THESIS

In order to obtain the PhD Diploma of

Computer Science

Prepared in the laboratory IRSEEM (ESIGELEC) and the department IA of IMT Lille Douai

Fast and accurate human action recognition using RGB-D cameras

Presented and written by
Enjie GHORBEL

Thesis publicly defended on October the 12th of 2017 in front of the jury composed of		
Mr. François Brémond	Research Director, INRIA Sophia Antipolis	Reviewer
Mrs. Saïda Bouakaz	Professor, Université Claude Bernard Lyon 1	Reviewer
Mr. Christian Wolf	Associate professor, INSA Lyon	Examinator
Mrs. Danielle Nuzillard	Professor, Université Reims Champagne Ardenne	Examinator
Mr. Xavier Savatier	Researcher Teacher, ESIGELEC	Director
Mr. Stéphane Lecoeuche	Professor, IMT Lille Douai	Co-Director
Mr. Rémi Boutteau	Researcher Teacher, ESIGELEC	Co-Supervisor
Mr. Jacques Boonaert	Associate Professor, IMT Lille Douai	Co-Supervisor

Thesis directed by Xavier SAVATIER, IRSEEM (ESIGELEC) and Stéphane LECOEUCHE, IA department (IMT Lille Douai)

Résumé

Récemment, les caméras RGB-D ont été introduites sur le marché et ont permis l'exploration de nouvelles approches de reconnaissance d'actions par l'utilisation de deux modalités autres que les images RGB, à savoir, les images de profondeur et les séquences de squelette. Généralement, ces approches ont été évaluées en termes de taux de reconnaissance. Cette thèse s'intéresse principalement à la reconnaissance rapide d'actions à partir de caméras RGB-D. Le travail a été focalisé sur une amélioration conjointe de la rapidité de calcul et du taux de reconnaissance en vue d'une application temps-réel.

Dans un premier temps, nous menons une étude comparative des méthodes existantes de reconnaissance d'actions basées sur des caméras RGB-D en utilisant les deux critères énoncés : le taux de reconnaissance et la rapidité de calcul. Suite aux conclusions résultant de cette étude, nous introduisons un nouveau descripteur de mouvement, à la fois précis et rapide, qui se base sur l'interpolation par splines cubiques de valeurs cinématiques du squelette, appelé *Kinematic Spline Curves (KSC)*. De plus, afin de pallier les effets négatifs engendrés par la variabilité anthropométrique, la variation d'orientation et la variation de vitesse, des méthodes de normalisation spatiale et temporelle rapides ont été proposées. Les expérimentations menées sur quatre bases de données prouvent la précision et la rapidité de ce descripteur.

Dans un second temps, un deuxième descripteur appelé *Hiearchical Kinematic Coavarince (HKC)* est introduit. Ce dernier est proposé dans l'optique de résoudre la question de reconnaissance rapide en ligne. Comme ce descripteur n'appartient pas à un espace euclidien, mais à l'espace des matrices Symétriques semi-Définies Positives (SsDP), nous adaptons les méthodes de classification à noyau par l'introduction d'une distance inspirée de la distance Log-Euclidienne, que nous appelons distance Log-Euclidienne modifiée. Cette extension nous permet d'utiliser des classifieurs adaptés à l'espace de caractéristiques (SPsD). Une étude expérimentale montre l'efficacité de cette méthode non seulement en termes de rapidité de calcul et de précision, mais également en termes de latence observationnelle. Ces conclusions prouvent que cette approche jointe à une méthode de segmentation d'actions pourrait s'avérer adaptée à la reconnaissance en ligne et ouvre ainsi de nouvelles perspectives pour nos travaux futurs.

Abstract

The recent availability of RGB-D cameras has renewed the interest of researchers in the topic of human action recognition. More precisely, several action recognition methods have been proposed based on the novel modalities provided by these cameras, namely, depth maps and skeleton sequences. These approaches have been mainly evaluated in terms of recognition accuracy. This thesis aims to study the issue of fast action recognition from RGB-D cameras. It focuses on proposing an action recognition method realizing a trade-off between accuracy and latency for the purpose of applying it in real-time scenarios.

As a first step, we propose a comparative study of recent RGB-D based action recognition methods using the two cited criteria: accuracy of recognition and rapidity of execution. Then, oriented by the conclusions stated thanks to our study, we introduce a novel, fast and accurate human action descriptor called Kinematic Spline Curves (KSC). This latter is based on the cubic spline interpolation of kinematic values. Moreover, fast spatial and temporal normalizations are proposed in order to overcome anthropometric variability, orientation variation and rate variability. The experiments carried out on four different benchmarks show the effectiveness of this approach in terms of execution time and accuracy.

As a second step, another descriptor is introduced, called Hierarchical Kinematic Covariance (HKC). This latter is proposed in order to solve the issue of fast online action recognition. Since this descriptor does not belong to a Euclidean space, but is an element of the space of Symmetric Positive semi-definite (SPsD) matrices, we adapt kernel classification methods by the introduction of a novel distance called Modified Log-Euclidean, which is inspired from Log-Euclidean distance. This extension allows us to use suitable classifiers to the feature space SPsD of matrices. The experiments prove the efficiency of our method, not only in terms of rapidity of calculation and accuracy, but also in terms of observational latency. These conclusions show that this approach combined with an action segmentation method could be appropriate to online recognition, and consequently, opens up new prospects for future works.

Acknowledgments

Firstly, I would like to express my sincere gratitude to my directors Prof. Xavier SAVATIER and Prof. Stéphane LECOEUCHE and my supervisors Dr. Rémi BOUTTEAU and Dr. Jacques BOONAERT for their continuous support of my Ph.D study and related research, for their patience, motivation, and knowledge. Their guidance helped me in all the time of research and writing of this thesis.

Besides my advisors, I would like to thank the rest of my thesis committee: Prof. Danielle Nuzillard and Dr. Christian Wolf for accepting to examine my thesis. Also, I would like to thank more particularly Dr. Prof. Francois Brémond and Prof. Saida Bouakaz, for their insightful comments, and interesting questions which participated to improve the quality of this manuscript and to widen my research from various perspectives.

My sincere thanks also goes to my colleagues and friends (Anis, Berba, Kawther, Safa, Fabien, Lavinius, Balsam, Ziad, Monica, Pau, Pablo, Tony) for the good ambiance and for making my stay in Douai and Rouen so perfect.

Finally, I would like to thank my parents, Ahmed and Nana for always being here during the good but also the hard moments. I would have never made it without you.

Synthèse en français

0.1 Introduction

La reconnaissance d'actions suscite, de plus en plus, l'intérêt de la communauté scientifique du domaine en raison de son large champ applicatif. Bien que cette tâche soit intuitive pour les humains, son automatisation par le biais d'outils informatiques n'en demeure pas moins complexe.

La majorité des méthodes proposées dans la littérature se sont inspirées de la structure anatomique humaine. Afin de capturer le flux de données relié au mouvement, l'homme utilise le plus souvent son sens de la vue. Ce flux d'informations est alors transféré au cerveau où il est analysé. En se basant sur des actions mémorisées par le cerveau, la scène observée est associée à une étiquette (le nom de l'action).

Ainsi, les méthodes de reconnaissance d'actions automatiques se basent généralement sur des capteurs visuels qui s'inspirent du système visuel humain. Ces derniers acquièrent l'information visuelle et la code de manière conventionnelle. Suite à cela des algorithmes de vision par ordinateur permettent l'extraction de caractéristiques discriminantes. Finalement, des méthodes de *machine learning* sont utilisées pour reconnaître les actions grâce aux caractéristiques extraites.

Plusieurs problématiques commencent déjà à apparaître : Quel type de capteur doit-on utiliser? Comment modéliser les actions et quels types de caractéristiques doit-on extraire? Quels sont les critères de performance adéquats qui permettent la comparaison des différentes méthodes? Quelles sont les limitations de chaque méthode?

Le but de cette thèse est donc d'analyser ces diverses questions et par conséquent d'y proposer des réponses adéquates et cohérentes.

Dans ce qui suit, nous présentons brièvement les motivations qui ont données lieu à nos travaux de recherche, les contributions scientifiques proposées, ainsi que l'organisation de ce manuscrit.

0.1.1 Motivations

De nos jours, les algorithmes de reconnaissance d'actions sont de plus en plus exploités vu leur utilité dans une grande variété d'applications (vidéo surveillance, santé, jeux vidéos, etc).

Les méthodes les plus classiques s'appuient sur l'utilisation de caméras RGB. Cependant, ce type de capteur présente certaines limitations telles que la sensibilité à la segmentation, aux occlusions, etc. Récemment, l'introduction des caméras RGB-D sur le marché a permis l'élaboration de nouvelles approches, grâce aux nouvelles modalités qu'elles proposent (images de profondeur et séquences de squelettes).

Cette thèse a donc pour objectif de proposer un système de reconnaissance d'action qui soit à la fois rapide et précis, en s'appuyant sur ces nouvelles modalités.

0.1.2 Contributions

Dans cette partie, nous présentons les différentes contributions proposées dans cette thèse:

1) L'évaluation des méthodes de l'état de l'art en termes de précision et de temps de calcul.

2) La proposition d'une nouvelle approche de normalisation temporelle appelée *Time Variable Replacement*.

3) L'introduction d'un nouveau descripteur rapide et précis:*Kinematic Spline Curves (KSC)*

4) L'extension des méthodes d'apprentissage à noyaux pour les matrices symétriques semi-définies positives.

5) La proposition d'un nouveau descripteur appelé *Hierarchical Kinematic Covariance (HKC)*

0.1.3 Organisation du manuscrit

Ce manuscrit est organisé comme suit: le chapitre 2 présente une vue globale de l'état de l'art des méthodes de reconnaissance d'actions. Puis, le chapitre 3 propose une étude comparative des méthodes de reconnaissance d'actions basées sur des caméras RGB-D. Le chapitre 4 est consacré à l'introduction d'un nouveau descripteur appelé KSC, tandis que le chapitre 5 décrit deux nouveaux descripteurs: *Kinematic Covariance* et *Hierarchical Kinematic Covariance*. Leur performance respective dans le contexte de la reconnaissance d'actions est également démontrée grâce à diverses expérimentations. Enfin, le chapitre 6 conclut ce travail et présente d'intéressantes perspectives pour nos travaux futurs. Dans cette synthèse, chaque section représente un résumé détaillé des différents chapitres de cette thèse.

0.2 État de l'art

Dans cette partie, nous commençons par présenter le schéma classique d'un système de reconnaissance d'actions. Puis, nous évoquons les méthodes de reconnaissance d'actions basées sur des caméras RGB, ainsi que leurs limitations. Les systèmes d'acquisition 3D les plus utilisés dans ce domaine sont alors présentés. Suite à cela, une vue globale de l'état de l'art des méthodes de reconnaissance d'actions à partir de caméras RGB-D est présentée. Pour finir, les méthodes de reconnaissance d'actions rapides et en ligne à partir caméras RGB-D sont revues.

0.2.1 Vue globale d'un système de reconnaissance d'actions

Généralement, la reconnaissance d'actions peut être vue comme la succession de deux étapes principales, à savoir, *la description et la classification de l'action.*

La figure 2.2 illustre ces deux étapes. Les caractéristiques sont tout d'abord extraites des images. Puis, ces caractéristiques sont représentées dans un même espace S. Les données d'apprentissage sont alors utilisées pour répartir l'espace S en différentes régions i (en fonction du type de l'action C_i). Cette séparation, correspondant à un modèle de classification, est utilisée pour prédire le type d'action présente dans une nouvelle vidéo.

