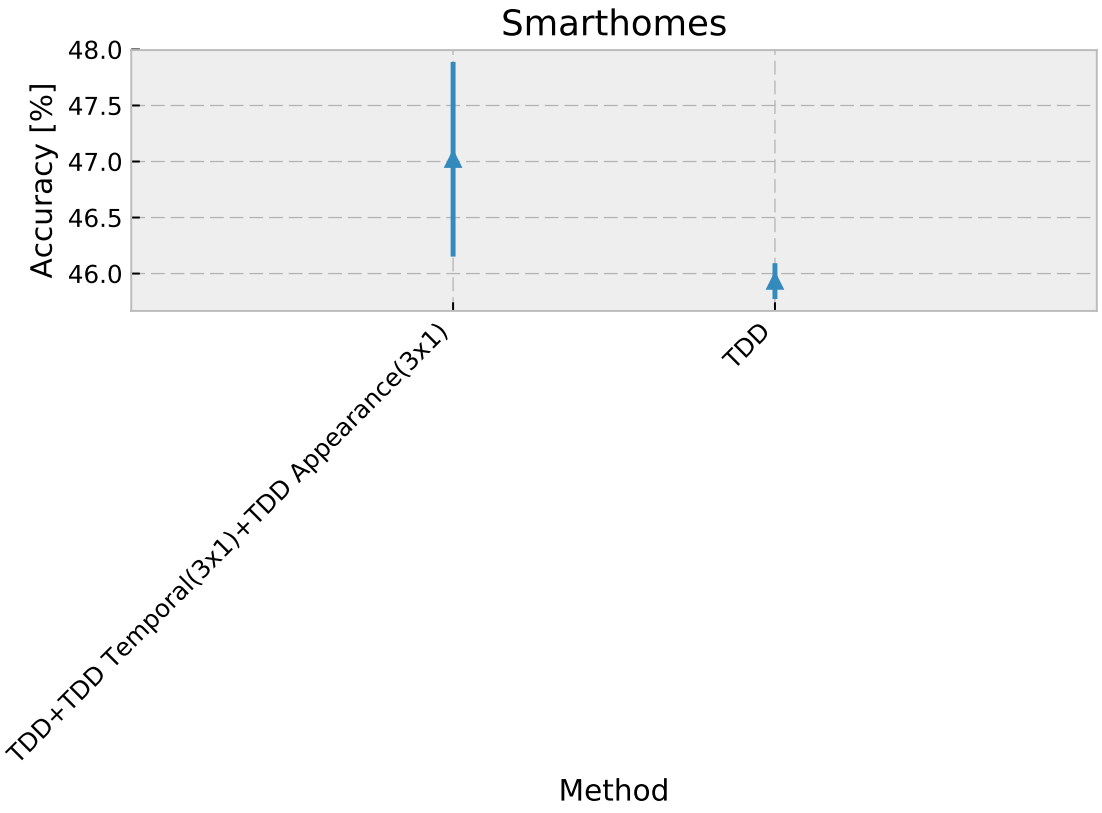


Figure 7.28: Fusion with TDD on Smarthomes dataset. Length of lines indicate standard deviation.

7.6 Conclusions

In this chapter we proposed how to improve Fisher Vector representation with spatio-temporal layout modeling. We proposed 3 methods which encodes spatial information in Fisher Vector: direct encoding, Grid Fisher Vector (GridFV) and Spatial Mixture of Gaussians (MoG). The best accuracy is achieved by spatial grids of size 3×1 or 3×2. The GridFV take advantage of implicit posture information (top grids belongs to head, bottom grids belongs to legs) and it is able to efficiently encode spatial information to Fisher Vector. GridFV with 3×1 layout showed good performance on all datasets and might be considered as good strategy to try this layout on new datasets.

Spatial Mixture of Gaussian methods achieved competitive results to GridFV method on Smarthomes dataset. As we explained in section 7.5.4 this is connected with amount of training data and variety of poses introduced in the dataset. In small dataset GMM focuses on areas with high features density. Spatial Mixture of Gaussian might be a good strategy to choose for datasets with big amount of data.

Direct encoding reported worse results as expected, but it was able to outperform baselines methods which ignore spatial layout of features. The advantage of direct encoding is that it does not introduce additional parameters to tune.

Finally we would like to highlight that propose spatial layout encoding methods improve both hand-crafted and CNN based representation and lead to significant improvements in recognition accuracy.

Chapter 8

Pose and skeleton based models

Contents

8.1 Introduction

For many years skeleton detection was a very difficult task in computer vision. The situation has changed after introduction of affordable RGB-D sensors such as Microsoft Kinect or Asus Xtion. Since then real-time skeleton detection became possible. And indeed many authors proposed action recognition methods based on skeleton, achieving good results comparing to local features methods. The results of skeleton based methods were confirming study of Johansson *et al.* [42], who showed that humans can recognize actions observing only positions of body joints.

One of the main disadvantage of skeleton based methods is that they cannot work when skeleton detection is missing or is too noisy. Skeleton detection algorithms based on RGB-D information fail when person is too far from sensor and signal to noise ration drops drops down. Many skeleton detection algorithms have problems when person is occluded for instance by furniture, which is often the case in daily-living surveillance scenarios. Recent advancements in deep-learning [14] show promising results of skeleton detection on RGB videos. Those methods do not have problems with depth-map noise, but suffer from higher false-positives rate. We believe that future improvements in skeleton detection will make it more reliable and skeleton joints information will play more important role in action recognition algorithms.

Nevertheless our experience with skeleton detection on Smarthomes dataset shows that skeleton detection is still unpractical in real-world scenarios. This holds especially true for RBG-D based methods.

When it comes to action recognition problem the one of the key advantages of skeleton based methods over local features methods is that skeleton based methods can better model actions with low amount of motion. For instance, actions like "using laptop" involve little amount of motion coming from fingers. Thus local features methods very often cannot detect enough points of interest to effectively model the action. Skeleton based methods on the other hand can recognize such actions based on static pose information.

In this chapter we propose static GHOG descriptor which captures rough pose information, requiring only people detection instead of skeleton detection. Such descriptor used with local features overcomes their issues with static actions, while being much more robust in terms of detection requirements. We fuse all methods proposed in previous chapters with GHOG descriptors and report results in experiments section.

In section 8.3 we evaluate also simple skeleton descriptor, which is based on pairwise distance between skeleton joints. Many public datasets including: CAD-60, CAD-120, MSRDailyActivity3D, provide detected skeleton joints with very good quality. Although we claim that such skeleton data is yet not possible to obtain in real-world scenario, we fuse all methods proposed in previous chapters with pairwise skeleton descriptor, to see the possible improvement and limitations. It is worth noting that our methods are still able to fall back and work when skeleton detection is unavailable. This makes them more robust than most state-of-the-art methods.

8.2 Grid HOG descriptor

We propose to encode static appearance by computing HOG descriptor inside person bounding box. Since HOG encodes gradient orientation and showed good performance on people detection task – it can also encode some useful information about person pose or appearance. To encode information about location of the HOG features we propose to divide bounding box into $n \times m$ grid. We compute HOG descriptor in each cell separately and then we form GHOG (GridHOG) descriptor by concatenating each cell descriptor. Since proposed descriptor is supposed to capture static information about pose and appearance we compute with step of t frames ($t = 10$). The overview of GHOG is presented in fig. 8.1. Please note that there is a key difference between modeling spatial layout of detected local features with grid in section 7.2.2 and GHOG. In GHOG we compute HOG descriptor for cropped cell and we do it with step of t frames, while in method from section 7.2.2 we compute separate grid Fisher Vector representation of local descriptors detected and calculated in person bounding box.

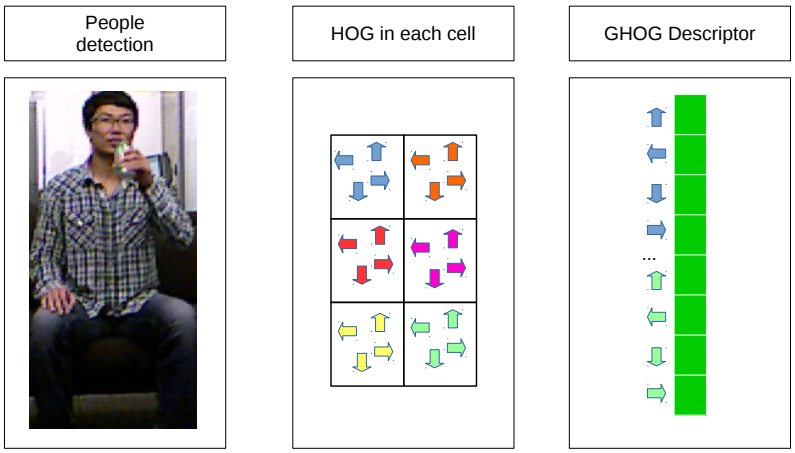


Figure 8.1: To obtain GHOG descriptor we divide detected person bounding box into $n \times m$ spatial grids. We compute HOG descriptor for each cell of the grid. Finally we concatenate HOG representation of each cell forming GHOG descriptor.

The primary goal of GHOG is to provide information about pose without need of skeleton detection and to help to recognize the actions with low amount of motion.

8.3 Skeleton descriptor

In this section we describe simple skeleton descriptor based on pairwise distances between joints. For each video frame we take the detected skeleton and compute distance between each pair of 3D joints positions. Assuming that we obtained n joints from the skeleton detector our descriptor will contains $\frac{n!}{2(n-2)!}$ distances. We normalize obtained distance vector with $L2$ norm. The outline of this method is available in fig. 8.2.

8.4 Experiments

In this section we concentrate on fusion of GHOG and skeleton descriptor with descriptors proposed in previous chapters. GHOG and skeleton descriptor alone do not perform well. This is expected as they encode only static pose information, which is not sufficient to recognize actions.

In table 8.2 we show results of fusion of GHOG with methods proposed in previous chapters on CAD-60 dataset. Results show that fusion is effective, thus GHOG carries complementary information. GHOG mostly improve actions with low amount of motion, where small number of local points of interest were detected and where pose is stronger

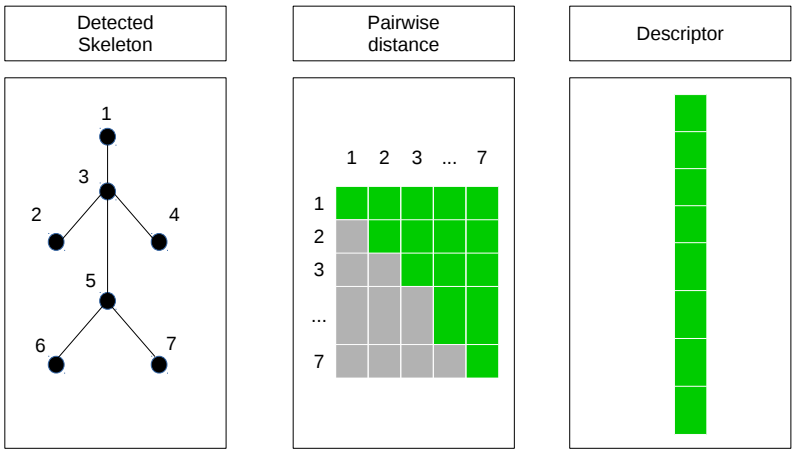


Figure 8.2: Pairwise distance skeleton descriptor. For each detected skeleton frame we compute distance between each pair of 3D joints. Thus we form the pairwise distance descriptor which we normalize with $L2$ norm.

clue (see table 8.1). Similarly on CAD-120 (table 8.3), MSRDailyActivity3D (table 8.4) and Smarthomes (table 8.5) fusion with GHOG descriptor leads to accuracy improvement. Thus we conclude that fusion of any local descriptor with GHOG can be considered as a good strategy.