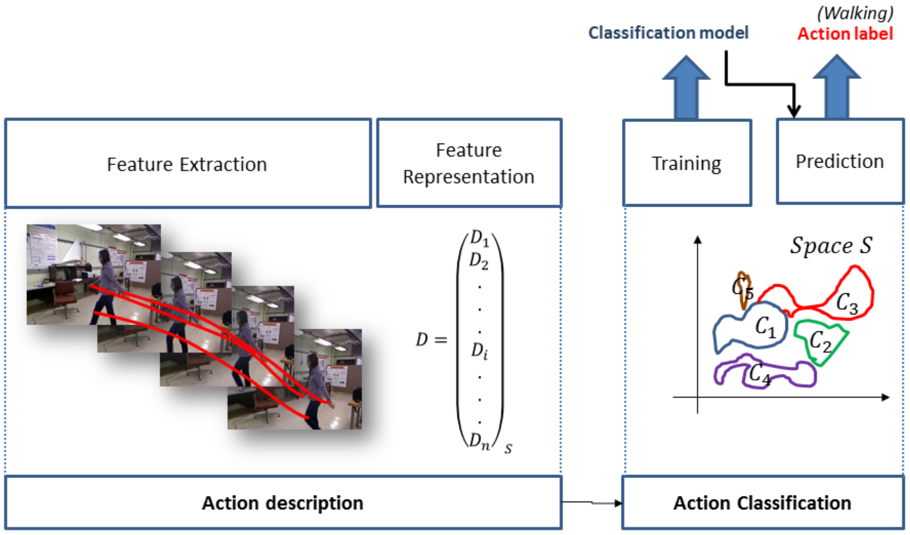


FIGURE 1: Schéma d'un système de reconnaissance d'actions

0.2.2 Les méthodes de reconnaissance d'actions basées sur des caméras RGB et leurs limitations

Les premières méthodes de reconnaissance d'actions se sont essentiellement basées sur des caméras RGB. Il existe dans la littérature plusieurs articles qui revoient l'ensemble de ces méthodes telles que [69, 97]. Bien que celles-ci aient prouvé leurs performances, elles présentent toutefois certaines limitations telles que la sensibilité à la segmentation, à la variation d'orientation, aux changements de luminosité et aux occlusions. La figure 2.3 illustre ces différentes limitations.

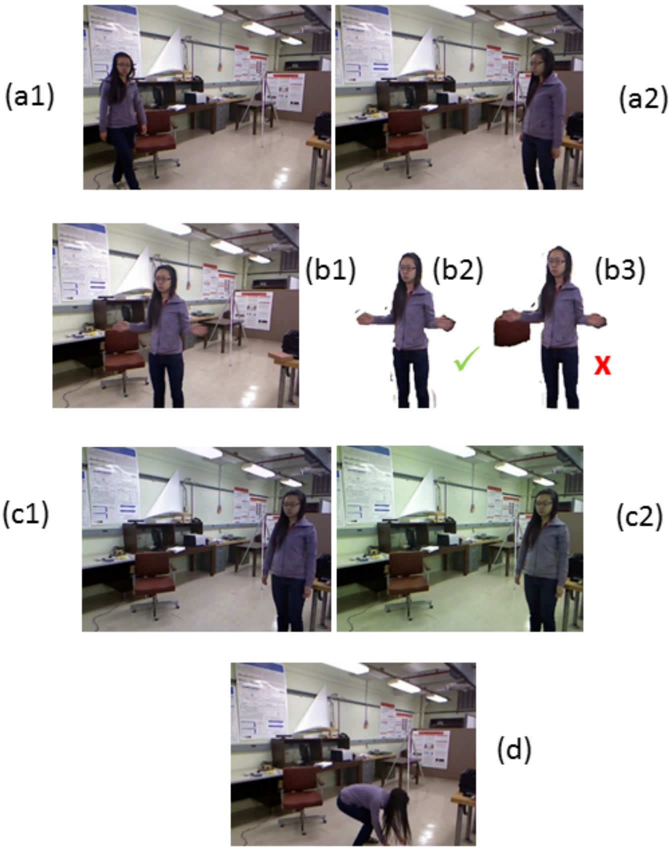


FIGURE 2: Exemples des effets causés par la variation d'orientation (a1,a2), la segmentation (b1,b2,b3), la variation de luminosité (c1,c2) et les occultations (d)

0.2.3 Les systèmes d'acquisition 3D

Afin de pallier ces limitations, des systèmes d'acquisition 3D ont été développés. Dans [3], Aggarwal et Xia présentent les dispositifs 3D les plus utilisés dans le domaine de la reconnaissance d'actions. Nous citons tout d'abord les systèmes stéréoscopiques mettant en jeu deux caméras [86], les systèmes de capture de mouvement, appelés plus communément MoCap[79, 57, 36] et les systèmes RGB-D.

En plus d'images classiques RGB, les caméras RGB-D fournissent en temps réel des images de profondeur et des séquences de squelette [81].

La figure 2.5 illustre ces deux nouvelles modalités.

Les caméras RGB-D (ex: Kinect) ont l'avantage d'être relativement peu coûteuses et ne nécessitent pas le port de capteurs. Pour cette raison, dans cette thèse, nous nous focalisons sur la reconnaissance d'actions basée sur des caméras RGB-D.

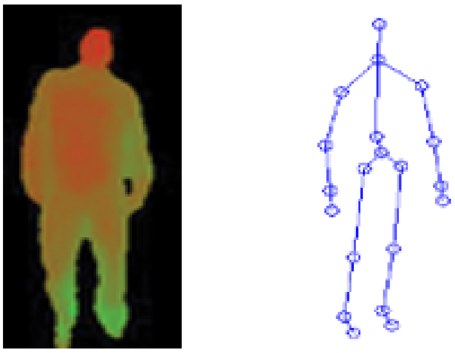


FIGURE 3: Un exemple d'image de profondeur (à gauche) et de squelette (à droite)

0.2.4 Les méthodes de reconnaissance d'actions basées sur des caméras RGB-D

Durant cette dernière décennie, de nouvelles méthodes, se basant sur les nouvelles modalités fournies par les caméras RGB-D, se sont développées. Cela est principalement dû aux nombreux avantages que possèdent ces capteurs. Des revues de l'état de l'art des méthodes de reconnaissance d'actions basées sur des caméras RGB-D ont été présentées dans les articles suivants [3, 108].

Ce dernier papier [108] a catégorisé les méthodes RGB-D en deux groupes distincts: les représentations *hand-crafted* (fixées à la main) et les représentations apprises.

Les méthodes hand-crafted sont plus souvent utilisées [63, 91, 94]. Elles suivent le schéma classique de reconnaissance d'actions présenté précédemment.

Grâce aux avancées récentes en *deep learning*, des représentations apprises ont été proposées. Au lieu de choisir des caractéristiques spécifiques, cette catégorie d'approches sélectionne automatiquement les plus appropriées ([44, 95, 70]).

Comme ces travaux s'intéressent à la description de l'action humaine et plus spécifiquement aux méthodes *hand-crafted*, la catégorisation des méthodes est faite en fonction de la modalité utilisée et par conséquent en fonction de la nature du descripteur. De ce fait, nous distinguons 3 différents types d'approches: les approches basées sur la profondeur, les approches basées sur le squelette et les approches hybrides. La figure 2.6 présente une vue globale de l'état de l'art des méthodes de reconnaissance d'actions basées sur des caméras RGB-D.

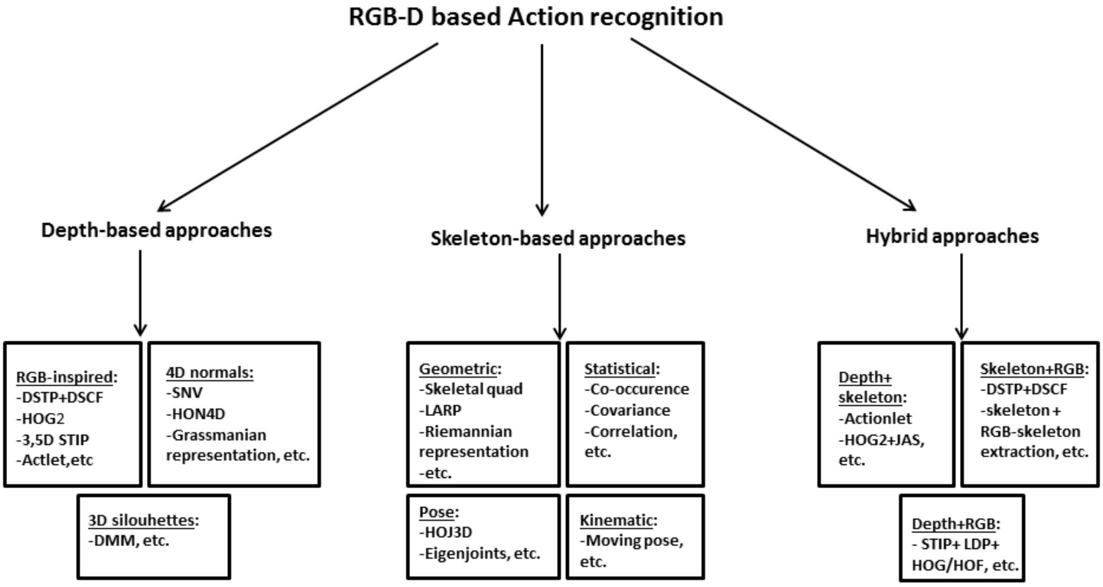


FIGURE 4: Vue globale de l'état de l'art des méthodes de reconnaissance d'actions basées sur des caméras RGB-D réparties en 3 groupes: des descripteurs basés profondeur, basés squelette et hybride

0.2.5 Reconnaissance d'actions en ligne et rapide basée sur des caméras RGB-D

Afin de satisfaire les contraintes temps réels requises par un grand nombre d'applications, on note dans la littérature plusieurs tentatives de systèmes de systèmes de reconnaissance d'actions temps-réel. D'après [58], le challenge principal d'une application temps réel est de réduire au maximum la latence. La latence est définie comme étant la somme de la latence observationnelle et de la latence calculatoire. La latence observationnelle représente le temps d'observation nécéssaire pour prédire correctement l'action en cours, tandis que la latence calculatoire correspond au temps d'exécution nécessaire à la réalisation des calculs.

La majorité des papiers traitant l'aspect temps réel se sont focalisés sur la réduction de la latence observationnelle et ont appelé cette tâche reconnaissance d'actions en ligne. Ainsi, nous proposons de définir la tâche qui bute à réduire la latence obsevationnelle par la reconnaissance d'actions rapide.

La figure 2.18 reprend les méthodes de reconnaissance d'actions en ligne les plus connues.

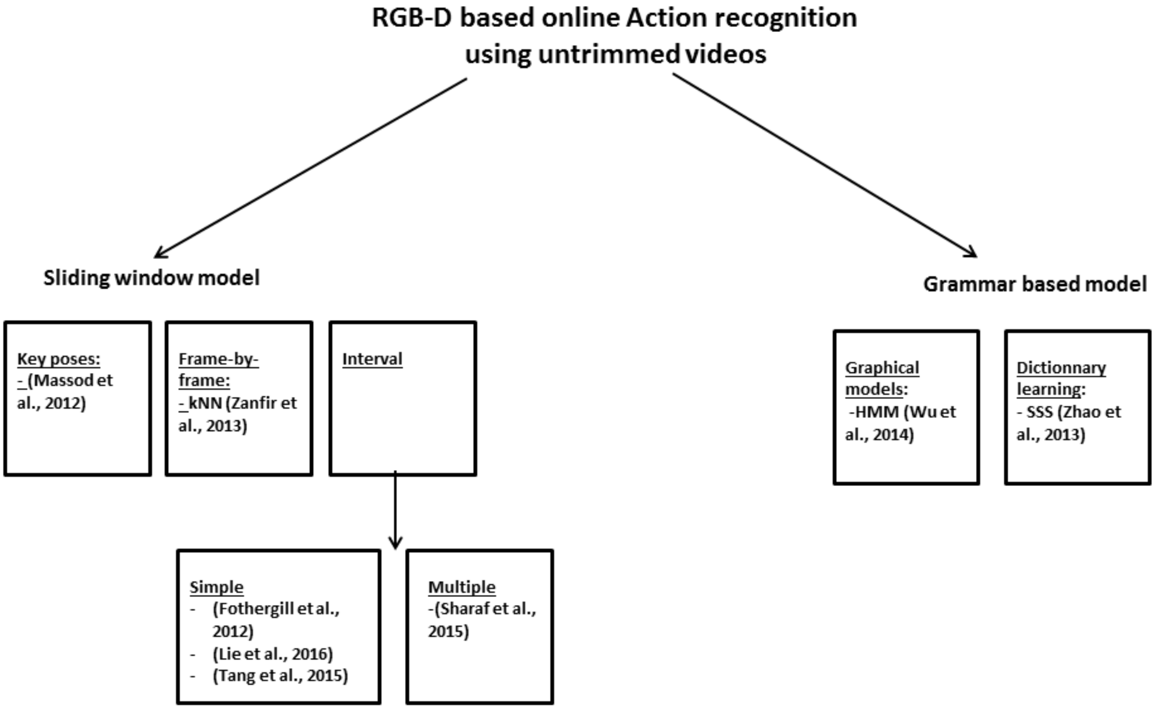


FIGURE 5: Vue globale de l'état de l'art des méthodes de reconnaissance d'actions en ligne basées sur des caméras RGB-D

0.3 Évaluation des descripteurs RGB-D

Afin de pouvoir orienter nos recherches et de choisir la modalité adéquate, nous proposons une étude comparative ayant pour objectif d'évaluer les descripteurs RGB-D. Dans cette étude, nous prenons en compte uniquement les descripteurs simples, c'est-à-dire les descripteurs qui se basent sur une seule modalité (descripteurs basés sur la profondeur et descripteurs basés sur le squelette).

Comme exposé dans [66], les protocoles d'expérimentation diffèrent d'un article à un autre. Pour cette raison, nous proposons de récupérer les codes des méthodes récentes et de les tester sur des bases de données similaires, en respectant les mêmes conditions expérimentales. Afin de ne pas biaiser nos expérimentations, nous choisissons des méthodes usant d'un même classifieur (SVM).

La figure 3.1 schématise le protocole d'évaluation proposé.

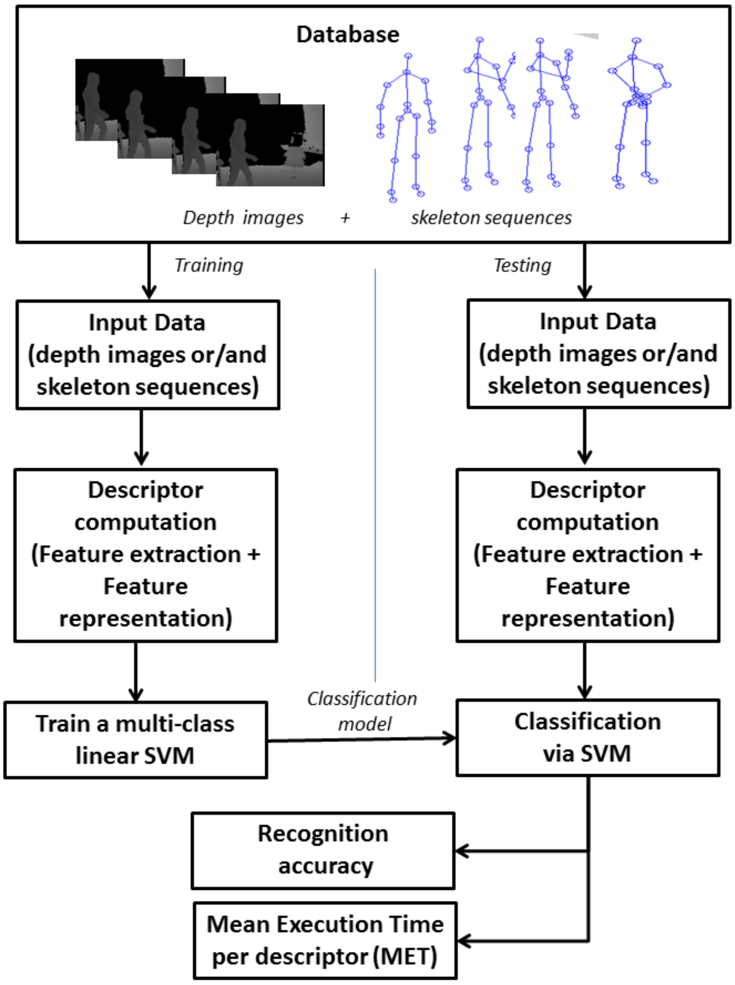

Database
Depth images + skeleton sequences
Training
Testing
Input Data (depth images or/and skeleton sequences)
Descriptor computation (Feature extraction + Feature representation)
Train a multi-class linear SVM
Classification model
Input Data (depth images or/and skeleton sequences)
Descriptor computation (Feature extraction + Feature representation)
Classification via SVM
Recognition accuracy
Mean Execution Time per descriptor (MET)

FIGURE 6: Le protocole d'évaluation proposé

0.3.1 Descripteurs testés

Comme évoqué précédemment, nous avons téléchargé les codes permettant le calcul de certains descripteurs basés sur des caméras RGB-D. Au total, nous avons récupéré 3 descripteurs de profondeur (HOG2 [63], HON4D [65] et SNV [102]) et 5 descripteurs de squelette [91] *Joint Position* (JP), *Relative Joint Position* (RJP), *Joint Angles* (JA), *individual Body Part Locations* (BPL), et *Lie Algebra Relative Positions* (LARP).

0.3.2 Critères d'évaluation

Le taux de reconnaissance est le critère le plus utilisé pour évaluer une méthode de reconnaissance. Vu que nous sommes également intéressés par l'aspect temps-réel, nous définissons

Descripteur	AS1(%)	AS2(%)	AS3(%)	Taux de reconnaissance(%)	MET (s)
HOG2 [63]	90.47	84.82	**98.20**	91.16	**6.44**
HON4D [65]	94.28	91.71	**98.20**	94.47	27.33
SNV [102]	**95.25**	**94.69**	96.43	**95.46**	146.57
JP [91]	82.86	68.75	83.73	78.44	**0.58**
RJP [91]	81.90	71.43	88.29	80.53	2.15
Q [91]	66.67	59.82	71.48	67.99	1.33
LARP [91]	83.81	84.82	92.73	**87.14**	17.61

TABLE 1: Taux de reconnaissance et temps d'exécution moyen par descripteur (MET) sur la base de données MSRAction3D: AS1, AS2 and AS3 représentent les 3 sous-groupes proposés dans l'expérimentation de [53]

Descripteur	Taux de reconnaissance(%)	MET (s)
HOG2 [63]	74.15	**5.03**
HON4D [65]	**90.92**	25.39
SNV [102]	79.80	1365.33
JP [91]	**100**	**0.43**
RJP [91]	97.98	1.91
Q [91]	88.89	1.40
LARP [91]	97.08	42.00

TABLE 2: Taux de reconnaissance et temps d'exécution moyen par descripteur (MET) sur la base de données UTKinect

un second critère que nous appelons *Mean Execution Time per descriptor (MET)*, qui correspond au temps d'exécution moyen par descripteur.

0.3.3 Bases de données RGB-D pour la reconnaissance d'actions

Comme nous nous intéressons essentiellement à la reconnaissance d'actions (et non d'activités), nous choisissons 3 bases de données contenant des actions humaines à savoir les bases de données MSRAction3D [53], UTKinect [99] et Multiview3D [41]. Les paramètres d'expérimentation choisis pour chaque bases de données sont respectivement similaires à ceux choisis dans [53], [91] et [41].

0.3.4 Résultats et discussion

Les résultats obtenus sur les bases de données MSRAction3D, UTKinect et Multiview3D sont respectivement affichés dans le tableau 3.2, le tableau 3.3 et le tableau 3.4.

Afin de visualiser le taux de reconnaissance en même temps que le temps d'exécution moyen par descripteur, nous proposons la figure 3.5 qui illustre les valeurs obtenues sur la base de données MSRAction3D. Chaque boule correspond à une méthode. Le centre de la boule représente le taux de reconnaissance de la méthode associée et la surface son temps d'exécution moyen par descripteur.

Descriptor	SV(%)	DV(%)	MET (s)
HOG2 [63]	87.8	74.2	9.06
HON4D [65]	89.3	76.6	17.51
SNV [102]	94.27	76.65	271.72
JP [91]	96.00	88.10	1.22
RJP [91]	97.70	92.7	4.58
Q [91]	91.30	72.10	2.52
LARP [91]	96.00	88.1	10.51

TABLE 3: Taux de reconnaissance et temps d'exécution moyen par descripteur (MET) sur la base de données Multiview3D

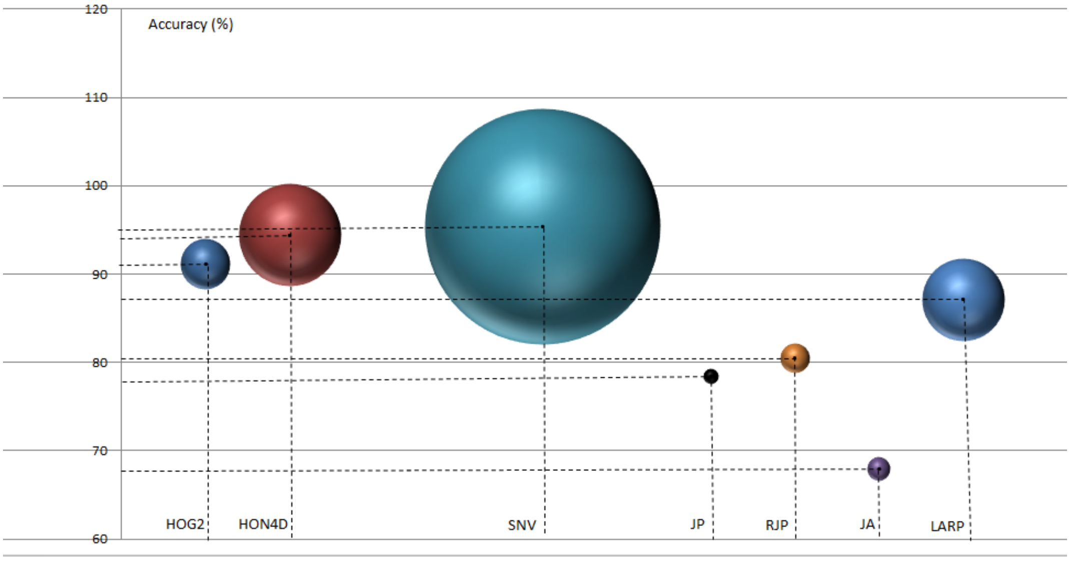


FIGURE 7: Illustration du taux de reconnaissance et du temps d'exécution moyen par descripteur (MET) sur MSRAction3D

Pour résumer, nous avons pu relever grâce à ces expérimentations des points communs et des points de différence entre les descripteurs de profondeur et de squelette. Le tableau 4 indique les différentes propriétés observées.

Cette analyse nous a donc orienté dans le choix de la modalité. Vu que nous avons besoin d'un descripteur qui soit à la fois précis et rapide, nous avons opté pour la modalité squelette.

0.4 Kinematic Spline Curves: un nouveau descripteur rapide et précis pour la reconnaissance d'actions basée sur des caméras RGB-D

Dans cette partie correspondant au chapitre 4 du manuscrit, nous proposons un nouveau descripteur de mouvement humain basé sur la modalité squelette, rapide à calculer et précis. Ce descripteur a été appelé *Kinematic Spline Curves* (KSC) et est calculé grâce à l'interpolation

descriptor	depth-based descriptors	skeleton-based descriptors
Accuracy (A)	nothing specific	
Descriptor Dimension	generally higher	generally lower
Computational Latency (CL)	generally higher	generally lower
Trade-off between CL and A	generally less good	generally better
Robustness to dataset changing	nothing specific	
Robustness to view-point variation	generally less robust	generally more robust

TABLE 4: Comparaison entre les descripteurs de squelette et de profondeur

par splines cubiques des caractéristiques cinématiques du squelette (la position, la vitesse et l'accélération), suivi d'un échantillonage uniforme. Cette procédure nous permet de calculer des descripteurs de même taille, quelque soit la taille de la vidéo (qui est variable d'une instance à une autre). Les différents algorithmes qui ont permis de calculer ce descripteur ont été choisis de sorte à réduire le temps de calcul.

D'un autre côté, pour améliorer le taux de reconnaissance du descripteur, deux facteurs doivent être pris en compte: la variabilité anthropométrique et la variation d'exécution d'une action (qui correspond à la variation dans la manière d'effectuer l'action). La variabilité anthropométrique est due à la variation des proportions du corps humain. La variation d'exécution, quant à elle, est principalement causée par une différente distribution de la vitesse du mouvement d'une instance à une autre.

Pour cette raison, afin de réduire les effets négatifs causés par la variabilité anthropométrique et la variation d'exécution, deux algorithmes sont proposés: l'extension de la normalisation du squelette proposé par Zanfir et al. [105] et une nouvelle méthode de normalisation temporelle que nous appelons *Time Variable Replacement* (TVR).