In last column of table 8.2, table 8.3, table 8.4, table 8.5 we provide results of fusion with GHOG and skeleton descriptor. In case of CAD-60 and MSRDailyActivity3D where skeleton is well detected, we observe further accuracy improvement. When it comes to CAD-120 and Smarthomes dataset we observe decrease in accuracy. CAD-120 introduces lateral view which is more challenging for skeleton detection. In case of Smarthomes dataset we use Pose Machines [14], which uses RGB frames as an input. We use net resolution of 656×496 and 4 spatial scales. The processing speed is 2 fps on single GPU. Skeleton was not correctly detected in around 30% of frames. This is a main factor which limits the performance of fusion with skeleton descriptor. In addition the obtained skeleton was two dimensional, which is a disadvantage in multiview environment. In case of GHOG we used SSD [59] which provided better detection than Pose Machines at 30 fps. We would like to highlight, that we appreciate work of Pose Machines and it's huge contribution in skeleton detection, our intention is to show the scale of difficulty.

Table 8.1: The table shows average number of detected features on CAD-60 dataset using Dense Trajectories. Third column shows relative improvement of fusion of DT descriptor with GHOG. The results show recognition accuracy for actions with small number of detected features is improved, when GHOG descriptor is employed.

Action	Number of features	GHOG improvement
Open pill container	1240	**+48%**
Relaxing on couch	1346	**+66%**
Working on computer	1356	**+25%**
Still	1510	0%
Talking on the phone	1718	**+13%**
Talking on couch	2060	**+66%**
Drinking water	3079	0%
Wearing contact lenses	4366	0%
Cooking – chopping	4448	-16%
Cooking – stirring	4961	0%
Brushing teeth	5527	**+12%**
Writing on white board	5616	0%
Rinsing mouth	23258	0%
Random	45340	0%

Table 8.2: CAD60 - action recognition accuracy of proposed method, alone, fused with GHOH, fused with GHOG and pairwise skeleton descriptor.

Method	Fusion					
	None		+GHOG		+GHOG+Skeleton	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
3DTSD (chapter 4)	73.21	0.00	83.04	3.79	90.36	0.00
HOG_EB (chapter 5)	71.84	1.26	80.36	2.53	87.50	0.00
Brownian (chapter 6)	74.86	1.26	85.71	2.53	88.39	1.26
GridFV(chapter 7)	75.89	1.26	83.57	0.00	91.58	0.00

Table 8.3: CAD120 - action recognition accuracy of proposed method, alone, fused with GHOH, fused with GHOG and pairwise skeleton descriptor.

Method	Fusion					
	None		+GHOG		+GHOG+Skeleton	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
3DTSD (chapter 4)	75.29	1.27	78.23	2.28	77.42	0.00
HOG_EB (chapter 5)	74.77	1.14	78.21	1.71	77.82	2.85
Brownian (chapter 6)	75.18	1.71	78.16	0.00	76.63	2.85
GridFV(chapter 7)	90.29	1.14	94.69	0.57	89.27	1.71

Table 8.4: MSR Daily Activity 3D - action recognition accuracy of proposed method, alone, fused with GHOH, fused with GHOG and pairwise skeleton descriptor.

Method	Fusion					
	None		+GHOG		+GHOG+Skeleton	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
3DTSD (chapter 4)	76.06	0.66	80.15	1.28	82.46	1.10
HOG_EB (chapter 5)	79.12	0.00	81.44	1.10	83.43	0.22
Brownian (chapter 6)	75.75	1.10	79.46	1.20	82.38	0.44
GridFV(chapter 7)	82.62	0.22	85.56	0.00	87.97	0.22

Table 8.5: Smarthomes - action recognition accuracy of proposed method, alone, fused with GHOH, fused with GHOG and pairwise skeleton descriptor.

Method	Fusion					
	None		+GHOG		+GHOG+Skeleton	
	Acc [%]	σ	Acc [%]	σ	Acc [%]	σ
HOG_EB (chapter 5)	49.56	0.37	51.87	0.10	51.26	0.40
Brownian (chapter 6)	50.21	1.46	51.70	0.43	49.38	0.96
GridFV(chapter 7)	52.83	0.92	53.73	0.13	51.50	0.17

8.5 Conclusions

In this chapter we proposed GHOG descriptor, which encodes static pose information. The fusion of GHOG with local descriptors leads to accuracy improvement, especially of actions with low amount of motion. Our GHOG descriptor requires only people detection. This as an advantage, because skeleton detection is still challenging task in environments such as Smarthomes. Problems in skeleton detection directly affects skeleton descriptor and recognition accuracy. Better strategy is to use our GHOG descriptor and skeleton descriptor, in this way if skeleton detection is missing, recognition system can fallback.

Chapter 9

Comparison with state-of-the-art

Contents

9.1 Introduction

In this chapter we provide comparison of proposed methods with state-of-the-art. For each state-of-the-art method we provide information if skeleton detection is required. All methods proposed in this thesis do not require skeleton detection, thus are more robust in terms of detection requirements. But since skeleton information is a strong clue to make the comparison more clear, for each proposed method we provide the performance of: method alone, fused with GHOG (which provides pose information without skeleton detection requirement – see chapter 8) and fused with GHOG and skeleton descriptor based on pairwise distance (see chapter 8).

In the previous chapters we provided deep analysis of each method, where we tested different parameters and fusions. The final results provided in this chapter refer to most effective fusion and parameter setting for each proposed method. For instance the result of Brownian on CAD-60 dataset refers to Brownian(9)+DT descriptor (see section 6.4.1). In following sections we present performance of all methods proposed in this thesis on CAD-60, CAD-120, MSRDailyActivity3D and Smarthomes dataset. Section 9.6 summarizes the results.

9.2 CAD-60

The methods: 3DTSD , HOG_EB and Brownian are pure local methods and do not require neither people detection nor skeleton detection, while GridFV requires people detection. The goal of the mentioned methods was to improve action representation in such a way to be able to distinguish actions which are similar to each other. Results in table 9.1 show that local features methods proposed in this thesis outperform all state-of-the-art methods which use local features, and outperform most of the methods which use skeleton information. Our methods are able to successfully distinguish actions such as: rinsing mouth, wearing contact lenses, brush teeth which share similar motion patterns. For instance Actionlet Ensemble [116] fails to recognize such actions. Our methods fail on the other hand to distinguish actions with low amount of motion: siting still, talking on the phone, relaxing on a couch. These actions are well recognized by Actionlet Ensemble, thus final accuracy of the proposed methods and Actionlet Ensemble is close. But most importantly our methods do not require skeleton detection.

To address the problem of actions with low amount of motion in section 8.2 we proposed GHOG – descriptor which models rough pose information, requiring only people detection instead of skeleton detection. Fusion with GHOG outperform all state-of-the-art methods, which require skeleton detection except P-CNN Fusion [105]. Fusion with GHOG helps to recognize actions like: relaxing on a couch, siting still and talking on the phone.

Further fusion with skeleton descriptor (see section 8.3) leads to significant improvement, again mostly on actions with low amount of motion. The skeleton detection on CAD-60 dataset is not difficult, in most cases people are standing in front of a camera and there are no serious occlusions. In such a case skeleton methods perform well, as they do not have to deal with noise and missing detection. The results show that our methods achieve good results standalone, but also they are able to take advantage of skeleton information if required. P-CNN Fusion method is the only which outperforms methods proposed in this thesis on CAD-60. The P-CNN Fusion is a method recently proposed by us, which uses skeleton detection and CNN features with different skeleton detection algorithms.

Table 9.1: CAD60: comparison of proposed methods with the state-of-the-art.

Method	Ch./Year	Accuracy [%]	Skeleton
3DTSD	4	73.21	Not Req.
3DTSD + GHOG	4	83.04	Not Req.
3DTSD + GHOG + Skeleton	4	90.36	Required
HOG_EB	5	71.84	Not Req.
HOG_EB + GHOG	5	80.36	Not Req.
HOG_EB + GHOG + Skeleton	5	87.50	Required
Brownian	6	74.86	Not Req.
Brownian + GHOG	6	85.71	Not Req.
Brownian + GHOG + Skeleton	6	88.39	Required
GridFV	7	75.89	Not Req.
GridFV+ GHOG	7	83.57	Not Req.
GridFV+ GHOG + Skeleton	7	91.58	Required
Dense Trajectories [113]	3	70.54	Not Req.
TDD [117]	3	65.18	Not Req.
STIP [142]	2014	62.50	Not Req.
MEMM [107]	2012	51.90	Required
Order Sparse Coding [73]	2012	65.30	Required
Object Affordance [46]	2013	71.40	Required
HON4D [78]	2013	72.70	Required
Actionlet Ensemble [116]	2012	74.70	Required
JOULE-SVM [38]	2015	84.10	Required
P-CNN Fusion [105]	2017	95.58	Required

9.3 CAD-120

CAD-120 dataset is described in section 2.3.2, the characteristic feature of this dataset, is that it contains actions with strong motion patterns: object stacking, object picking, object arranging. Also actions are performed in the way that inter-class variation is high, for instance taking medicine action involves opening big container, which makes this action very different to taking food in terms of motion pattern. Because of that MBH descriptor reports best performance (section 3.5.1.2), as MBH has been proven to model action motion pattern very well. Strong motion patterns in CAD-120 are also the reason of good performance of TDD descriptor (section 3.5.2.2), as CNN convolutional filters which model local motion are similar to MBH descriptor.

The methods: HOG_EB and Brownian proposed in this thesis are designed to improve appearance representation, which plays a marginal role in CAD-120 dataset. There are two groups of actions with similar motion pattern in CAD-120: picking object – arranging object and stacking objects, unstacking objects, which could be improved by appearance information, but in case of CAD-120 same object are used in mentioned actions pairs. Because of that appearance information modeled by HOG_EB and Brownian deos not lead to significant accuracy improvement. We managed to outperform only MBH proposed by Wang *et al.* [113] and P-CNN Fusion [105] with HOG_EB and Brownian .

Similarly 3DTSD with 3D trajectory shape we cannot distinguish actions like: picking object, arranging objects, stacking objects and unstacking objects. Trajectory shape information does not encode enough information to catch detailed nuances.

GridFV method proposed in section 7.2.2 on the other hand is able to improve both, motion and temporal features and leads to state-of-the-art results. We report significant improvement (almost 10%) over baseline TDD method, we outperform also all state-of-the-art methods except STS [45]. But STS method requires skeleton detection, while we require only people detection. The proposed method allow to model details of motion pattern, thus actions like: arranging objects, unstacking objects, stacking objects are recognized with much higher accuracy rate. Further fusion of GridFV with our GHOG descriptor leads to further improvement and leads to state-of-the-art result. It is worth noting that because CAD-120 does not contain actions with low amount of motion the fusion with GHOG is not as effective as it was in CAD-60, but modeling pose with GHOG still improves the accuracy. Lin *et al.* in RSVM+LCNN [56] also models spatio-temporal layout of CNN features *w.r.t.* global image coordinates, in addition it arbitrarily divide each action into 4 temporal segments. This might be a limiting factor, as we showed in section 7.5 that arbitrary division of GridFV to 3 temporal volumes does not lead to accuracy improvement.

Further fusion of proposed methods with pairwise skeleton descriptor resulted in slight

Table 9.2: CAD120: comparison of proposed methods with the state-of-the-art.