Ainsi, les contributions majeures de ce chapitre sont la proposition du nouveau descripteur KSC et l'algorithme de normalisation temporelle TVR.

0.4.1 Calcul du descripteur KSC

Dans cette partie, nous décrivons les différents procédés qui nous ont permis de calculer le descripteur KSC.

La figure 4.2 illustre ces différentes procédures.

Dans cette étude, une action est représentée par une séquence de squelette. A chaque instant, le squelette est représenté par une pose $\mathbf{P}(t)$, qui est composée de *n* articulations. La position de chaque articulation j est définie par $\mathbf{p}_\mathrm{j}(t) = [x_j(t), y_j(t), z_j(t)]$ avec $j \in [\![1, n]\!]$.

$$\mathbf{P}(t) = [\mathbf{p_1}(t), ..., \mathbf{p_j}(t), ..., \mathbf{p_n}(t)] \tag{1}$$

Comme dans la majorité des études biomécaniques, la position initiale de l'articulation de la hanche est considérée comme étant l'origine du repère utilisé. La figure 4.3 indique la position de l'articulation de la hanche.

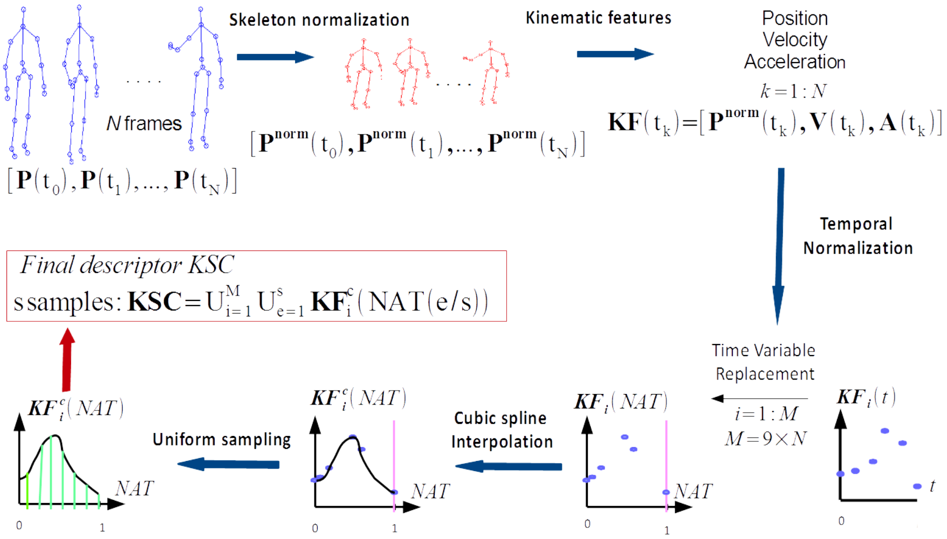

Skeleton normalization
Kinematic features
Position
Velocity
Acceleration
$k=1:N$
N frames
$[\mathbf{P}(t_0), \mathbf{P}(t_1), ..., \mathbf{P}(t_N)]$
$[\mathbf{P}^{norm}(t_0), \mathbf{P}^{norm}(t_1), ..., \mathbf{P}^{norm}(t_N)]$
$\mathbf{KF}(t_k) = [\mathbf{P}^{norm}(t_k), \mathbf{V}(t_k), \mathbf{A}(t_k)]$
Temporal Normalization
Final descriptor KSC
s samples: $\mathbf{KSC} = \mathrm{U}_{i=1}^{M} \mathrm{U}_{e=1}^{s} \mathbf{KF}_i^c(\mathrm{NAT}(e/s))$
Time Variable Replacement
$i=1:M$
$M=9\times N$
$KF_i^c(NAT)$
Uniform sampling
$KF_i^c(NAT)$
Cubic spline Interpolation
$KF_i(NAT)$
$KF_i(t)$
NAT
t
0
1

FIGURE 8: Vue globale du desripteur KSC

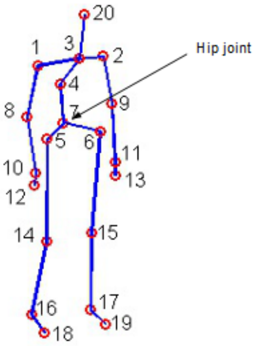


FIGURE 9: Illustration de l'articulation de la hanche d'un squelette

$$\mathbf{P}(t) = [\mathbf{p}_1(t) - \mathbf{p}_{\text{hip}}, ..., \mathbf{p}_{\text{j}}(t) - \mathbf{p}_{\text{hip}}, ..., \mathbf{p}_{\text{n}}(t) - \mathbf{p}_{\text{hip}}] \tag{2}$$

Tout d'abord, un algorithme de normalisation de squelette est appliqué aux données d'entrées.

En partant de l'articulation de la hanche, les différents membres sont normalisés de manière incrémentale (au sens Euclidien). L'algorithme 1 résume l'approche de normalisation proposée.

Puis, les caractéristiques cinématiques KF sont calculées en utilisant la position du squelette normalisée (avec $\mathbf{V}(t) = \frac{d\mathbf{P}^{\mathbf{norm}}(t)}{dt}$ la vitesse et $\mathbf{A}(t) = \frac{d^2\mathbf{P}^{\mathbf{norm}}(t)}{dt}$ l'accélération) (4.7).

$$\mathbf{KF}(t) = [\mathbf{P}^{\text{norm}}(t), \mathbf{V}(t), \mathbf{A}(t))] \tag{3}$$

```
Algorithm 1: Normalisation du squelette à un instant $t$
    Entrées: $(\mathbf{p_{a_i}}(t), \mathbf{p_{b_i}}(t))_{1\leq i \leq C}$ représentent les extrémités des segments du squelette
             ordonnées et $C$ représente le nombre de connexions avec $a_i$ la racine et $b_i$
             l'autre extrémité du segment $i$
    Sorties : $(\mathbf{p^{norm}_{a_i}(t)}, \mathbf{p^{norm}_{b_i}(t)})_{1\leq i \leq C}$ avec $a_i, b_i \in [\![1, n]\!]$
    $\mathbf{p^{norm}_{a_1}}(t) := \mathbf{p_{a_1}}(t)(\mathbf{p_{a_1}}(t) = \mathbf{p}_{hip}(t)$ représente la position de l'articulation de la
      hanche)
    for $i \leftarrow 1$ to $C$ (C:Nombre de segments) do
        $\mathbf{S_i} := \mathbf{p_{a_i}}(t) - \mathbf{p_{b_i}}(t)$
        $\mathbf{s'_i} : = \frac{\mathbf{S}_i}{\|(\mathbf{p_{a_i}}(t) - \mathbf{p_{b_i}}(t))\|_2}$
        $\mathbf{p^{norm}_{b_i}}(t) := \mathbf{s'_i} + \mathbf{p^{norm}_{a_i}}(t)$
    end
```

Pour pallier l'effet engendré par la variation d'exécution, un nouvel algorithme TVR a été proposé. Cela consiste en un changement de variable (la variable temps est remplacé par une autre variable que l'on nomme *Normalized Action Time (NAT))* (4.12).

$$\mathbf{KF}(NAT) = [\mathbf{P}^{\text{norm}}(NAT), \mathbf{V}(NAT), \mathbf{A}(NAT)] \tag{4}$$

Afin d'effectuer ce changement de variable, une fonction appelée *Time Variable Replacement Function (TVRF)* est utilisée. Celle-ci doit respecter les conditions suivantes:

1) Elle doit avoir un sens physique et être invariante à la variation d'exécution.

2) Elle doit réaliser une bijection avec la variable temps.

3) Elle doit varier dans un même intervalle, quelle que soit la taille de la séquence de squelette (figure 4.5).

Soit $TVRF_{\mathscr{I}}$ (avec $\mathscr{I}$ l'index de l'action) une fonction croissante et bijective, où NAT varie:

$$\forall \mathscr{I}, TVRF_{\mathscr{I}} : \left| \begin{array}{rcl} [0, N_{\mathscr{I}}] & \longrightarrow & [a, b] \\ t & \longmapsto & TVRF_{\mathscr{I}}(t) = NAT \end{array} \right.$$

a et b représentent des valeurs constantes telles que $[a, b]$ est l'intervalle où varie NAT. $N_{\mathscr{I}}$ est la taille de la séquence $\mathscr{I}$. Deux fonctions TVRF ont été présentées dans ce manuscrit: *the Normalized Accumulated kinetic Energy (NAE) of the skeleton* et *the Normalized Pose Motion Signal Energy (NPMSE).*

Suite à cela les caractéristiques KF sont interpolées par splines cubiques. Enfin, pour obtenir le descripteur final KSC, les courbes continues obtenues KF^c sont échantillonnées uniformément.

L'algorithme 2 résume le calcul du descripteur KSC.

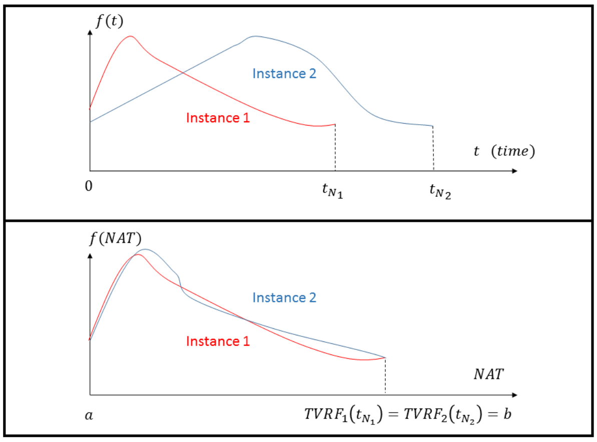


FIGURE 10: Exemple de l'effet de la normalisation temporelle

```
Algorithm 2: Calcul du descripteur KSC à partir d'une instance $\mathscr{I}$
  Entrées: Séquence de squelette $(\mathbf{P_j}(t_k))_{1\leq j\leq n, 1\leq k\leq N}$
  Sorties : KSC
  Normaliser le squelette $(\mathbf{P_j^{norm}}(t_k))_{1\leq j\leq n, 1\leq k\leq N}$
  Calculer les caractéristiques cinématiques $\mathbf{KF_i}(t_k))_{1\leq i\leq M}$ (4.12)
  Calculer $(TVRF_{\mathscr{I}}(t_k))_{1\leq k\leq N}$ (4.13)
  for $i \leftarrow 1$ to $M$ do
      Interpolation:
      $\mathbf{KF_i^c}(NAT) = \mathbf{KF_i^c}(TVRF_{\mathscr{I}}(t)) := Spline(\mathbf{KF_i}(TVRF_{\mathscr{I}}(t_k)))_{1\leq k\leq N}$
  end
  Échantillonner uniformément avec $s$ le nombre d'échantillons:
   KSC $:= \cup_{i=1..M} \cup_{e=0..s-1} \mathbf{KF_i^c}(NAT(a + \frac{(b-a)e}{s-1}))$
```

0.5 Expérimentations

Afin de valider notre méthode, une évaluation a été menée sur quatre bases de données, à savoir, MSRAction3D [53], UTKinect [100], Multiview3D Action [41] et MSRC12 [35].

Le tableau 4.2, le tableau 4.4, le tableau 4.5 et le tableau 4.6 rapportent respectivement le taux de reconnaissance, ainsi que le MET obtenus pour les bases de données MSRAction3D, UTKinect, MSRC12 et Multiview3D.

La figure 4.12, quant à elle, permet de visualiser en même temps, le taux de reconnaissance et le MET par descripteur.

Les résultats obtenus prouvent que notre méthode est à la fois rapide et précise comparée aux méthodes de l'état de l'art.