Method	Ch./Year	Accuracy [%]	Skeleton
3DTSD	4	75.29	Not Req.
3DTSD + GHOG	4	78.23	Not Req.
3DTSD + GHOG + Skeleton	4	77.42	Required
HOG_EB	5	74.77	Not Req.
HOG_EB + GHOG	5	78.21	Not Req.
HOG_EB + GHOG + Skeleton	5	77.82	Required
Brownian	6	75.18	Not Req.
Brownian + GHOG	6	78.16	Not Req.
Brownian + GHOG + Skeleton	6	76.63	Required
GridFV	7	90.29	Not Req.
GridFV+ GHOG	7	94.69	Not Req.
GridFV+ GHOG + Skeleton	7	89.27	Required
Dense Trajectories [113]	3	73.00	Not Req.
TDD [117]	3	80.38	Not Req.
Salient Proto-Objects[91]	2014	78.20	Not Req.
SVM+CNN[56]	2016	78.30	Not Req.
RSVM+LCNN[56]	2016	90.10	Not Req.
P-CNN Fusion [105]	2017	70.96	Required
Object Affordance [46]	2013	84.70	Required
R-HCRF [58]	2016	89.80	Required
STS [45]	2013	93.50	Required

decrease in accuracy. CAD-120 introduces both frontal and lateral views, in lateral view skeleton detection very often is noisy as some joints are not visible. Thus descriptor based on pairwise distance does not handle such situations well.

9.4 MSRDailyActivity3D

The details of MSRDailyActivity3D dataset are available in section 2.3.3. The local features (3DTSD , HOG_EB and Brownian) proposed in this thesis outperform baseline methods and NBNN [94]. They achieve better accuracy because they are able to model more efficiently appearance information and distinguish actions with similar motion pattern. For instance in MSRDailyActivity3D eating and drinking look similar. Both actions are confused by DT [113] method, but when we apply proposed HOG normalization with HOG_EB (chapter 5), both actions are perfectly recognized. Better appearance modeling allows classifier to distinguish actions which involves a soda can from action which involves a snack.

Again best performing method among proposed ones is GridFV. In MSRDailyActivity3D actions are performed in standing and sitting positions. Modeling spatial layout helps in recognition of actions performed in standing position, because people tend to bow or walk while performing action. This generates a lot of body motion which is unrelated to action. Modeling spatial layout with spatial grid helps to model such actions. For instance actions like read book, sit, toss paper are significantly improved, because many motion features unrelated to action were assigned to different spatial grids. MSRDailyActivity3D also contains actions with low amount of motion: sit, play game (motion only from fingers on game pad), play guitar. Thus fusion with our GHOG descriptor leads to further accuracy improvement. Fusion of GridFV and GHOG outperforms most of state-of-the-art methods which require skeleton detection. Our method is more robust and require only people detection.

The further fusion with skeleton descriptor is outperformed by JOULE-SVM [38] and Range Sample [61]. In our opinion the actions in MSRDailyActivity3D introduce more complicated spatio-temporal structure and skeleton descriptor based on pairwise distance does not provide enough complementary information to make fusion effective.

Table 9.3: MSR Daily Activity 3D: comparison of proposed methods with the state-of-the-art.

Method	Ch./Year	Accuracy [%]	Skeleton
3DTSD	4	76.06	Not Req.
3DTSD + GHOG	4	80.15	Not Req.
3DTSD + GHOG + Skeleton	4	82.46	Required
HOG_EB	5	79.12	Not Req.
HOG_EB + GHOG	5	81.44	Not Req.
HOG_EB + GHOG + Skeleton	5	83.43	Required
Brownian	6	75.75	Not Req.
Brownian + GHOG	6	79.46	Not Req.
Brownian + GHOG + Skeleton	6	82.38	Required
GridFV	7	82.62	Not Req.
GridFV+ GHOG	7	85.56	Not Req.
GridFV+ GHOG + Skeleton	7	87.97	Required
Dense Trajectories [113]	3	75.47	Not Req.
TDD [117]	3	73.28	Not Req.
NBNN [94]	2013	70.00	Required
Amor *et al.* [1]	2016	70.00	Required
HON4D [78]	2013	80.00	Required
STIP + Skeleton [142]	2014	80.00	Required
SSFF [97]	2014	81.90	Required
DSCF [129]	2013	83.60	Required
P-CNN Fusion [105]	2017	84.37	Required
Depth CNN [122]	2013	85.00	Required
RGGP + fusion [57]	2013	85.60	Required
Actionlet Ensemble [116]	2012	85.80	Required
SNV [134]	2017	86.25	Required
Super Normal [133]	2014	86.26	Required
DCSF + joint [129]	2013	88.20	Required
JOULE-SVM [38]	2015	95.00	Required
Range Sample [61]	2015	95.60	Required

9.5 Smarthomes

The Smarthomes dataset is bigger and more challenging than previously presented datasets. Because the dataset is not publicly available we will compare our methods to DT and TDD baselines defined in chapter 3. The table 9.4 shows that all proposed methods outperformed the baseline methods. The HOG_EB and Brownian improved recognition of actions like: "take pills" which was confused with "drinking" by DT and TDD. The improvement comes from better appearance modeling. Similarly recognition of all cooking actions was improved. Also actions like "use laptop", "use tablet", "read book" were much better improved thanks to improved appearance modeling.

GridFV method thanks to better modeling of motion features and as well appearance features improved recognition rate of almost all action class available in data-set. Thus as we can see from the results improvement is quite significant regarding size of the dataset.

The fusion with GHOG descriptor is less effective than in previously presented dataset. Actions in Smarthomes dataset are done in more natural way, there is high variety of different poses, in addition people are much further from the camera, so GHOG descriptor is not able to capture enough details to describe rough pose information.

Fusion with skeleton descriptor lead to accuracy decrease. In case of Smarthomes dataset simple pairwise distance descriptor is too simple and cannot handle well missing and noisy joints detection. In addition the skeleton detection method use with Smarthomes dataset provided only 2D joints positions and this might be another reason why fusion with skeleton features did not improve the results.

Table 9.4: Smarthomes: comparison of proposed methods with the state-of-the-art.

Method	Ch./Year	Accuracy [%]	Skeleton
HOG_EB	5	49.56	Not Req.
HOG_EB + GHOG	5	51.87	Not Req.
HOG_EB + GHOG + Skeleton	5	51.26	Required
Brownian	6	50.21	Not Req.
Brownian + GHOG	6	51.70	Not Req.
Brownian + GHOG + Skeleton	6	49.38	Required
GridFV	7	52.83	Not Req.
GridFV+ GHOG	7	53.73	Not Req.
GridFV+ GHOG + Skeleton	7	51.50	Required
Dense Trajectories [113]	3	47.23	Not Req.
TDD [117]	3	45.93	Not Req.

9.6 Conclusions

This chapter focuses on performance analysis of proposed techniques. We showed evaluation of our methods, investigating their pros and cons. Our techniques are explored in comparison with state-of-the-art approaches on various datasets, showing competitive performance.

The best performance of our methods is achieved by GridFV, which models spatial layout of features *w.r.t.* to person bounding box. The proposed method achieves comparable results to methods which require skeleton detection. Apart from recent advancements in skeleton detection on RGB, we claim that people detection is still more robust and reliable. In addition most of state-of-the-art methods use 3D skeleton information, thus skeleton obtained from RGB cannot be directly used without further post-processing. RGB-D skeleton detection methods on the other hand are difficult to use in environments like Smarthomes, where RGB-D sensor is located out of it's sweet spot position, is interfering with other sensor or fails to get depth information from materials which absorb IR beam.

Our local feature based methods (3DTSD, HOG_EB and Brownian) show worse performance. Such conclusion was expected as these methods ignore spatial layout of features and are bounded by performance of local feature detector. On the other hand their fusion with GHOG descriptor usually leads to significant accuracy improvement. Thus their performance can be simply improved by simple features fusion.

Our GHOG descriptor, which models static pose of person within the bounding box, showed that though it does not achieve accuracy alone, it's fusion with other descriptors can be very effective. GHOG is able to improve recognition accuracy especially when action does not contain high amount of motion, and local features methods fail to detect enough features.

Finally fusion with skeleton descriptor showed moderate improvements and in some cases slight accuracy decrease. The skeleton descriptor based on pairwise distance used in fusion does not handle well noise and missing joint detection. Fusion of methods proposed in this thesis with more advanced skeleton descriptor may lead to future accuracy improvement.

We would like to highlight that methods proposed in this thesis can easily fallback if skeleton detection or people detection is missing and still offer good action recognition accuracy.

Chapter 10

Online Action Detection

Contents

10.1 Introduction

In all previous methods and experiments described in this thesis, we assumed that all videos have been pre-segmented by an oracle. Thus we have precise video segments, which contain single action. In such case our task can be seen as classification problem, where we assign label to each video segment. We will refer this problem as Action Recognition problem.

In many practical applications (*eg.* surveillance, robotics), cameras provide long untrimmed video stream, thus computer vision algorithms have to perform temporal action detection, followed by action recognition. Another practical aspect of Action Detection and Action Recognition systems is their ability to work in Real-Time. It is especially important in robotics, where event from Action Detection system cannot arrive too late, that robot cannot take appropriate action anymore. We claim that Real-Time property of Action Recognition is tightly connected with quality of Action Detection. This is because the faster Action Recognition system is, the bigger temporal space can be searched to find temporal segments. The overall speed of Action Detection can be also improved by effi-

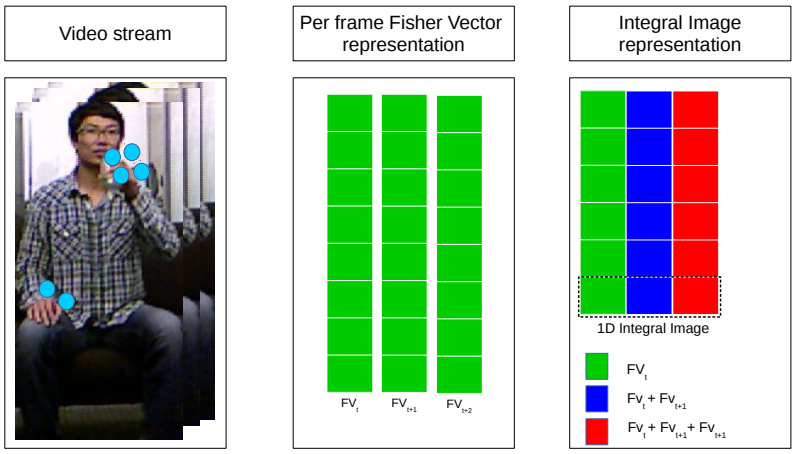


Figure 10.1: Fisher Vector integral image representation. For each frame of video stream we compute local descriptors and Fisher Vector representation. Thus we have separate Fisher Vector for each frame of video stream. This representation is saved to integral image.

cient search strategy *eg.* by action proposals [99, 37, 27, 13], but fast Action Recognition module is always an advantage.

In this chapter we introduce online Action Detection system, which can be seen as a proof of concept, that Action Recognition methods proposed in this thesis can be used as a component of Real-Time Action Detection system. We use multi-scale sliding window approach to detect an action. To make the system Real-Time we use 1-D integral image representation over unnormalized Fisher Vector representation. This allows us to efficiently compute sliding window representation on multiple temporal scales.

In the reminder of this chapter we will provide brief review of state-of-the-art Action Detection methods, describe our Action Detection framework and provide experimental results on Smarthomes dataset. It is worth mentioning that our system has been successfully deployed into Toyota partner robot system.