Descripteur	AS1(%)	AS2(%)	AS3(%)	Global(%)	MET (s)
HOG2 [63]	90.47	84.82	**98.20**	91.16	**6.44**
HON4D [65]	94.28	91.71	**98.20**	94.47	27.33
SNV [102]	**95.25**	**94.69**	96.43	**95.46**	146.57
JP [91]	82.86	68.75	83.73	78.44	0.58
RJP [91]	81.90	71.43	88.29	80.53	2.15
Q [91]	66.67	59.82	71.48	67.99	1.33
LARP [91]	83.81	84.82	92.73	87.14	17.61
KSC-NAE (ours)	86.92	72.32	94.59	84.61	**0.092**
KSC-NPMSE-l_1 (ours)	85.05	85.71	96.4	89.05	**0.091**
KSC-NPMSE-l_∞(ours)	**85.71**	86.61	93.69	88.69	**0.091**
KSC-NPMSE-l_2(ours)	83.81	**87.50**	**97.30**	**89.54**	**0.092**

TABLE 5: Taux de reconnaissance et MET par descripteur sur la base de données MSRAction3D

Descripteur	Taux de reconnaissance (%)	MET (s)
HOG2 [63]	74.15	**5.03**
HON4D [65]	**90.92**	25.39
SNV [102]	79.80	1365.33
JP [91]	**100**	0.43
RJP [91]	97.98	1.91
Q [91]	88.89	1.40
LARP [91]	97.08	42.00
Random Forrest [110]	87.90	-
KSC-NAE (ours)	84.00	**0.08**
KSC-NPMSE-l_1 (ours)	94.00	0.09
KSC-NPMSE-l_∞ (ours)	94.00	**0.08**
KSC-NPMSE-l_2 (ours)	**96.00**	0.10

TABLE 6: Taux de reconnaissance et MET par descripteur sur la base de données UTKinect

Descripteur	Taux de reconnaissance (%)	MET (s)
Logistic Regression [58]	91.2	-
Covariance descriptor [45]	91.7	-
KSC-NAE (ours)	84.00	0.137
KSC-NPMSE-l_1 (ours)	93.22	**0.134**
KSC-NPMSE-l_∞ (ours)	94.17	**0.132**
KSC-NPMSE-l_2 (ours)	**94.27**	0.134

TABLE 7: Taux de reconnaissance et MET par descripteur sur la base de données MSRC12

Descripteur	Même orientation (%)	Orientations différentes (%)	MET (s)
Actionlet [94]	87.1	69.7	-
HOG2 [63]	87.8	74.2	9.06
HON4D [65]	89.3	76.6	17.51
SNV [102]	94.27	76.65	271.72
JP [91]	96.00	88.10	1.22
RJP [91]	97.70	92.7	4.58
Q [91]	91.30	72.10	2.52
LARP [91]	96.00	88.1	10.51
KSC-NAE (ours)	82.12	79.43	0.09
KSC-NPMSE-l_1 (ours)	**90.63**	89.67	0.091
KSC-NPMSE-l_{infty} (ours)	89.93	89.06	**0.09**
KSC-NPMSE-l_2 (ours)	90.45	**90.10**	0.092

TABLE 8: Taux de reconnaissance et MET par descripteur sur la base de données Multiview3D

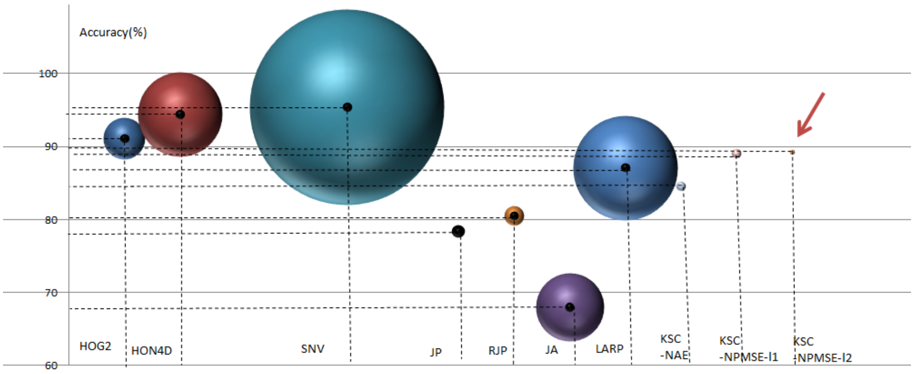


FIGURE 11: Illustration du taux de reconnaissance et de la MET par descripteur sur MSRAction3D

0.6 Utilisation des matrices Symétriques semi-définies positives comme descripteurs de mouvement

Le chapitre 4 a été dédié à la présentation du nouveau descripteur KSC. Ce dernier est précis et rapide à calculer. Cependant, il est difficilement adaptable aux applications en ligne, vu qu'on doit connaître la vidéo entière pour pouvoir appliquer la normalisation temporelle.

Pour cela, nous proposons d'utiliser les descripteurs de covariance qui contrairement au descripteur KSC, ne nécessitent pas la connaissance a priori de la vidéo entière: ils peuvent être calculés sur des sous-parties de la vidéo.

Deux approches sont alors introduites: une approche statique et une approche dynamique.

0.6.1 Approche statique

Ainsi, nous proposons un nouveau descripteur appelé *Kinematic Covariance* (KC) qui est calculé en intégrant les caractéristiques cinématiques KF dans une matrice de covariance.

A chaque instant t_k, $\mathbf{KF}(t_k)$ est calculé par la concaténation de la position des articulation $\mathbf{P^{norm}}(t_k)$, de leur vitesse $\mathbf{V}(t_k)$ et de leur accélération $\mathbf{A}(t_k)$ (equation 5).

$$\mathbf{KF}(t_k) = [\mathbf{p^{norm}}(t_k), \mathbf{V}(t_k), \mathbf{A}(t_k)] \tag{5}$$

Nous rappelons que la dimension du vecteur $\mathbf{KF}(t_k)$ est égale à $d_1 = 9 \times n$, avec n le nombre d'articulations. Nous supposons que N correspond au nombre de frames. Ainsi, le nouveau descripteur KC est calculé comme décrit par l'équation (5.17),

$$\mathbf{KC} = \frac{1}{N} \sum_{k=1}^{N} (\mathbf{KF}(t_k) - \nu)(\mathbf{KF}(t_k) - \nu)^T \tag{6}$$

où $\nu = \sum_{k=1}^{N} \mathbf{KF}(t_k)$ représente le vecteur de caractéristiques cinématiques. Nous pouvons aisément déduire que le descripteur KC représente une matrice carrée de dimension $d_1 \times d_1$.

Les matrices de covariance calculées sont symétriques semi-définies positives. Pour cette raison, des algorithmes à noyaux ont été étendus à l'espace des matrices symétriques semi-définies positives noté Sym_d^+. Nous utilisons plus particulièrement le noyau *Radial Basis Functions* (RBF). Ce dernier est exprimé en fonction de la distance Euclidienne, lorsque les caractéristiques sont placées dans l'espace $\mathbb{R}^n$. Ainsi, nous proposons une nouvelle distance pour l'espace Sym_d^+ que nous appelons *Modified Log-Euclidean.*

Nous définissons cette distance d_{MLE} entre deux matrices symétriques semi-définies positives A et B comme suit:

$$\begin{aligned} d_{MLE}(\mathbf{A}, \mathbf{B}) &= \|Log(\psi(\mathbf{A})) - Log(\psi(\mathbf{B}))\|_F \\ &= \|Log(\mathbf{A} + \epsilon \mathbf{I}_d) - Log(\mathbf{B} + \epsilon \mathbf{I}_d)\|_F \\ &= \|Log(\mathbf{A}_1) - Log(\mathbf{B}_1)\|_F = d_{LE}(\mathbf{A}_1, \mathbf{B}_1) \end{aligned} \tag{7}$$

Les démonstrations liées à cette distance sont présentées dans la dissertation.

Ainsi, nous avons pu déduire le corollaire suivant, qui nous a permis d'utiliser le noyau K_G^{SPsD} conjointement avec le classifieur SVM.

Corollaire 0.6.1. *Soit* $K_G^{SPsD} : Sym_d^+ \times Sym_d^+ \to \mathbb{R}$ *un noyau tel que* $K_G^{SPsD}(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{d_{MLE}^2(\mathbf{x}_i, \mathbf{x}_j)}{2\sigma^2})$ *et* $d_{MLE}(\mathbf{x}_i, \mathbf{x}_j) = \|Log(\mathbf{x}_i + \epsilon \mathbf{I}_d) - Log(\mathbf{x}_j + \epsilon \mathbf{I}_d)\|_F$ *pour* $\mathbf{x}_i, \mathbf{x}_j \in Sym_d^+$*. Alors,* K_G^{SPsD} *et un noyau définie positive* $\forall \sigma \in \mathbb{R}$ *et* $\forall \epsilon > 0$.

La figure 5.2 et l'algorithme 3 présentent une vue globale de la méthode statique proposée.

0.6.2 Approche dynamique

Le descripteur KC et plus généralement les descripteurs de covariance sont limités par le fait qu'ils n'incluent pas l'information temporelle. En effet, ils n'informent pas sur l'évolution

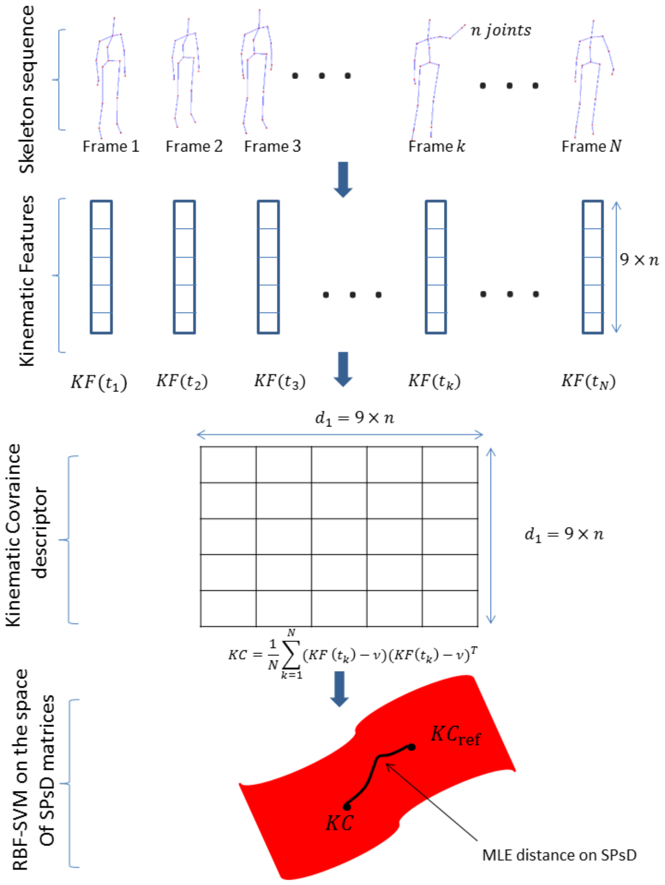


FIGURE 12: L'approche statique proposée qui combine le descripteur KC et le classifieur SVM basé sur le noyau RBF-MLE

Algorithm 3: Reconnaissance d'action à partir d'instance $\mathscr{I}$ en sebasant sur l'approche statique proposée

```
Entrées: Séquence de squelette $(\mathbf{P_j}(t_k))_{1\leq j\leq n, 1\leq k\leq N}$
Sorties : $label$
Normaliser le squelette $(\mathbf{P_j^{norm}}(t_k))_{1\leq j\leq n, 1\leq k\leq N}$
Calculer les caractéristiques cinématiques $\mathbf{KF_i}(t_k))_{1\leq i\leq M}$
Calculer $(\mathbf{KC}_{\mathscr{I}})$ (5.17)
$label$ =Reconnaissance d'action en utilisant la version étendue de SVM ( avec noyau
    RBF) en se basant sur la distance MLE
```

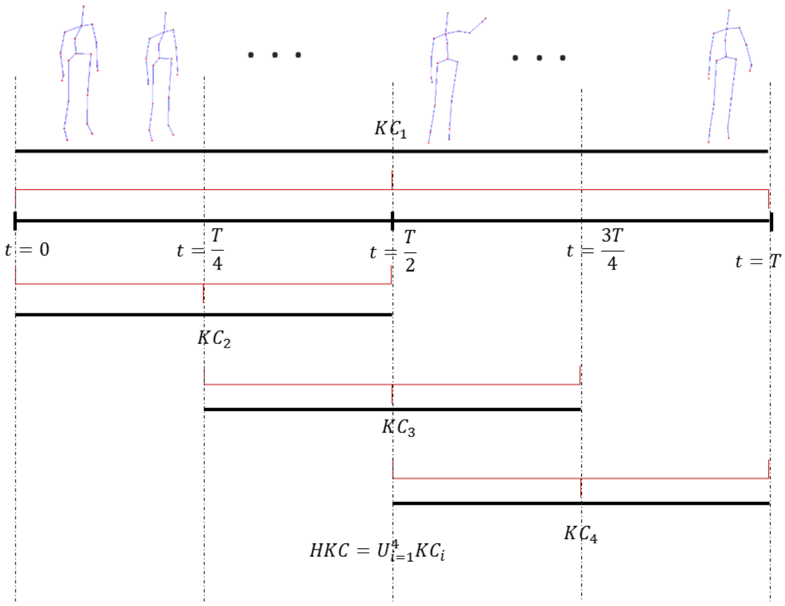


FIGURE 13: Calcul du descripteur HKC

dynamique des actions. Pour cela, nous proposons un second descripteur nommé *Hierarchical Kinematic Covariance* (HKC).

Comme dans [45], 4 dscripteurs KC $(\mathbf{KC}_i)_{1 \leqslant i \leqslant 4}$ sont extraits d'une séquence de squelette en utilisant respectivement 3 sous-intervalles, ainsi que l'intervalle entier. L'ensemble de ces descripteurs KC est appelé HKC, comme le décrit l'équation (5.24).

$$\mathbf{HKC} = \cup_{i=1}^{4} \mathbf{KC}_i \tag{8}$$

La figure 5.3 illustre la manière avec laquelle sont extraits les différents sous-intervalles.

Afin de classifier les actions et fusionner l'information provenant des différents descripteurs KC, une stratégie *Multiple Kernel Learning* est choisie. Le noyau utilisé pour la classification SVM est donc le suivant:

$$K_{MKL}^{SPsD} = \sum_{i=1}^{4} \mu_i K_G^{SPsD}(\mathbf{KC}_i) \tag{9}$$

La figure 5.4, ainsi que l'algorithme 4 décrivent l'approche dynamique proposée.

0.6.3 Expérimentations

Afin de prouver la performance des approches présentées dans cette partie, nous avons présenté respectivement, dans le tableau 5.1, le tableau 5.2 et le tableau 5.3, les résultats obtenus (taux de reconnaissance et MET) sur les bases de données MSRAction3D, UTKinect et Multiview3D. De plus, la figure 5.5 a permis d'illustrer simultanément sur un même graphe le taux de reconnaissance ainsi que le MET.

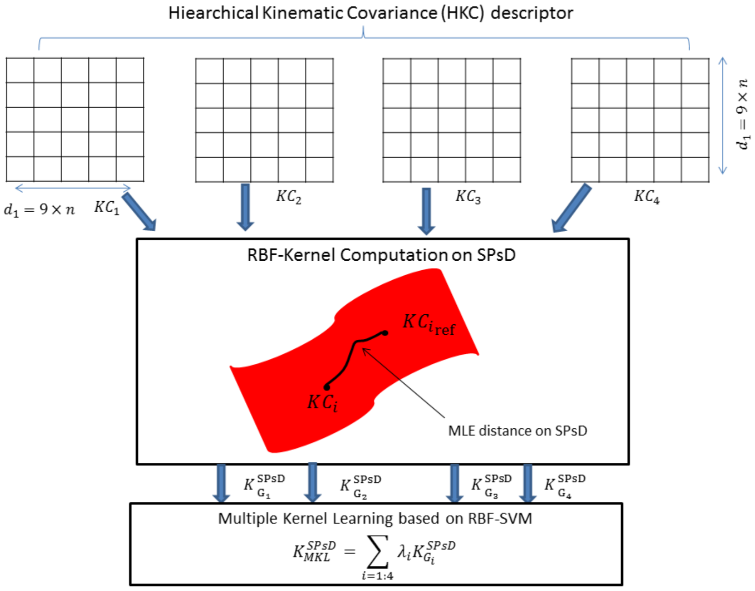


FIGURE 14: L'approche dynamique proposée qui combine le descripteur HKC et le classifieur SVM-MKL basé sur le noyau RBF-MLE

Algorithm 4: Reconnaissance d'actions à partir d'une instance $\mathscr{I}$ en se basant sur l'approche dynamique proposée

```
   Entrées: Séquence de squelette $(\mathbf{P_j}(t_k))_{1\leq j\leq n, 1\leq k\leq N}$
   Sorties : $label$
1  Normaliser le squelette $(\mathbf{P_j^{norm}}(t_k))_{1\leq j\leq n, 1\leq k\leq N}$
2  Calculer les caractéristiques cinématiques $\mathbf{KF_i}(t_k))_{1\leq i\leq M}$
3  Calculer $(\mathbf{HKC}_{\mathscr{I}})$ (5.25)
4  $label$ =Reconnaissance d'actions en utilisant la version étendue de MKL (noyau
    RBF) en se basant sur la distance MLEL
```

Descripteur	AS1(%)	AS2(%)	AS3(%)	Global(%)	MET
HOG2 [63]	90.47	84.82	**98.20**	91.16	**6.44**
HON4D [65]	94.28	91.71	**98.20**	94.47	27.33
SNV [102]	**95.25**	**94.69**	96.43	**95.46**	146.57
JP [91]	82.86	68.75	83.73	78.44	0.58
RJP [91]	81.90	71.43	88.29	80.53	2.15
Q [91]	66.67	59.82	71.48	67.99	1.33
LARP [91]	83.81	84.82	92.73	87.14	17.61
KSC (ours)	83.81	87.5	97.3	89.54	0.092
KC+SVM-MLE (ours)	88.57	83.04	92.79	88.133	**0.043**
HKC+MKL-MLE (ours)	**91.42**	**92.85**	**92.79**	**92.35**	0.044

TABLE 9: Taux de reconnaissance et MET par descripteur sur la base de données MSRAction3D

Descripteur	Taux de reconnaissance (%)	MET (s)
HOG2 [63]	74.15	5.025
SNV [102]	79.80	1365.33
HON4D [65]	90.92	25.33
Random Forest* [110]	87.90	-
LARP [91]	**97.08**	42.00
KSC (ours)	96.00	0.082
KC+SVM-MLE (ours)	90.91	**0.032**
HKC+MKL-MLE (ours)	94.95	**0.033**

TABLE 10: Taux de reconnaissance et MET par descripteur sur la base de données UTKinect

Descripteur	Même orientation (%)	Orientations différentes (%)	MET (s)
HOG2 [63]	87.8	74.2	9.06
HON4D [65]	89.3	76.6	17.51
SNV [102]	94.27	76.65	271.3
Actionlet* [94]	87.1	69.7	0.139
LARP [91]	96.00	88.1	10.51
KSC (ours)	90.45	90.10	0.099
KC+SVM-MLE (ours)	80.72	75.17	**0.033**
HKC+MKL-MLE (ours)	**96.17**	**93.40**	**0.035**

TABLE 11: Taux de reconnaissance et MET par descripteur sur la base de données Multiview3D

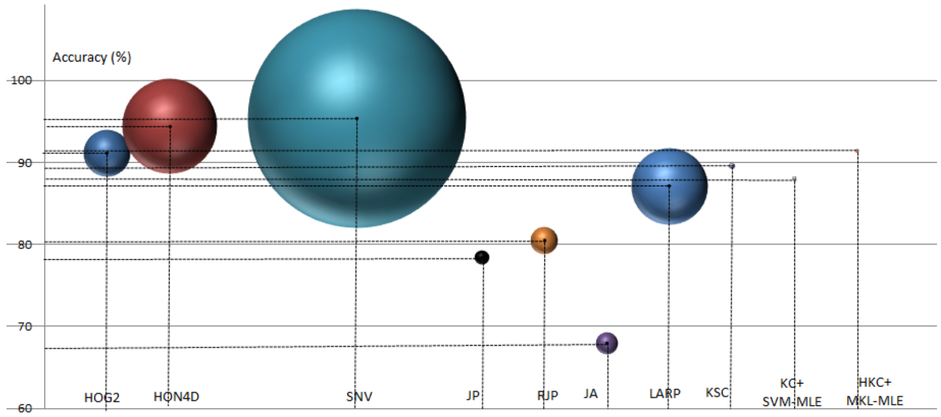


FIGURE 15: Illustration du taux de reconnaissance et du MET par descripteur sur la base de données MSRAction3D

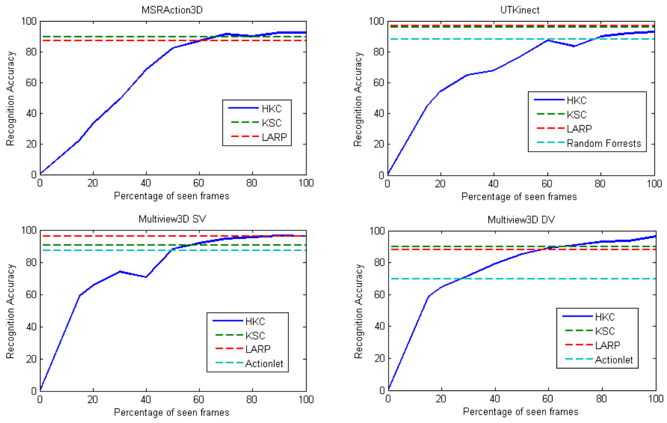


FIGURE 16: Taux de reconnaissance en fonction du pourcentage de frames observées

Ainsi, les résultats obtenus prouvent la performance de notre méthode par rapport aux méthodes de l'état de l'art.

Pour évaluer notre méthode en termes de latence observationnelle, nous proposons également de calculer le taux de reconnaissance en fonction du pourcentage de frames observées. Ces résultats sont illustrés dans la figure 5.10.

0.7 Conclusion

Ces travaux se sont intéressés principalement à la reconnaissance rapide d'actions à partir de caméras RGB-D. Pour cela, nous avons tout d'abord évalué certains descripteurs de l'état de l'art et prouvé l'adéquation des descripteurs de squelette avec la reconnaissance rapide. Suite à cela, nous avons proposé un nouveau descripteur de mouvement humain (KSC), ainsi qu'une nouvelle méthode de normalisation temporelle (TVR). La performance de ces algorithmes a été par la suite prouvée par diverses expérimentations. Bien que ce dernier soit assez performant, il reste cependant difficilement adaptable à un contexte en ligne. En effet, le calcul de ce descripteur nécessite la connaissance a priori de toute la vidéo.

C'est alors que nous avons proposé un second descripteur (HKC) adaptable aux scénarios en ligne. L'utilisation de ce descripteur a toutefois nécessité l'extension des méthodes de classification à noyaux (telles que SVM et MKL) aux matrices symétriques semi-définies positives. Par conséquent, une nouvelle distance adaptée à ces matrices a également été proposée: la distance MLE. Les expérimentations ont également prouvé la validité de cette méthode et a ouvert ainsi plusieurs perspectives pour nos travaux futurs telles que: la reconnaissance d'actions continue, l'apprentissage d'actions incrémental, la reconnaissance d'activité rapide et la reconnaissance d'actions à partir d'un système composé de multiples caméras RGB-D.

Publications

Journal publications

Ghorbel, E., Boonaert, J., Boutteau, R., Lecoeuche, S. Savatier, X., An extension of kernel learning methods using a Modified Log-Euclidean distance for fast skeleton-based human action recognition. Under review, Submitted to *Journal of Computer Vision and Image Understanding* on July 2017.

Ghorbel, E., Boutteau, R., Boonaert, J., Savatier, X., Lecoeuche, S., Kinematic Spline Curves: A temporal invariant descriptor for fast action recognition, Under review, Submitted to *Journal of Image and Vision Computing* on February 2017.

International Conference publications

Ghorbel, E., Boutteau, R., Boonaert, J., Savatier, X., Lecoeuche, S. A fast and accurate motion descriptor for human action recognition applications. In *International Conference on Pattern Recognition (ICPR)*, 2016, Cancun, Mexico.

Hammouche, M., Ghorbel, E., Fleury, A., Ambellouis, S., Toward a real time view-invariant 3D action recognition. In *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, 2016, Roma, Italy.

Ghorbel, E., Boutteau, R., Boonaert, J., Savatier, X., Lecoeuche, S., 3D real-time human action recognition using a spline interpolation approach. In *IEEE International Conference on Image Processing Theory, Tools and Applications (IPTA)*, 2015, Orléans, France.