10.2 Action Detection Framework

Our Action Detection system is designed to work with a continuous video stream as an input. The input can consist of RGB or RBG-D frames. In our processing framework we utilize classifier, which was trained on clipped images. Our processing pipeline involves the following steps: first we extract features for each frame. Then we compute Fisher Vector and save it to Integral Image representation (see fig. 10.1). To detect actions we

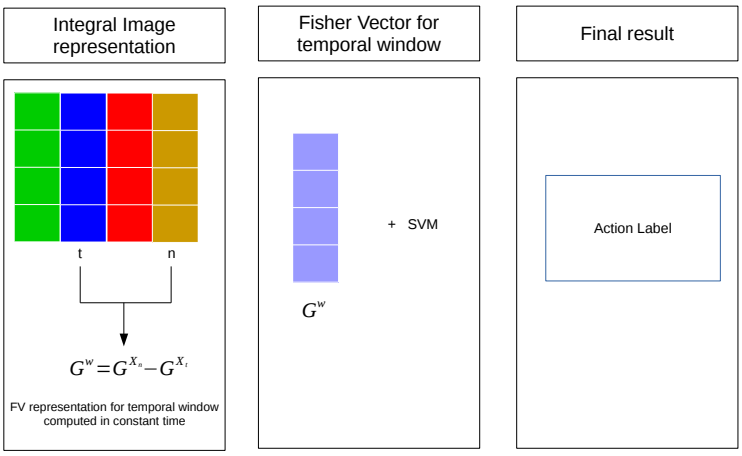


Figure 10.2: We use integral image representation to obtain Fisher Vector representation of time window W. This computation is constant in terms of computational complexity. Once Fisher Vector for time window W is obtained, we use it as an input for SVM classifier.

select multiple sizes of temporal windows, for each window we compute Fisher Vector and perform classification to obtain window's label and confidence score (see fig. 10.2). Finally we prune temporal windows with low confidence score and we apply Non Maximum Suppression algorithm to merge overlapping detections. In the remainder of this section we are going to describe all steps in details.

10.2.1 Feature Extraction and Fisher Vector encoding with Integral Image

This step is invoked for each incoming frame. First we extract features for incoming frame. In features extraction step any method proposed in chapters 3, 4, 5, 6 and 7 can be applied. After the features are extracted we compute unnormalized Fisher Vector representation. If we look at Fisher Vector definition in eq. (3.7) and eq. (3.8) in chapter 3 we can notice that Fisher Vector is normalized by N_x which is the cardinality of the feature set $\mathbb{X}$. In our case $\mathbb{X}_t$ is a feature set for frame t. Unfortunately because of normalization term N_x:

$$G^{\mathbb{X}_t \bigcup \mathbb{X}_{t+1}} \neq G^{\mathbb{X}_t} + G^{\mathbb{X}_{t+1}}, \tag{10.1}$$

where $G^{\mathbb{X}_t}$ is a Fisher Vector representation for frame t. The property: $G^{\mathbb{X}_t \bigcup \mathbb{X}_{t+1}} = G^{\mathbb{X}_t} + G^{\mathbb{X}_{t+1}}$ is required to make a use of Integral Image representation. To solve this issue for each feature set $\mathbb{X}_t$ we compute Fisher Vector $\hat{G}^{\mathbb{X}}$ which is not normalized by N_{x_t} term.

Then we obtain the following property:

$$\hat{G}^{\mathbb{X}_t \bigcup \mathbb{X}_{t+1}} = \hat{G}^{\mathbb{X}_t} + \hat{G}^{\mathbb{X}_{t+1}}, \tag{10.2}$$

$$\tag{10.3}$$

Then to compute Fisher Vector representation of $G^{\mathbb{X}_t \bigcup \mathbb{X}_{t+1}}$ we can use the following equotation:

$$G^{\mathbb{X}_t \bigcup \mathbb{X}_{t+1}} = \frac{\hat{G}^{\mathbb{X}_t} + \hat{G}^{\mathbb{X}_{t+1}}}{N_{x_t} + N_{x_{t+1}}}, \tag{10.4}$$

The unnormalized Fisher Vector $\hat{G}^{\mathbb{X}_t}$ and N_{x_t} values are stored in Integral Image representation I_G and I_N respectively. The details about Integral Image are provided in section 10.2.4.

10.2.2 Temporal window proposals and classification

In the second step of our Action Detection framework we compute Fisher Vector for temporal windows, and then we classify each window obtaining action label and confidence score. Let's define temporal window of size n as W_t^{t+n}. Window W_t^{t+n} begins at frame t and ends at frame $t+n$. Temporal window W_t^{t+n} represents feature-set $\mathbb{X}_W = \mathbb{X}_t \bigcup \mathbb{X}_{t+1} \bigcup .. \bigcup \mathbb{X}_{t+n}$. To compute Fisher Vector of window W_t^{t+n} we take advantage of Integral Image representation. The Fisher Vector of window W_t^{t+n} can be defined as follows:

$$G^W = \frac{I_G(t+n) - I_G(t-1)}{I_N(t+n) - I_N(t-1)} \tag{10.5}$$

We would like to highlight that computation time of G^W is constant *w.r.t.* window size. This property is very important as it allows us to compute many temporal window overlapping window proposals on different temporal scales. G_W representation is then passed on SVM classifier, thus action label and classifier confidence score is obtained for each temporal window.

10.2.3 Window proposals pruning

In the final step we remove temporal windows which classification confidence is below defined threshold. But even after this step we might end up with many overlapping window proposals. To merge overlapping windows we use Non Maximum Suppression algorithm [30]. The algorithm clusters window proposals with a fixed distance threshold, to prune overlapping window proposals and keep best one according to distance threshold criterion. After this step we obtain list of non-overlapping window proposals with action labels assigned to each proposal, which is a final result of our Action Detection framework.

Table 10.1: Processing speed of methods proposed in this thesis i Frames Per Second (FPS).

Method	FPS
Dense Trajectories [113] (chapter 3)	10.57
3DTSD (chapter 4)	14.34
HOG_EB (chapter 5)	11.58
Brownian (chapter 6)	1.16
GridFV (chapter 7)	9.94

10.2.4 Integral Image

An Integral Image is a data structure and algorithm for quick and efficient generation of a sum of values in a rectangular subset of a grid. It is also known as "Summed Area table". Let's define $\boldsymbol{U}$ as a matrix which values will be encoded in Integral Image. The Integral Image I for matrix $\boldsymbol{U}$ can be defined as:

$$I_{\sum}(x,y) = \sum_{\substack{x' \leq x \\ y' \leq y}} U(x',y') \tag{10.6}$$

Once we obtain Integral Image representation we can compute sum of any rectangular area in constant time. Let's define $A = (x_0, y_0)$, $B = (x_1, y_0)$, $C = (x_0, y_1)$ and $D = (x_1, y_1)$, the sum of U over the rectangle spanned by A, B, C and D is equal to:

$$\sum_{\substack{x_0 < x \leq x_1 \\ y_0 < y \leq y_1}} U(x,y) = I(D) + I(A) - I(B) - I(C) \tag{10.7}$$

The original Integral Image method is designed to 2D matrices. In our Action Detection framework we propose to use 1D integral image. Let's define vector $\boldsymbol{u}$ which values will be encoded by 1D Integral Image I. Then we can define Integral Image as:

$$I_{\sum}(x) = \sum_{x' \leq x} u(x'). \tag{10.8}$$

The sum of $\boldsymbol{u}$ spanned over x_0 and x_1 is equal to:

$$\sum_{x_0 < x \leq x_1} u(x) = I(x_1) - I(x_0 - 1) \tag{10.9}$$

10.3 Experiments

In the first part of this section we will focus on processing time of each component of the action detection system. In table 10.1 we show processing speed for each method proposed in this thesis, as well as Dense Trajectories baseline. The evaluation was done of videos 640×480 pixels and we use custom implementation of Dense Trajectories method. The processing speed of the proposed methods is similar, because most processing time

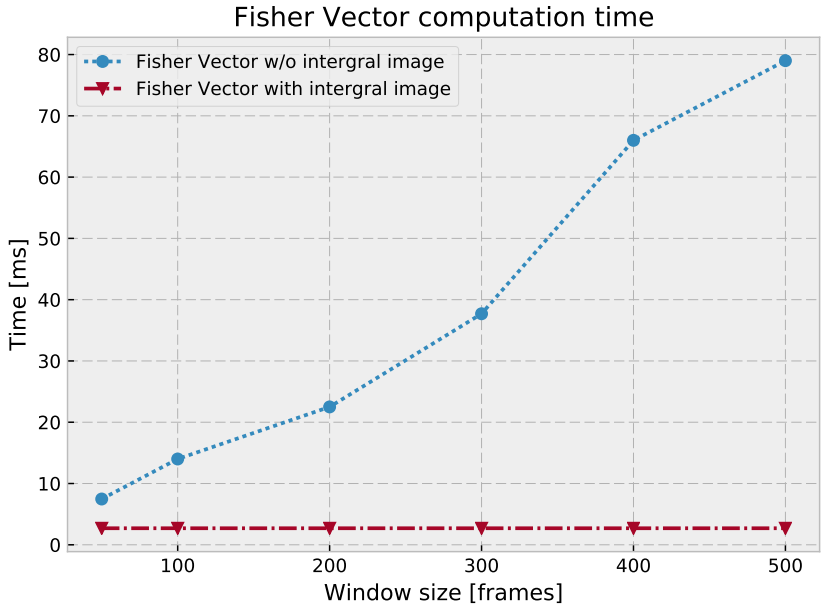


Figure 10.3: Comparison of computation time need to compute Fisher Vector representation *w.r.t.* to temporal window size. Time needed to compute Fisher Vector representation with integral image does not depend on window size.

is spent on optical flow computation, which is needed to detect local points of interest. GridFV method is slower due to the fact that people detection is needed and it already includes Fisher Vector computation step. Brownian descriptor is the slowest and yet not ready to be used in real-time application. The slow processing time of Brownian descriptor is directly related to sub-optimal implementation which does not use integral image. Brownian descriptor with integral image implementation should be able to achieve similar speed as other presented methods.

In fig. 10.3 we compare time needed to compute Fisher Vector representation for time windows of different size. We compare Fisher Vector computed with and without 1D integral image described in this chapter. The most important property of Fisher Vector implementation with integral image, is that processing time is constant and does not depend on window size.

Finally we have evaluated small subset of Smarthomes dataset with MBH descriptor and mentioned Fisher Vector representation with integral image. We have used temporal windows with 20, 50, 100, 200 size. The average processing speed was 8.22 fps. We achieved 0.33 precision score and 0.42 recall. The results were mostly bounded by False Positives rate, because many gaps between actions were detected as positive examples. The Smarthomes dataset is challenging in terms of detection, as gaps between actions

varies from couple of second to minutes. Also in the gaps between the action subject tend to move creating motion which is detected and additional local features are computed. We believe that simple model with background action class is not sufficient to handle this challenges. Thus semantic models such as ones, which consider context of person location or involved objects [18, 72], discover zones where actions take place [71, 19], might improve our action detection framework. We would like to highlight that the goal of this chapter was to explain details of action detection framework implemented as a part of this thesis. The system was successfully applied in Toyota robotics system, but in less challenging scenarios than one present in Smarthomes dataset. We believe that this chapter will lay a foundation for our future improvements and contributions in action detection.

Chapter 11

Conclusion

Contents

In this thesis we proposed and evaluated several methods for action recognition in videos, as well as we built a real-time action detection framework. Our experiments demonstrated that we outperformed state-of-the-art methods. We conclude our work pointing out key contributions (section 11.1) and their limitations (section 11.2). Finally we discuss short and long-term perspectives (section 11.3).

11.1 Key Contributions

3D Trajectories

We proposed novel descriptor based on 3D trajectory shape. Such a descriptor cannot be computed entirely on depth-map, due to lack of texture and problem with points of interest detection. We propose to compute 2D trajectory based on RGB information and then translate it to 3D space using depth-map and pinhole camera model.

Removing hallucinations by histogram normalization

We proposed new normalization schema, which removes "hallucination" effect for histogram based descriptors. The "hallucination" effect occurs when representation of two different entities becomes indistinguishable due to normalization. This problem is vital problem in action recognition, due to high popularity of histogram based descriptors

(HOG, HOG, MBH). We propose to remove this effect by adding extra bin to histogram, before normalization is applied. The value of the bin can be found by cross-validation or by proposed Expectation Maximization algorithm.

Brownian Covariance Descriptor

We have proposed low level descriptor based on Brownian Covariance. Such a descriptor models relations between low level features such as color intensities, gradients *etc.* Brownian Covariance is able to model all kind of relations including non-linear and non-monotone. With such descriptor we propose to use high level statistics comparing to histogram based descriptors.