National Conference publications

Ghorbel, E., Boutteau, R., Boonaert, J., Savatier, X., Lecoeuche, S., Vers une reconnaissance en ligne d'actions à partir de caméras RGB-D. In *Congrés national Reconnaissance de Formes et Intelligence Artificielle (RFIA)*, 2016, Clermont-Ferrand, France.

Abbreviations

BPL: Individual Body Part Locations
DMM: Depth Motion Maps
DSTIP: Depth Spatio-Temporal Invariant Points
DTW: Dynamic Time Warping
DV: Different View
ES: Ergodic State
FTP: Fourrier Temporal Pyramid
GMM: Gaussian Mixture Models
HKC: Hierarchical Kinematic Covariance
HMM: Hidden Markov Models
HMU: Human Motion Understanding
HOG: Histogram of Oriented Gradients
HOG2: square Histogram of Oriented Gradients
HOG3D: 3D Histogram of Oriented Gradients
HOJ3D: 3D Histogram of Joints
HON4D: Histogram of Oriented 4D Normals
JA: Joint Angles
JAS: Joint Angle Similaities
JP: Joint Position
KC: Kinematic Covariance
KF: Kinematic Features
k**NN**: k Nearest Neighbors
KSC: Kinematic Spline Curves
LARP: Lie Algebra Relative Pairs
LOP: Local Occupancy Patterns
LSTM: Long Short-Term Memory
MET: Mean Execution Time per descriptor
MKL: Multiple Kernel Learning
MLE: Modified Log-Euclidean distance
MoCap: Motion Capture
NAE: Normalized Accumulated kinetic Energy
NPMSE: Normalized Pose Motion Signal Energy
RBF: Radial Basis Functions

RGB: Red Green Blue
RGB-D: Red Green Blue- Depth
RJP: Relative Joint Position
RNN: Reccurent Neural Networks
SL: Stuctured-Light
SPD: Symmetric Positive Definite
SPsD: Symmetric Positive semi-Definite
SNV: Support Normal Vector
SRVF: Square-Root Velocity Function
STIP: Spatio-Temporal Invariant Points
SV: Same View
SVM: Support Vector Machines
ToF: Time-of-Flight
TVR: Time Variable Replacement
TVRF: Time Variable Replacement Function

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Analyzing and understanding human behaviour is primordial to interact in a social environment. For humans, this ability is generally intuitive despite the fact that it results from a complex interpretation of the human body. Human behaviour can be reflected by emotions, actions, activities, etc. In this dissertation, we are mainly interested in human actions.

Generally, humans use their different senses to capture the relevant information related to human behaviour. For behaviour like actions which does not imply emotions, the most common sens used is the sight. This flow of information is then transferred to the brain in order to be analyzed. Based on the previous observed actions, the novel scene is associated to a label.

Nowadays, in an age where intelligent machines are more and more needed, the issue of automatic action recognition using visual sensors is increasingly attracting researchers. Because of the complexity of this task, it still represents an open issue. The main idea of this research topic is to imitate the real human understanding system and to create an artificial one which can be used in a wide range of applications. This subject lies at the crosswords of two main fields, which are computer vision and pattern recognition. Visual sensors play the role of the human visual system, since it captures the visual information and codes it in a conventional way. Then, computer vision tools allow the extraction of important features from the acquired data. Finally, pattern recognition methods represent a way to recognize actions using the extracted features. The most common used methods are the machine learning based methods, which require an a priori knowledge. Some important issues start already to emerge: What kind of visual sensor should we use? How to model actions using specific features? How to efficiently recognize actions? What are the criteria allowing the comparison of method performance? What are the limitations of each method?

This thesis aims to analyze and propose possible answers to the above mentioned questions.

In what follows, we present: the motivations that gave rise to this thesis, the scientific contributions that have been introduced in this work and the organization of this manuscript.

1.1 Motivations

Nowadays, action recognition algorithms represent an expanding trend in research due to their importance in numerous fields of application such as video surveillance, health, entertainment, etc.

The first generation of approaches has used classical cameras which provide Red Green Blue (RGB) images to recognize actions. However, this kind of sensor presents some limitations such as sensitivity to background extraction, illumination changes, etc. Recently, with the availability of low cost Red Green Blue-Depth (RGB-D) cameras, it has been noted a renewed interest for action recognition. This camera provides in real-time, additionally to RGB images, depth maps. Furthermore, a recent algorithm proposed by Shotton et al. [81] gave the opportunity to instantly extract skeleton sequences from depth maps, providing a third modality.

Motivated by the availability of these two novel modalities, this thesis aims to propose a fast and accurate action recognition system. Indeed, the accuracy as well as, the rapidity of calculation are two important criteria in real-world applications. For example, a method which is accurate, but very slow to compute tends to be unsuitable to the majority of real scenarios. In this way, the applicability in real conditions has motivated us to investigate in the subject of low-latency RGB-D based human action recognition.

1.2 Contributions

It is possible to summarize the different contributions proposed in this thesis as follows.

1.2.1 The evaluation of state-of-the-art methods in terms of computational latency and accuracy

Since we consider that the accuracy and the latency are both important parameters to take into account, we propose to evaluate some recent RGB-D based action recognition methods following a strict protocol. This protocol is carefully chosen in order to make the comparison as fair as possible using standard datasets. This study allows us to find the differences existing between depth-based descriptors and skeleton-based descriptors.

1.2.2 The proposal of a fast approach for temporal normalization: Time Variable Replacement (TVR)

Since one of the most challenging task in an action recognition system is the Temporal Normalization (TN), we propose a novel algorithm to make features invariant to action execution rate variability. This algorithm is called Time Variable Replacement (TVR). As the name of the method is indicating, the main idea is to make a change of variable by replacing time by another variable (invariant to velocity variation). To do that, two functions called Time Variable Replacement Functions (TVRF) are proposed which are the Normalized Accumulated kinetic Energy (NAE) and the Normalized Pose Motion Signal Energy (NPMSE). The most relevant characteristics of this algorithm are its rapidity and its efficiency.

1.2.3 The introduction of a descriptor for fast and accurate action recognition: Kinematic Spline Curves (KSC)

A novel descriptor which is fast and accurate is proposed in this work, called Kinematic Spline Curves (KSC). Furthermore, this descriptor has the advantage to be spatially and temporally invariant. It is built based on the interpolation of skeleton kinematic features using a cubic spline approach. A skeleton normalization, an alignment algorithm as well as the Time Variable Replacement (TVR) method are respectively used to avoid the anthropometric variability, the viewpoint variation and the execution rate variability.

1.2.4 The extension of kernel-based methods for Symmetric Positive semi-Definite matrices (SPsD)

We propose to extend kernel-based methods, normally designed for features belonging to a Euclidean space, to the space of Symmetric Positive semi Definite matrices (SPsD). Indeed, there exist some descriptors such as covariance matrices which are elements of the SPsD space, which is not a Euclidean space. In this way, in order to design a kernel based recognition method exploiting descriptors that belong to an SPsD space, a novel distance inspired by the Log-Euclidean distance, called Modified Log-Euclidean, is proposed.

1.2.5 The proposal of a novel descriptor called Hierarchical Kinematic Covariance (HKC)

We propose a second new descriptor which is fast and accurate, but also extensible to online applications. This descriptor which takes advantage from statistical and kinematic aspects, is calculated by integrating Kinematic values as features in the covariance matrix. Furthermore, to take into account the temporal information which is lost in covariance matrices, we propose to follow a hierarchical strategy by calculating this covariance matrix on different sub-segments of a sequence. The resulting final descriptor containing four Kinematic Covariance (KC) matrices is called Hierarchical Kinematic Covariance (HKC) descriptor. To classify actions using this representation, a Multiple Kernel Learning (MKL), based on an extended version of RBF (Radial Basis Functions) kernel for SPsD matrices, is trained.

1.3 Manuscript organization

This manuscript is organized as follows: Chapter 2 presents an overview of the state-of-the-art related to our work. In Chapter 3, we present a comparative study of RGB-D based human action recognition methods. Chapter 4 introduces the novel descriptor KSC, as well as the experimentation showing its efficiency compared to state-of-the-art methods. In Chapter 5, two descriptors making use of covariance descriptor are proposed: the Kinematic Covariance descriptor and the Hierarchical Kinematic Covariance descriptor. Furthermore, to carry out the classification, kernel learning methods are extended to the space of SPsD matrices. The validity of this approach is also shown by experimentation. Finally, Chapter 6 contains the conclusion of this work and presents future works.

Chapter 2

State-of-the-art

2.1 Introduction

Over the last five decades, understanding human motion has been an active topic of research in the fields of computer vision and pattern recognition. The growing interest of scientists for this issue is certainly due to the wide range of possible applications in several areas as in Human Computer Interaction, e-health, video surveillance, etc. This chapter is devoted to human action recognition state-of-the-art presentation. In Section 2.2, a general overview of human action recognition methods is given. Then, Section 2.3 presents novel human action recognition methods making use of RGB-D cameras. Since the latency represents an important criterion in many real-world applications, Section 2.4 outlines the issues of fast and online RGB-D based action recognition and summarizes recent methods.

2.2 Overview of human action recognition methods

In computer vision, the main idea of human motion understanding is to recognize actions from videos which in reality represent sequences of images. Figure 2.1 illustrates an example of Human Motion Understanding (HMU) system.

We start by recalling important terminology related to human behaviour understanding. Then, an overview of human action recognition methods is given. After that, since RGB devices have been widely used for the task of human behaviour understanding, RGB based action recognition methods as well as their limitations are presented. Subsequently, 3D

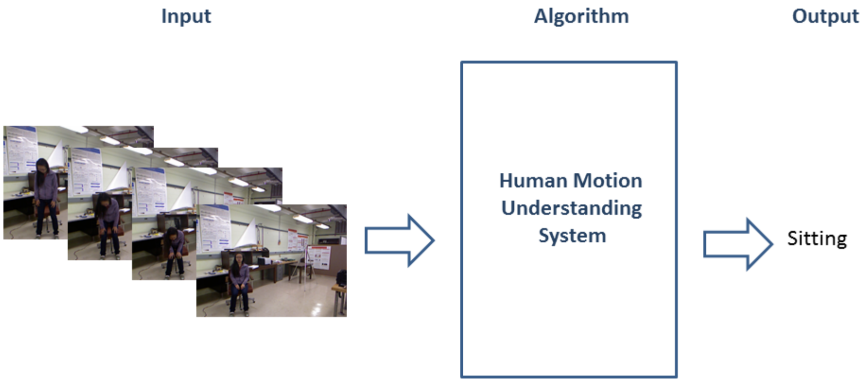


FIGURE 2.1: An example of a human motion understanding system: In this example, the input data represents an RGB video (a sequence of RGB images) where we can observe someone sitting. Therefore, using these input data, the system is able to recognize the observed action and returns as output its label "sitting".

visual sensors (with a focus on RGB-D cameras), which offer novel possibilities in the field of human action recognition, are reviewed. Finally, the positive impact of RGB-D cameras on action recognition systems is shown by giving numerous examples on various fields of applications.

2.2.1 Taxonomy

To distinguish between different levels of human motion, we adopt the same terminology defined in [87, 2, 26]. Therefore, human motions are categorized into three different sets, namely, *gesture*, *action* and *activity*.

Gestures

Gestures are the atomic elements of motion, which are visually perceptible by humans and which are easily annotated. This kind of motion involves only one part of the human body, such as the arm, the head or the leg, etc. In general, gestures are very brief, lasting only a few seconds. Furthermore, they are realized without the use of any object. Examples of gestures:

waving an arm, raising a foot, raising an arm, etc.

Actions

Actions are defined as a temporally organized sequence of gestures. Thus, such a variety of motion can include the movement of more than one body part, contrary to gestures. The duration of actions is obviously more important than the gesture duration and it can last up to one minute. Actions can involve an object, but this one has to be present from the beginning to the end of the action. Examples of actions: jumping, walking, running, swimming, swinging a golf ball, throwing an object etc.

Activities

Activities represent the highest level of motion and are composed of a sequence of actions. They involve interactions with objects, making them more complex to understand and to recognize. Moreover, the knowledge of the context is sometimes necessary to evaluate correctly the activity. Their duration is relatively important compared to actions and gestures (order of minutes). Examples of activities: talking on the phone, eating an apple, playing video games, etc.