Spatio-temporal Fisher Vector representation

In this method we propose to model spatial-layout of feature *w.r.t.* person bounding box. We propose three different methods for spatial-layout modeling: direct, fixed grid, mixture of Gaussians. The performance of local features is greatly improved, while we require much simpler and robust people detection, rather than skeleton detection. Our method is generic and can be used together with any local feature method.

Real-time Action Detection

We built action detection framework, which is able to work in real-time. The computational speed gain is achieved due to effective use of integral image representation for Fisher Vector encoding.

Evaluation on Public Datasets

We performed extensive evaluation of handcrafted (Dense Trajectories) and deep-learning based (Trajectory-Pooled Deep-Convolutional Descriptors) on daily-living action recognition datasets.

Smarthomes Datasets

We built new large scale action recognition dataset with 17 000 video clips. Actions are performed by 20 senior people. Dataset contains 28 actions performed in un-acted and un-supervised way. This way we introduced new challenges which were missing in many state-of-the-art datasets. The Smarthomes dataset can be also used in action detection evaluation and as it also contains long video sequences.

11.2 Limitations

The methods proposed in this thesis have still some limitations. Some of them can be seen as extensions and can be solved in near future. While others are still open questions. In this section we present the limitations while final section of this thesis provides discussion of short-term and long-term perspectives.

3D Trajectories
Although we show that proposed 3D trajectory shape descriptor can significantly outperform 2D trajectory shape descriptor, it requires reliable depth-map from the RGB-D sensors. If a person is far from the sensor the amount of noise in the depth map rises, which has serious impact on 3D trajectory shape descriptor performance.

Removing hallucinations by histogram normalization
Main limitation of this method, is that it can be only applied to histogram based descriptors.

Brownian Covariance Descriptor
In case of linear relationship – Brownian Covariance is is not able to distinguish if relationship is positive or negative.

Modeling Spatial Layout of Features with People Detection
The method proposed in this thesis assumes that there is only one person in the scene. In real-world scenarios, more people might be present on the scene. In particular some action involves more than one person – this case is not covered by the proposed method.

Modeling spatial layout only with person bounding box may not work when actions are done in horizontal position, where head is not on the top of the bounding box.

In addition our Mixture of Gaussian model concentrates too much on areas with high features density, which limits it's performance.

11.3 Future Work

11.3.1 Short-Term Perspectives

Removing hallucinations by histogram normalization
The proposed normalization method could be applied to other histogram based descriptors such as: HOF, MBH. We could provide extended analysis – when to use proposed expectation maximization algorithm over cross-validation.

Brownian Covariance Descriptor

We could explore the performance of Brownian Covariance descriptor applied to motion features *eg.* based on optical flow. We could also explore the relationships between feature maps obtained from CNNs with Brownian Covariance.

Modeling Spatial Layout of Features with People Detection

The method could be extended to cases where more than one person is involved in the action, by modeling spatio-temporal of features from different bounding boxes.

In addition our Mixture of Gaussian model concentrates too much on areas with high features density. This cause overfitting problem on datasets with small amount of data. Additional mechanism which let to control positions of inferred Gaussians or let prune them if they were to close to each other, should improve performance.

Real-time Action Detection

The extensive search with overlapping sliding windows, could be replaced with method which generates sparse set of action proposals. In addition the described system could be used together with methods which model global context and semantics *i.e.* action zones.

11.3.2 Long-Term Perspectives

Cross camera action recognition

Many public action recognition datasets introduce only one or only few camera viewpoints. It would be very interesting to evaluate how the proposed methods behave when they are tested on new camera viewpoint which was not available in training set.

Recognition of complex actions

The state-of-the-art methods achieve good performance on rather simple and short actions. Our next goal would be to recognize long complicated action such as: cooking or repairing the car. Such actions consist of many short actions, which can be performed in different order.

Recognition of similar actions

Recent stat-of-the-art methods perform well on datasets with rather big inter-class variance. Our goal would be to improve action recognition accuracy in cases where actions have very similar motion pattern and are performed in same environment. In such case the algorithm should be able to focus on details without overfitting the training data. The problem of recognition of similar actions is especially crucial in daily-living surveillance and robotics applications.

Attentions mechanism

Many state-of-the-art methods achieve very good performance on rather short actions. We believe that long, complex action recognition or action detection can be improved with attention mechanism. Such mechanism lets the algorithm focus on important subsequences of action.

Appendix A

Smarthomes dataset details

A.1 Introduction

The Smarthomes dataset was recorded in 2015, our goal was to record large scale dataset, with daily-living actions performed in most realistic way as possible. At that time biggest daily-living action datasets contained couple of hundreds of video clips, in addition they were recorded in quite constrained and controlled environment. In publicly available datasets action are performed by students, while one of the key application of daily-living action recognition is patient monitoring. Thus in our dataset age of people performing actions varies from 60-80 years. Throughout recording process we managed to gather more than 1000 hours of video footage. The manual annotation of the videos took more than 6 months, and Smarthomes dataset was ready at the end of 2016.

A.2 Recording setup

7 Kinect v1 sensors were used in the recording process, the apartment plan and camera location are available in fig. A.1. Cameras 1 and 2 cover the dining room area, 3 and 6 kitchen, 5 and 4 living room (see fig. A.4). Thus we have coverage of all room from 2 view angles. Camera 7 records kitchen work space from close distance. The videos are recorded at 20 frames per second, the size of RGB frame is 640×480, the size of depth-frame is 320×240.

Depth information suffers from high noise rate especially when a person is further than 3 meters from the sensor. Interference between Kinect is another source of noise. Finally shinny surfaces like floor or kitchen work space absorb IR beam thus depth information is not available (see fig. A.2). We have found that utilization of depth-map is quite challenging. This directly translates to skeleton detection problem. The Microsoft skeleton detection algorithm shipped with Kinect sensor fail in most of the cases. Finally

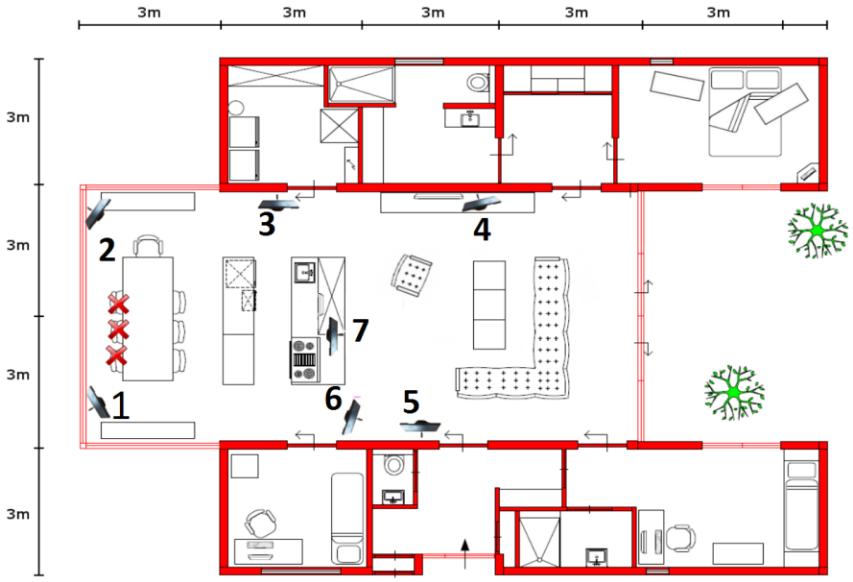


Figure A.1: Plan of the apartment where Smarthomes dataset was recorded. Numbers indicate Kinect sensors locations.

we managed to obtain skeleton in only 20% of frames.

A.3 Recording protocol

Each person was recorded for 8 hours in one day starting from morning until afternoon. The day was divided into 8 sessions 1 hour each. Before the session person was asked to perform selected action during the session. For instance, in the morning session a person was asked to prepare a breakfast and eat it. No further guidance was provided about how the action should be performed.

A.4 Annotation

The dataset was manually annotated to provide ground truth information. Each camera was annotated separately. The annotation process took more than 6 months, including verification and quality checks. Finally we obtained 16 974 video clips with 28 action classes. The distribution of action classes is available in fig. A.3. In the class distribution plot we can observe long tail of classes with low frequency. This situation is quite common in big datasets (*eg.* ImageNet [22]), as it is difficult to repeat some actions as many times as others. In addition, we did not want to interfere into the way and how often actions are performed, to keep them as realistic as possible.

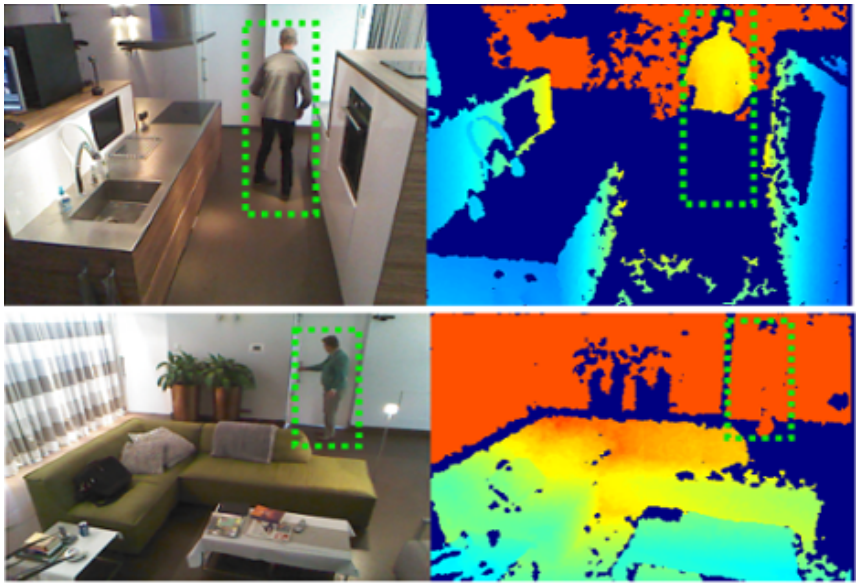


Figure A.2: Depth map quality in Smarthomes dataset. Dark blue color indicates missing depth values. Depth value is not available for a floor and a kitchen workspace, also depth information cannot be measured on black jeans (top row). In the bottom row person is too far from the sensor.

We also provide annotations for whole 1 hour sessions, in this way also action detection methods can by evaluated with the dataset.

Figure A.3: Frequency of action class instances in Smarthomes dataset.

A.5 Conclusion

In this chapter we presented details of our large scale daily-living Smarthomes dataset. We provide almost 17 000 video clips with 28 actions, 7 view points, 20 people performing actions. The dataset can be used to evaluate action recognition as well as action detection methods.

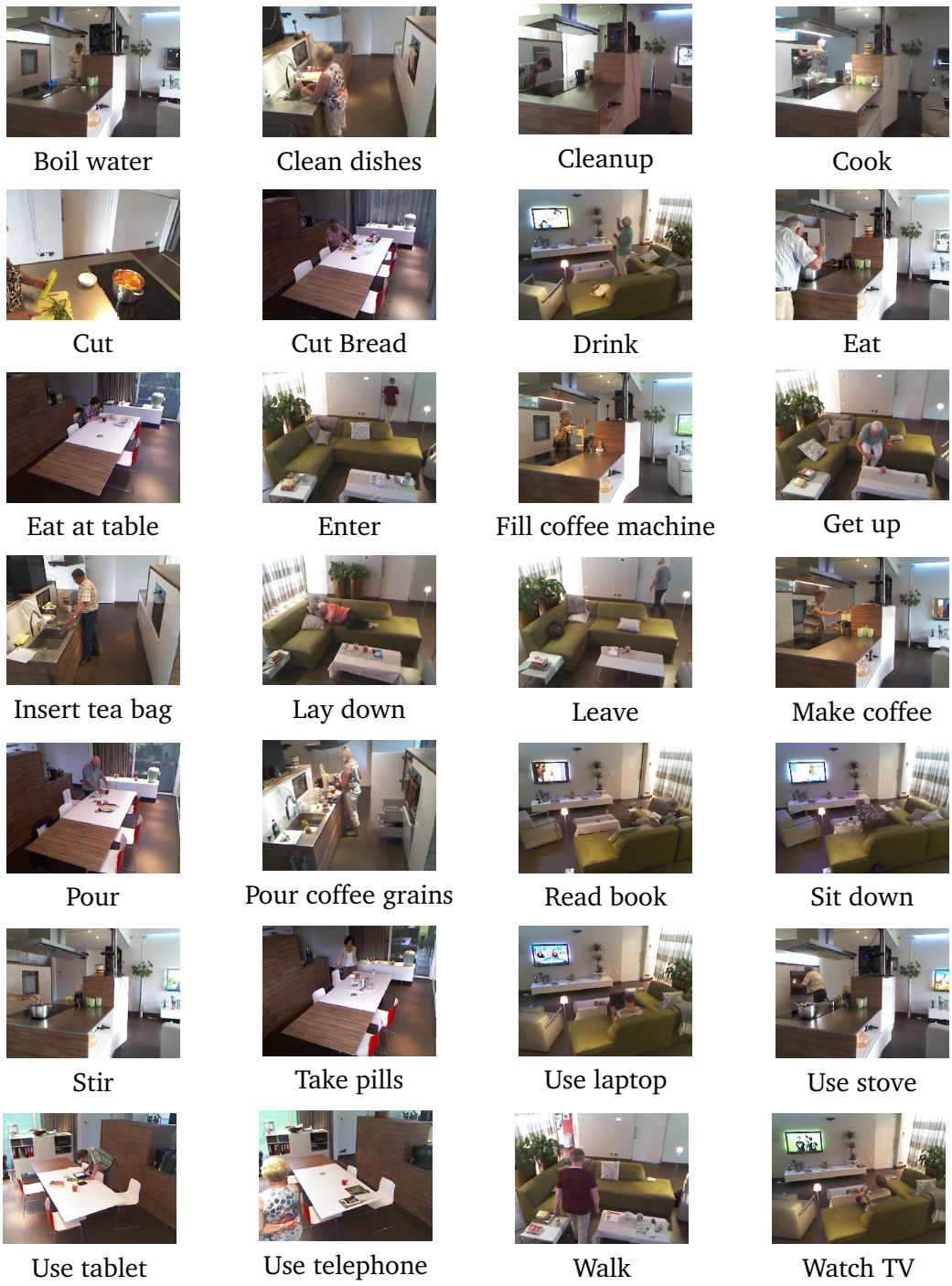


Figure A.4: Sample frames of each action class for Smarthomes dataset.

Bibliography

[1] AMOR, B. B., SU, J., AND SRIVASTAVA, A. Action recognition using rate-invariant analysis of skeletal shape trajectories. *IEEE Transactions on Pattern Analysis and Machine Intelligence 38*, 1 (Jan 2016), 1–13. (Cited on page 171.)

[2] BAK, S., BIAGIO, M. S., KUMAR, R., MURINO, V., AND BREMOND, F. Exploiting feature correlations by brownian statistics for people detection and recognition. *IEEE Transactions on Systems, Man, and Cybernetics: Systems PP*, 99 (2016), 1–12. (Cited on pages 98 and 99.)

[3] BAK, S., KUMAR, R., AND BREMOND, F. Brownian descriptor: a Rich Meta-Feature for Appearance Matching. In *WACV* (2014). (Cited on page 98.)

[4] BAKIROV, N. K., AND SZEKELY, G. J. Brownian covariance and central limit theorem for stationary sequences. *Theory of Probability and its Applications 55*, 3 (2011), 371–394. (Cited on page 98.)

[5] BAY, H., ESS, A., TUYTELAARS, T., AND VAN GOOL, L. Speeded-up robust features (surf). *Comput. Vis. Image Underst. 110*, 3 (June 2008), 346–359. (Cited on page 17.)

[6] BILINSKI, P., AND BREMOND, F. Contextual Statistics of Space-Time Ordered Features for Human Action Recognition. In *9th IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)* (Beijing, China, Sept. 2012). (Cited on page 17.)

[7] BILINSKI, P., AND BREMOND, F. Statistics of Pairwise Co-occurring Local Spatio-Temporal Features for Human Action Recognition. In *4th International Workshop on Video Event Categorization, Tagging and Retrieval (VECTaR), in conjunction with 12th European Conference on Computer Vision (ECCV)* (Florence, Italy, Oct. 2012), A. Fusiello, V. Murino, and R. Cucchiara, Eds., vol. 7583, Springer, pp. 311–320. (Cited on page 17.)

[8] BILINSKI, P., AND BREMOND, F. Video Covariance Matrix Logarithm for Human Action Recognition in Videos. In *IJCAI 2015 - 24th International Joint Conference on Artificial Intelligence (IJCAI)* (Buenos Aires, Argentina, July 2015). (Cited on page 17.)

[9] BILINSKI, P., CORVEE, E., BAK, S., AND BREMOND, F. Relative Dense Tracklets for Human Action Recognition. In *10th IEEE International Conference on Automatic Face and Gesture Recognition* (Shanghai, China, Apr. 2013), IEEE, pp. 1–7. (Cited on page 17.)

[10] BILINSKI, P., KOPERSKI, M., BAK, S., AND BREMOND, F. Representing visual appearance by video brownian covariance descriptor for human action recognition. In *AVSS* (Seoul, South Korea, Aug. 2014), IEEE. (Cited on pages 9 and 98.)

[11] BJORK, E., AND BJORK, R. *Memory: Handbook of Perception and Cognition*. Academic Press, 1996. (Cited on page 7.)

[12] BOBICK, A. F., AND DAVIS, J. W. The recognition of human movement using temporal templates. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE 23* (2001), 257–267. (Cited on page 14.)

[13] BUCH, S., ESCORCIA, V., SHEN, C., GHANEM, B., AND NIEBLES, J. C. SST: Single-stream temporal action proposals. In *CVPR* (2017). (Cited on page 176.)

[14] CAO, Z., SIMON, T., WEI, S.-E., AND SHEIKH, Y. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR* (2017). (Cited on pages 14, 56, 157 and 160.)

[15] CHENG, Z., QIN, L., YE, Y., HUANG, Q., AND TIAN, Q. Human daily action analysis with multi-view and color-depth data. In *Proceedings of the 12th International Conference on Computer Vision - Volume 2* (Berlin, Heidelberg, 2012), ECCV'12, Springer-Verlag, pp. 52–61. (Cited on page 20.)

[16] CHERON, G., LAPTEV, I., AND SCHMID, C. P-CNN: Pose-Based CNN Features for Action Recognition. pp. 3218–3226. (Cited on page 23.)

[17] CORTES, C., AND VAPNIK, V. Support-vector networks. *Machine Learning 20*, 3 (Sep 1995), 273–297. (Cited on page 34.)

[18] CRISPIM-JUNIOR, C., GOMEZ URIA, A., STRUMIA, C., KOPERSKI, M., KÃ¶NIG, A., NEGIN, F., COSAR, S., TUAN NGHIEM, A., CHAU, D. P., CHARPIAT, G., AND BREMOND, F. Online recognition of daily activities by color-depth sensing and knowledge models. 1528. (Cited on page 181.)

[19] CRISPIM-JUNIOR, C. F., KOPERSKI, M., COSAR, S., AND BREMOND, F. Semi-supervised understanding of complex activities from temporal concepts. In *13th International Conference on Advanced Video and Signal-Based Surveillance* (Colorado Springs, United States, Aug. 2016). (Cited on page 181.)

[20] DALAL, N., AND TRIGGS, B. Histograms of oriented gradients for human detection. In *CVPR* (2005). (Cited on pages 16, 17, 32, 73 and 74.)

[21] DAVIS, J. W., AND BOBICK, A. F. The representation and recognition of human movement using temporal templates. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)* (Washington, DC, USA, 1997), CVPR '97, IEEE Computer Society, pp. 928–. (Cited on page 15.)

[22] DENG, J., DONG, W., SOCHER, R., LI, L.-J., LI, K., AND FEI-FEI, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09* (2009). (Cited on pages 35 and 190.)

[23] DOLLÁR, P., RABAUD, V., COTTRELL, G., AND BELONGIE, S. Behavior recognition via sparse spatio-temporal features. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)* (Beijing, China, 2005). (Cited on pages 16 and 30.)

[24] DURAND, T., THOME, N., AND CORD, M. Mantra: Minimum maximum latent structural svm for image classification and ranking. In *2015 IEEE International Conference on Computer Vision (ICCV)* (Dec 2015), pp. 2713–2721. (Cited on page 34.)

[25] DURAND, T., THOME, N., CORD, M., AND PICARD, D. Incremental learning of latent structural svm for weakly supervised image classification. In *2014 IEEE International Conference on Image Processing (ICIP)* (Oct 2014), pp. 4246–4250. (Cited on page 34.)

[26] ELLIS, C., MASOOD, S. Z., TAPPEN, M. F., LAVIOLA, JR., J. J., AND SUKTHANKAR, R. Exploring the trade-off between accuracy and observational latency in action recognition. *Int. J. Comput. Vision 101*, 3 (Feb. 2013), 420–436. (Cited on pages 8, 14 and 21.)

[27] ESCORCIA, V., CABA HEILBRON, F., NIEBLES, J. C., AND GHANEM, B. *DAPs: Deep Action Proposals for Action Understanding*. 2016, pp. 768–784. (Cited on page 176.)

[28] FEICHTENHOFER, C., PINZ, A., AND ZISSERMAN, A. Convolutional Two-Stream Network Fusion for Video Action Recognition. pp. 1933–1941. (Cited on pages 18 and 19.)

[29] FELZENSZWALB, P. F., GIRSHICK, R. B., MCALLESTER, D., AND RAMANAN, D. Object detection with discriminatively trained part based models. *TPAMI* (2010). (Cited on page 73.)

[30] FELZENSZWALB, P. F., GIRSHICK, R. B., MCALLESTER, D., AND RAMANAN, D. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 9 (Sept 2010), 1627–1645. (Cited on page 178.)

[31] FERNANDO, B., GAVVES, E., ORAMAS, J. M., GHODRATI, A., AND TUYTELAARS, T. Modeling Video Evolution for Action Recognition. pp. 5378–5387. (Cited on page 19.)

[32] GIRSHICK, R. Fast r-cnn. In *International Conference on Computer Vision (ICCV)* (2015). (Cited on page 125.)

[33] GORELICK, L., BLANK, M., SHECHTMAN, E., IRANI, M., AND BASRI, R. Actions as space-time shapes. *IEEE Trans. Pattern Anal. Mach. Intell. 29*, 12 (Dec. 2007), 2247–2253. (Cited on pages 14, 15 and 19.)

[34] GOWAYYED, M. A., TORKI, M., HUSSEIN, M. E., AND EL-SABAN, M. Histogram of oriented displacements (hod): Describing trajectories of human joints for action recognition. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence* (2013), IJCAI '13, AAAI Press, pp. 1351–1357. (Cited on page 22.)

[35] HALIM, A. A., DARTIGUES-PALLEZ, C., PRECIOSO, F., RIVEILL, M., BENSLIMANE, A., AND GHONEIM, S. Human action recognition based on 3d skeleton part-based pose estimation and temporal multi-resolution analysis. In *2016 IEEE International Conference on Image Processing (ICIP)* (Sept 2016), pp. 3041–3045. (Cited on page 14.)

[36] HARRIS, C., AND STEPHENS, M. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference* (1988), pp. 147–151. (Cited on page 16.)

[37] HEILBRON, F. C., NIEBLES, J. C., AND GHANEM, B. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016), pp. 1914–1923. (Cited on page 176.)

[38] HU, J.-F., ZHENG, W.-S., LAI, J., AND ZHANG, J. Jointly learning heterogeneous features for RGB-D activity recognition. In *CVPR* (2015). (Cited on pages 167, 170 and 171.)

[39] HUSSEIN, M. E., TORKI, M., GOWAYYED, M. A., AND EL-SABAN, M. Human action recognition using a temporal hierarchy of covariance descriptors on 3d joint locations. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence* (2013), IJCAI '13, AAAI Press, pp. 2466–2472. (Cited on page 22.)

[40] INSAFUTDINOV, E., PISHCHULIN, L., ANDRES, B., ANDRILUKA, M., AND SCHIEKE, B. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In *European Conference on Computer Vision (ECCV)* (2016). (Cited on pages 21 and 56.)

[41] JI, S., XU, W., YANG, M., AND YU, K. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*, 1 (Jan 2013), 221–231. (Cited on page 18.)

[42] JOHANSSON, G. Visual perception of biological motion and a model for its analysis. *Perception & Psychophysics 14*, 2 (1973), 201–211. (Cited on pages 14, 21 and 157.)

[43] KOPERSKI, M., BILINSKI, P., AND BREMOND, F. 3d trajectories for action recognition. In *2014 IEEE International Conference on Image Processing (ICIP)* (Oct 2014), pp. 4176–4180. (Cited on pages 9 and 56.)

[44] KOPERSKI, M., AND BREMOND, F. Modeling Spatial Layout of Features for Real World Scenario RGB-D Action Recognition. In *AVSS 2016* (Colorado Springs, United States, Aug. 2016), pp. 44 – 50. (Cited on pages 8 and 9.)

[45] KOPPULA, H., AND SAXENA, A. Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation. In *ICML* (2013). (Cited on pages 168 and 169.)

[46] KOPPULA, H. S., GUPTA, R., AND SAXENA, A. Learning human activities and object affordances from rgb-d videos. *Int. J. Rob. Res. 32*, 8 (July 2013), 951–970. (Cited on pages 22, 167 and 169.)

[47] KRAPAC, J., VERBEEK, J., AND JURIE, F. Modeling Spatial Layout with Fisher Vectors for Image Categorization. In *ICCV* (2011). (Cited on page 122.)

[48] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. (Cited on page 17.)

[49] KUEHNE, H., JHUANG, H., GARROTE, E., POGGIO, T., AND SERRE, T. HMDB: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)* (2011). (Cited on pages 7, 18, 19 and 24.)

[50] LAN, Z., LIN, M., LI, X., HAUPTMANN, A. G., AND RAJ, B. Beyond Gaussian Pyramid: Multi-Skip Feature Stacking for Action Recognition. pp. 204–212. (Cited on page 19.)

[51] LAPTEV, I. On space-time interest points. *Int. J. Comput. Vision 64*, 2-3 (Sept. 2005), 107–123. (Cited on pages 14, 15, 16, 17, 30 and 34.)

[52] LAPTEV, I., MARSZALEK, M., SCHMID, C., AND ROZENFELD, B. Learning realistic human actions from movies. In *2008 IEEE Conference on Computer Vision and Pattern Recognition* (June 2008), pp. 1–8. (Cited on pages 14, 16, 29, 78 and 118.)

[53] LAZEBNIK, S., SCHMID, C., AND PONCE, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR* (2006). (Cited on page 14.)

[54] LEIGHTLEY, D. *3D Human Action Recognition and Motion Analysis using Selective Representations*. PhD thesis, Manchester Metropolitan University, 2015. (Cited on page 13.)

[55] LI, W., ZHANG, Z., AND LIU, Z. *Action Recognition Based on a Bag of 3D Points*. IEEE, June 2010. (Cited on page 19.)

[56] LIN, L., WANG, K., ZUO, W., WANG, M., LUO, J., AND ZHANG, L. A deep structured model with radius–margin bound for 3d human activity recognition. *International Journal of Computer Vision 118*, 2 (Jun 2016), 256–273. (Cited on pages 168 and 169.)

[57] LIU, L., AND SHAO, L. Learning discriminative representations from rgb-d video data. In *IJCAI* (2013). (Cited on page 171.)

[58] LIU, T., WANG, X., DAI, X., AND LUO, J. Deep recursive and hierarchical conditional random fields for human action recognition. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)* (March 2016), pp. 1–9. (Cited on page 169.)

[59] LIU, W., ANGUELOV, D., ERHAN, D., SZEGEDY, C., REED, S. E., FU, C.-Y., AND BERG, A. C. Ssd: Single shot multibox detector. In *ECCV* (2016). (Cited on pages 125 and 160.)

[60] LOWE, D. G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision 60*, 2 (Nov. 2004), 91–110. (Cited on page 17.)

[61] LU, C., JIA, J., AND TANG, C.-K. Range-sample depth feature for action recognition. In *CVPR* (2014). (Cited on pages 170 and 171.)

[62] LUCAS, B. D., AND KANADE, T. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2* (San Francisco, CA, USA, 1981), IJCAI'81, Morgan Kaufmann Publishers Inc., pp. 674–679. (Cited on page 17.)

[63] LV, F., AND NEVATIA, R. Recognition and segmentation of 3-d human action using hmm and multi-class adaboost. In *Proceedings of the 9th European Conference on Computer Vision - Volume Part IV* (Berlin, Heidelberg, 2006), ECCV'06, Springer-Verlag, pp. 359–372. (Cited on pages 14, 15 and 22.)

[64] MA, S., SIGAL, L., AND SCLAROFF, S. Space-time tree ensemble for action recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015), pp. 5024–5032. (Cited on page 22.)

[65] MA, S., SIGAL, L., AND SCLAROFF, S. Learning activity progression in lstms for activity detection and early detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016), pp. 1942–1950. (Cited on page 18.)

[66] MARGOLIN, R., ZELNIK-MANOR, L., AND TAL, A. Otc: A novel local descriptor for scene classification. In *ECCV* (2014). (Cited on page 75.)

[67] MATIKAINEN, P., HEBERT, M., AND SUKTHANKAR, R. Trajectons: Action recognition through the motion analysis of tracked features. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops* (Sept 2009), pp. 514–521. (Cited on pages 14 and 17.)

[68] MESSING, R., PAL, C., AND KAUTZ, H. Activity recognition using the velocity histories of tracked keypoints. In *2009 IEEE 12th International Conference on Computer Vision* (Sept 2009), pp. 104–111. (Cited on page 17.)

[69] MINETTO, R., THOME, N., CORD, M., LEITE, N. J., AND STOLFI, J. T-hog: An effective gradient-based descriptor for single line text regions. *Pattern Recogn. 46*, 3 (Mar. 2013), 1078–1090. (Cited on page 73.)

[70] MOESLUND, T. B., HILTON, A., AND KRÜGER, V. A survey of advances in vision-based human motion capture and analysis. *Comput. Vis. Image Underst. 104*, 2 (Nov. 2006), 90–126. (Cited on page 2.)

[71] NEGIN, F., COGAR, S., BREMOND, F., AND KOPERSKI, M. Generating unsupervised models for online long-term daily living activity recognition. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)* (Nov 2015), pp. 186–190. (Cited on page 181.)

[72] NEGIN, F. F., COSAR, S., KOPERSKI, M. F., CRISPIM-JUNIOR, C. F., AVGERINAKIS, K., AND BREMOND, F. F. A hybrid framework for online recognition of activities of daily living in real-world settings. In *13th IEEE International Conference on Advanced Video and Signal Based Surveillance - AVSS 2016* (Colorado springs, United States, Aug. 2016), IEEE. (Cited on page 181.)

[73] NI, B., MOULIN, P., AND YAN, S. Order-preserving sparse coding for sequence classification. In *ECCV* (2012). (Cited on page 167.)

[74] NI, B., WANG, G., AND MOULIN, P. Rgbd-hudaact: A color-depth video database for human daily activity recognition. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)* (Nov 2011), pp. 1147–1153. (Cited on pages 15 and 19.)

[75] OFLI, F., CHAUDHRY, R., KURILLO, G., VIDAL, R., AND BAJCSY, R. Sequence of the most informative joints (smij): A new representation for human skeletal action recognition. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* (June 2012), pp. 8–13. (Cited on pages 21 and 22.)

[76] OHN-BAR, E., AND TRIVEDI, M. M. Joint angles similarities and hog2 for action recognition. In *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops* (June 2013), pp. 465–470. (Cited on pages 8, 14 and 21.)

[77] OIKONOMOPOULOS, A., PATRAS, I., AND PANTIC, M. Spatiotemporal salient points for visual recognition of human actions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 36*, 3 (June 2005), 710–719. (Cited on page 16.)

[78] OREIFEJ, O., AND LIU, Z. Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. In *2013 IEEE Conference on Computer Vision and Pattern Recognition* (June 2013), pp. 716–723. (Cited on pages 20, 117, 167 and 171.)

[79] ORTIZ, J., BAK, S., KOPERSKI, M., AND BRÉMOND, F. Minimizing hallucination in Histogram of Oriented Gradients. In *The 12th IEEE International Conference on Advanced Video and Signal-based* (Karlsruhe, Germany, Aug. 2015). (Cited on page 9.)

[80] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., AND DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830. (Cited on page 35.)

[81] PERRONNIN, F., LIU, Y., SANCHEZ, J., AND POIRIER, H. Large-Scale Image Retrieval with Compressed Fisher Vectors. In *CVPR* (2010). (Cited on pages 33, 34 and 121.)

[82] PERRONNIN, F., SANCHEZ, J., AND MENSINK, T. Improving the Fisher Kernel for Large-Scale image classification. In *ECCV* (2010). (Cited on pages 33, 34, 78 and 121.)

[83] PISHCHULIN, L., INSAFUTDINOV, E., TANG, S., ANDRES, B., ANDRILUKA, M., GEHLER, P., AND SCHIELE, B. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016). (Cited on pages 14 and 21.)

[84] POPPE, R. Vision-based human motion analysis: An overview. *Comput. Vis. Image Underst. 108*, 1-2 (Oct. 2007), 4–18. (Cited on page 15.)

[85] PORIKLI, F., TUZEL, O., AND MEER, P. Covariance tracking using model update based means on riemannian manifolds. In *CVPR* (2006). (Cited on page 98.)

[86] QIAN, H., MAO, Y., XIANG, W., AND WANG, Z. Recognition of human activities using svm multi-class classifier. *Pattern Recogn. Lett. 31*, 2 (Jan. 2010), 100–111. (Cited on page 15.)

[87] RAHMANI, H., MAHMOOD, A., Q HUYNH, D., AND MIAN, A. *HOPC: Histogram of Oriented Principal Components of 3D Pointclouds for Action Recognition*. Springer International Publishing, Cham, 2014, pp. 742–757. (Cited on page 20.)

[88] RAMAN, N. *Action Recognition in Depth Videos using Nonparametric Probabilistic Graphical Models*. PhD thesis, University of London, 2016. (Cited on page 13.)

[89] RAZAVIAN, A. S., AZIZPOUR, H., SULLIVAN, J., AND CARLSSON, S. Cnn features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops* (Washington, DC, USA, 2014), CVPRW '14, IEEE Computer Society, pp. 512–519. (Cited on page 17.)

[90] REN, S., HE, K., GIRSHICK, R., AND SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information*

Processing Systems 28 (2015), C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., pp. 91–99. (Cited on page 125.)

[91] RYBOK, L., SCHAUERTE, B., AL-HALAH, Z., AND STIEFELHAGEN, R. Important stuff, everywhere! activity recognition with salient proto-objects as context. In *WACV* (2014). (Cited on page 169.)

[92] SABANADESAN, U. *Human Action Recognition from Video Sequences*. PhD thesis, Queensland University of Technology, 2016. (Cited on page 13.)

[93] SCOVANNER, P., ALI, S., AND SHAH, M. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th ACM International Conference on Multimedia* (New York, NY, USA, 2007), MM '07, ACM, pp. 357–360. (Cited on page 17.)

[94] SEIDENARI, L., VARANO, V., BERRETTI, S., DEL BIMBO, A., AND PALA, P. Recognizing actions from depth cameras as weakly aligned multi-part bag-of-poses. In *CVPRW* (2013). (Cited on pages 170 and 171.)

[95] SEMPENA, S., MAULIDEVI, N. U., AND ARYAN, P. R. Human action recognition using dynamic time warping. In *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics* (July 2011), pp. 1–5. (Cited on page 21.)

[96] SHAHROUDY, A., LIU, J., NG, T.-T., AND WANG, G. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016). (Cited on page 24.)

[97] SHAHROUDY, A., WANG, G., AND NG, T.-T. Multi-modal feature fusion for action recognition in rgb-d sequences. In *ISCCSP* (2014). (Cited on page 171.)

[98] SHOTTON, J., SHARP, T., KIPMAN, A., FITZGIBBON, A., FINOCCHIO, M., BLAKE, A., COOK, M., AND MOORE, R. Real-time human pose recognition in parts from single depth images. *Commun. ACM 56*, 1 (Jan. 2013), 116–124. (Cited on pages 14 and 21.)

[99] SHOU, Z., WANG, D., AND CHANG, S.-F. Temporal action localization in untrimmed videos via multi-stage cnns. In *CVPR* (2016). (Cited on page 176.)

[100] SIMONYAN, K., AND ZISSERMAN, A. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems 27* (2014), Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., Curran Associates, Inc., pp. 568–576. (Cited on pages 14, 15, 18, 19 and 35.)

[101] SIMONYAN, K., AND ZISSERMAN, A. Two-stream convolutional networks for action recognition in videos. In *Proceedings of the 27th International Conference on Neural Information Processing Systems* (Cambridge, MA, USA, 2014), NIPS'14, MIT Press, pp. 568–576. (Cited on page 18.)

[102] SIMONYAN, K., AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014). (Cited on page 35.)

[103] SOOMRO, K., ZAMIR, A. R., AND SHAH, M. Ucf101: A dataset of 101 human actions classes from videos in the wild. *CoRR abs/1212.0402* (2012). (Cited on pages 7, 18, 19 and 24.)

[104] SPINELLO, L., AND ARRAS, K. O. People detection in rgb-d data. In *IROS* (2011). (Cited on page 125.)

[105] SRIJAN, D., KOPERSKI, M., BREMOND, F., AND FRANCESCA, G. Action recognition based on a mixture of rgb and depth based skeleton. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* (Aug 2017), pp. 228–234. (Cited on pages 166, 167, 168, 169 and 171.)

[106] SUN, J., WU, X., YAN, S., CHEONG, L. F., CHUA, T. S., AND LI, J. Hierarchical spatio-temporal context modeling for action recognition. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (June 2009), pp. 2004–2011. (Cited on page 17.)

[107] SUNG, J., PONCE, C., SELMAN, B., AND SAXENA, A. Unstructured human activity detection from rgbd images. In *ICRA* (2012). (Cited on pages 10, 23, 24 and 167.)

[108] SZEKELY, G. J., AND RIZZO, M. L. Brownian distance covariance. *The Annals of Applied Statistics 3*, 4 (2009), 1236–1265. (Cited on pages 98, 99 and 100.)

[109] TAN, P.-N., STEINBACH, M., AND KUMAR, V. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005. (Cited on pages 34 and 35.)

[110] VAPNIK, V. *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982. (Cited on page 34.)

[111] VIEIRA, A. W., NASCIMENTO, E. R., OLIVEIRA, G. L., LIU, Z., AND CAMPOS, M. F. M. *STOP: Space-Time Occupancy Patterns for 3D Action Recognition from Depth*

Map Sequences. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 252–259. (Cited on page 19.)

[112] VONDRICK, C., KHOSLA, A., MALISIEWICZ, T., AND TORRALBA, A. Hoggles: Visualizing object detection features. *ICCV* (2013). (Cited on page 73.)

[113] WANG, H., KLASER, A., SCHMID, C., AND LIU, C. L. Action recognition by dense trajectories. In *CVPR 2011* (June 2011), pp. 3169–3176. (Cited on pages 10, 14, 16, 17, 23, 29, 30, 34, 56, 57, 78, 103, 118, 167, 168, 169, 170, 171, 173 and 179.)

[114] WANG, H., AND SCHMID, C. Action recognition with improved trajectories. In *2013 IEEE International Conference on Computer Vision* (Dec 2013), pp. 3551–3558. (Cited on pages 14, 15, 17 and 19.)

[115] WANG, J., LIU, Z., AND WU, Y. *Random Occupancy Patterns*. Springer International Publishing, Cham, 2014, pp. 41–55. (Cited on pages 8, 20 and 21.)

[116] WANG, J., LIU, Z., WU, Y., AND YUAN, J. Mining actionlet ensemble for action recognition with depth cameras. In *2012 IEEE Conference on Computer Vision and Pattern Recognition* (June 2012), pp. 1290–1297. (Cited on pages 10, 20, 22, 23, 24, 78, 117, 166, 167 and 171.)

[117] WANG, L., QIAO, Y., AND TANG, X. Action recognition with trajectory-pooled deep-convolutional descriptors. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015), pp. 4305–4314. (Cited on pages 10, 14, 15, 17, 18, 19, 30, 35, 167, 169, 171 and 173.)

[118] WANG, L., XIONG, Y., WANG, Z., AND QIAO, Y. Towards Good Practices for Very Deep Two-Stream ConvNets. *arXiv:1507.02159 [cs]* (July 2015). arXiv: 1507.02159. (Cited on page 19.)

[119] WANG, L., XIONG, Y., WANG, Z., QIAO, Y., LIN, D., TANG, X., AND VAN GOOL, L. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. *arXiv:1608.00859 [cs]* (Aug. 2016). arXiv: 1608.00859. (Cited on page 18.)

[120] WANG, P., LI, W., GAO, Z., TANG, C., ZHANG, J., AND OGUNBONA, P. ConvNets-Based Action Recognition from Depth Maps Through Virtual Cameras and Pseudo-coloring. In *Proceedings of the 23rd ACM International Conference on Multimedia* (New York, NY, USA, 2015), MM '15, ACM, pp. 1119–1122. (Cited on page 21.)

[121] WANG, P., LI, W., GAO, Z., ZHANG, J., TANG, C., AND OGUNBONA, P. O. Action recognition from depth maps using deep convolutional neural networks. *IEEE*

Transactions on Human-Machine Systems 46, 4 (Aug 2016), 498–509. (Cited on page 21.)

[122] WANG, P., LI, W., GAO, Z., ZHANG, J., TANG, C., AND OGUNBONA, P. O. Action recognition from depth maps using deep convolutional neural networks. *IEEE Transactions on Human-Machine Systems 46*, 4 (Aug 2016), 498–509. (Cited on page 171.)

[123] WANG, X., THOME, N., AND CORD, M. Gaze latent support vector machine for image classification. In *2016 IEEE International Conference on Image Processing (ICIP)* (Sept 2016), pp. 236–240. (Cited on page 34.)

[124] WEI, S.-E., RAMAKRISHNA, V., KANADE, T., AND SHEIKH, Y. Convolutional pose machines. In *CVPR* (2016). (Cited on pages 14, 21 and 56.)

[125] WILLEMS, G., TUYTELAARS, T., AND VAN GOOL, L. *An Efficient Dense and Scale-Invariant Spatio-Temporal Interest Point Detector*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 650–663. (Cited on pages 16 and 17.)

[126] WONG, S. F., AND CIPOLLA, R. Extracting spatiotemporal interest points using global information. In *2007 IEEE 11th International Conference on Computer Vision* (Oct 2007), pp. 1–8. (Cited on page 16.)

[127] WU, Z., WANG, X., JIANG, Y.-G., YE, H., AND XUE, X. Modeling Spatial-Temporal Clues in a Hybrid Deep Learning Framework for Video Classification. In *Proceedings of the 23rd ACM International Conference on Multimedia* (New York, NY, USA, 2015), MM '15, ACM, pp. 461–470. (Cited on pages 14, 15 and 19.)

[128] XIA, L. *Recognizing Human Activity Using RGBD Data*. PhD thesis, University of Texas, 2014. (Cited on page 13.)

[129] XIA, L., AND AGGARWAL, J. K. Spatio-temporal depth cuboid similarity feature for activity recognition using depth camera. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition* (Washington, DC, USA, 2013), CVPR '13, IEEE Computer Society, pp. 2834–2841. (Cited on page 171.)

[130] XIA, L., CHEN, C.-C., AND AGGARWAL, J. K. View invariant human action recognition using histograms of 3d joints. In *CVPR Workshops* (2012), IEEE Computer Society, pp. 20–27. (Cited on pages 8, 21 and 22.)

[131] XU, B., FU, Y., JIANG, Y.-G., LI, B., AND SIGAL, L. Video emotion recognition with transferred deep feature encodings. In *Proceedings of the 2016 ACM on International*

Conference on Multimedia Retrieval (New York, NY, USA, 2016), ICMR '16, ACM, pp. 15–22. (Cited on page 18.)

[132] YANG, X., AND TIAN, Y. Eigenjoints-based action recognition using naÃ¯ve-bayes-nearest-neighbor. In *CVPR Workshops* (2012), IEEE Computer Society, pp. 14–19. (Cited on pages 14 and 21.)

[133] YANG, X., AND TIAN, Y. Super normal vector for activity recognition using depth sequences. In *CVPR* (2014). (Cited on page 171.)

[134] YANG, X., AND TIAN, Y. Super normal vector for human activity recognition with depth cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence 39*, 5 (May 2017), 1028–1039. (Cited on page 171.)

[135] YANG, X., ZHANG, C., AND TIAN, Y. Recognizing actions using depth motion maps-based histograms of oriented gradients. In *Proceedings of the 20th ACM International Conference on Multimedia* (New York, NY, USA, 2012), MM '12, ACM, pp. 1057–1060. (Cited on page 19.)

[136] YU, E., AND AGGARWAL, J. *Human action recognition with extremities as semantic posture representation*. 2009, pp. 1–8. (Cited on page 15.)

[137] ZAIDENBERG, S., BILINSKI, P., AND BREMOND, F. Towards Unsupervised Sudden Group Movement Discovery for Video Surveillance. In *VISAPP - 9th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - 2014* (Lisbon, Portugal, Jan. 2014), S. Battiato, Ed., Jose Braz, SCITEPRESS Digital Library. (Cited on page 17.)

[138] ZHA, S., LUISIER, F., ANDREWS, W., SRIVASTAVA, N., AND SALAKHUTDINOV, R. Exploiting Image-trained CNN Architectures for Unconstrained Video Classification. *arXiv:1503.04144 [cs]* (Mar. 2015). arXiv: 1503.04144. (Cited on page 19.)

[139] ZHANG, J., MARSZAŁEK, M., LAZEBNIK, S., AND SCHMID, C. Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision 73*, 2 (Jun 2007), 213–238. (Cited on page 35.)

[140] ZHAO, Y., LIU, Z., YANG, L., AND CHENG, H. Combing rgb and depth map features for human activity recognition. In *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference* (Dec 2012), pp. 1–4. (Cited on page 22.)

[141] ZHU, P., HU, W., LI, L., AND WEI, Q. *Human Activity Recognition Based on R Transform and Fourier Mellin Transform*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 631–640. (Cited on page 15.)

[142] ZHU, Y., CHEN, W., AND GUO, G. Evaluating spatiotemporal interest point features for depth-based action recognition. *Image and Vision Computing 32*, 8 (2014), 453 – 464. (Cited on pages 167 and 171.)