In this thesis, we focus essentially on simple motions which include the two first kind of movement: gestures and actions. However, it is important to specify that the expression action recognition in the scientific community is general and can involve very often the three kinds of motion.

2.2.2 Schema of a human action recognition system

As specified above, action recognition represents a very challenging issue in computer vision and pattern recognition. Several attempts have been made in order to propose suitable solutions to automatically recognize actions. Generally, the task of action recognition can be divided into two main steps: *action description* and *action classification*.

Action description

The step of action description, involving principally the field of computer vision, is primordial for the good functioning of an action recognition system. It consists in extracting the useful information (called features) from the input data which mostly represents an image sequence and then in modeling it in a formal way. This step depends very heavily from the sensor that is used to capture the motion. As shown in Figure 2.2, this step can also be viewed as the succession of two processes, which are: *Feature extraction* and *Feature representation*.

Feature extraction: The huge amount of information contained in images makes the automatic recognition from raw data difficult. For this reason, the extraction of relevant features from images which have the power to discriminate actions is necessary.

Feature representation: The number of extracted features can vary from a video to another, since the number of frames varies. Final features (descriptors) should have the same size for each sequence of images in order to make their comparison possible during the classification. This step allows the expression of extracted features in a same and unique space, whatever the input data (RGB images in Figure 2.2) is.

Action Classification

This step, involving the field of machine learning, aims at assigning a label to each video based on the chosen feature representation. To do that, a classification model is learned based on training data. The training phase consists in allocating to each region of the feature space a label. Then, during the classification phase, the descriptor of each new input video is calculated, allowing the prediction of its label according to its position in the feature space.

Figure 2.2 illustrates these different steps. Features are first extracted from the images, corresponding to trajectories of particular points in this example (Feature Extraction). Then, the features are represented in a same space S, giving a final descriptor D (Feature Representation). After that, an action classification step allows the identification of action labels. Training data are used to separate the space S in different regions i according to the associated label C_i (Training). Then, this separation corresponding to the classification model is used in order to predict the label of new videos (Prediction).

2.2.3 RGB based action recognition methods and their limitations

The first generation of action recognition methods is mainly based on classical cameras which provide a sequence of RGB (Red Green Blue) images. Many surveys can be found in the literature, exposing different RGB-based methods for action recognition [69, 97]. Because of the high amount of information contained in RGB images, a pre-processing of human detection is required, before the beginning of the action recognition task. Since RGB

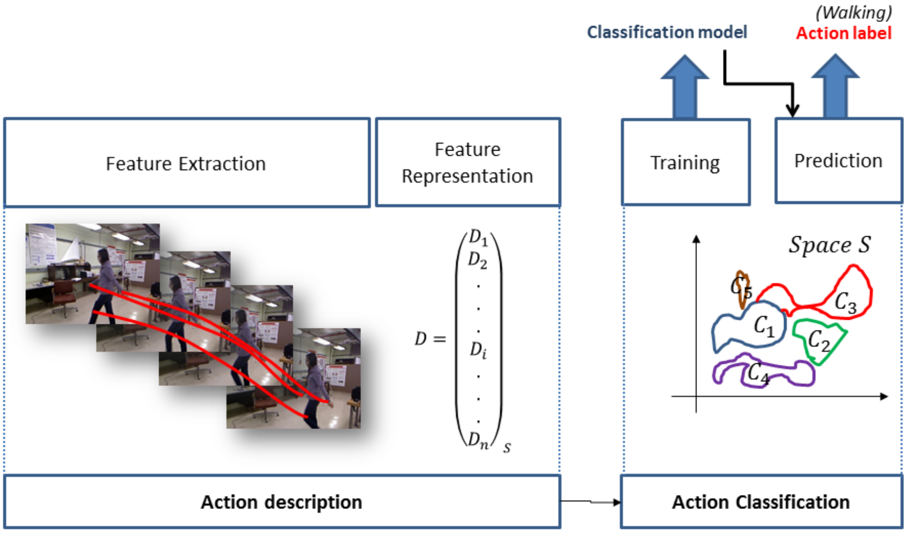


FIGURE 2.2: Schema of an action recognition system: In this example, as illustrated by images the input represents a woman walking (from UTKinect dataset [100]). The red trajectories represent the extracted features. Then, using these features, a descriptor D is built. Thanks to the descriptors calculated from training data, a classification model is estimated, separating the feature space S in various regions C_1, C_2, C_3, C_4, C_5. Finally, by placing the descriptor extracted from an input data in the space S, the system is able to recognize the label of the action (walking) according to its membership of one of the estimated regions.

images can be viewed as matrices where each cell contains the RGB color information, advanced suitable segmentation and tracking algorithms are required. The sensitivity of RGB images to many factors impacts on the accuracy of RGB based action recognition methods. The main limitations of RGB-based approaches are: the sensitivity to background extraction, to viewpoint variation and to illumination changes and occlusions.

Viewpoint variation

RGB based approaches are generally affected by the viewpoint variation, as the nature of data is two dimensional. Images (a1) and (a2) in Figure 2.3 show an example of viewpoint variation. In both images, the performed action is "walking". However, the orientation of the actor body is different. This fact can make the descriptors extracted from each video

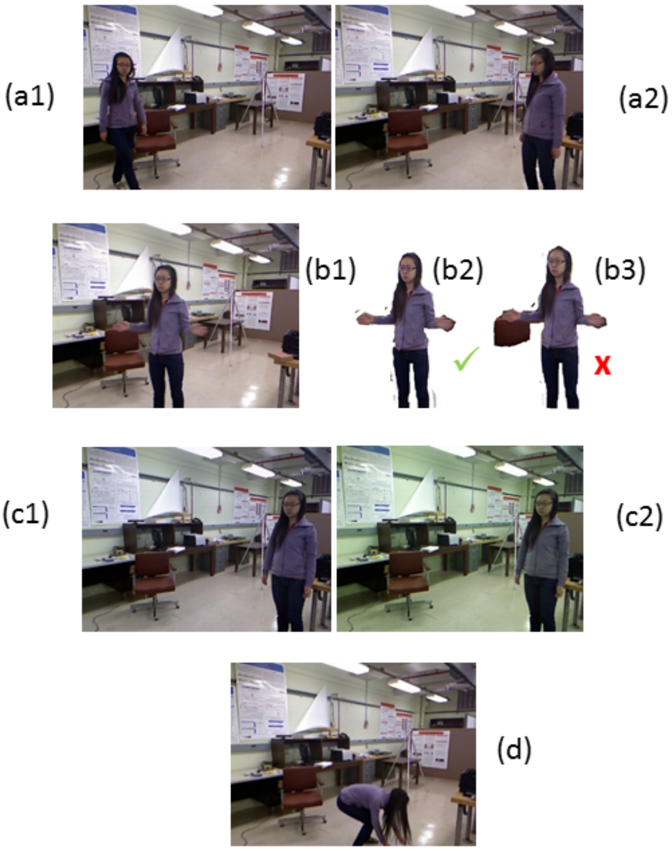


FIGURE 2.3: Examples of viewpoint variation (a1,a2), background extraction (b1,b2,b3), illumination changes (c1,c2) and self-occlusion (d)

containing a specific action very different.

Background extraction

Because the relevant information related to the action is located only on the human body, a process of segmentation has to be performed. It is also called background extraction because the background which contains useless information is removed. Nevertheless, the segmentation is sometimes not perfect. Figure 2.3 illustrates two examples of background extraction: Image (b1) represents the original image before segmentation, Image (b2) represents Image (b1) after a correct segmentation and Image (b3) also represents Image (b1) but after a biased segmentation. It is easy to imagine that a wrong segmentation will badly affect the descriptor extraction step and consequently the action recognition task.

Illumination changes

Since the information of RGB images is based on colors, illumination changes can directly influence the content of features. Images (c1) and (c2) of Figure 2.3 represent the same image but with different illumination conditions.

Occlusions

Occlusions occur when a part of the human body is hidden by an object or another part of the body. In the second case, we speak, more precisely about self-occlusion. As RGB images are not three dimensional, occlusions can lead to many errors in the context of action recognition. Even, if objects are handled during the acquisition, self-occlusions can occur as presented in Image (d) of Figure 2.3. Indeed, a part of the arm is hidden by the subject hair.

Recently, 3D visual sensors have been introduced in order to overcome these limitations which are mainly due to the 2D structure of RGB images. In the next section, an overview of these 3D acquisition systems is presented.

2.2.4 3D visual acquisition systems

Automatically recognizing actions using RGB images proves to be difficult because of the limitations of the sensor used as presented in the previous section. The emergence of novel technologies allowing the capture of 3D data has renewed the interest of researchers in the field of pattern recognition and more specifically in action recognition. In a recent survey, Aggarwal and Xia [3] summarize the majority of 3D data acquisition systems used in human action analysis applications.

One of the first systems proposed to obtain the 3D information of the scene was the stereo vision system. Inspired by the human vision which is based on two eyes (two sensors), this visual system is composed of two different RGB cameras with different viewpoints used to generate a disparity image (which contains the depth information). To fuse the two images coming from two independent sources, they make use of stereo correspondence algorithms which have the role to find the point correspondences existing between every image pair. They mostly require the use of epipolar geometry notions. A recent review of stereo vision algorithms is presented in [86]. Some researches have explored human activity recognition using stereo vision systems [89, 43]. Nevertheless, the 3D reconstruction from stereo images is still an open problem due to geometric difficulties. Indeed, the matching task is often biased by the lack of texture, depth discontinuities, transparencies and reflections, etc.

These limitations have participated to reduce the use of stereoscopy in human behaviour understanding applications.

A second technology used to obtain the human 3D information is the Motion Capture (MoCap) system. This kind of acquisition system is very practical for human tracking and human behaviour analysis. The majority of MoCap systems such as VICON are based on markers placed on specific zones of the human body. These markers allowing the tracking of human motion return their 3D positions over time. Since the 21^{th} century, many human action recognition methods with the use of MoCap data have been proposed [79, 57, 36]. These approaches are based generally on joint positions, joint angles, etc. These devices have proven their high accuracy and their efficiency in many applications. However, they present three major drawbacks: they are very expensive, they are hard to displace and human subjects need to wear markers. These constraints limit their applicability in many scenarios as in video surveillance or gaming.

Recently, low-cost real-time depth sensors have emerged such as Kinect or Asus Xion PRO LIVE. These devices are also called RGB-D cameras, as they provide two kind of images. Additionally to the classical RGB images, RGB-D sensors also capture depth images. To provide depth images, two strategies are proposed: Time-of-Flight (ToF) and Structured-Light (SL). These two techniques are compared in the paper of [75]. The Structured-Light technique represents an active stereo vision approach. It consists in projecting known patterns on the observed scene and in measuring the distortion of these patterns to estimate the depth image. On the other hand, Time-of-Flight technique consists in emitting a light and in estimating the deepness based on the time that takes this light to arrive to the object. For example, Kinect 1 is based on the Structured-Light technique while Kinect 2 is based on the Time-of-Flight technique. These two sensors are illustrated in Figure 2.4.

Thus, RGB-D cameras offer an additional modality: depth images. Furthermore, a recent algorithm proposed by Shotton et al. [81] allows the extraction in real-time of human skeletons from depth images. Therefore, a third modality is easily provided which is human skeleton sequences. These skeletons are in reality a set of human joints with the knowledge of their 3D position. Figure 2.5 shows an example of the two modalities: a depth map and a human skeleton.

In this way, RGB-D cameras which offer multiple modalities, are easy to displace, are relatively cheap and represent an interesting way to be used as sensors for human action recognition. For this reason, in this thesis, we will focus on RGB-D based human action recognition.

FIGURE 2.4: Examples of RGB-D cameras: Kinect 1 (on the top) and Kinect 2 (in the bottom)

2.2.5 The impact of RGB-D cameras on action recognition applications

The increasing interest of researchers for human action recognition is mainly due to its utility in a large variety of applications. These applications include: Medicine and e-health, video surveillance, gaming and entertainment, rehabilitation, etc. The release of RGB-D cameras has encouraged researchers to improve these applications. In this section, the positive impact of RGB-D cameras on action recognition applications is described.

Medicine and e-health

Medical applications represent one of the most coveted applications in computer vision, because of the constant need of evolution in medicine. Simple algorithms can sometimes help clinicians in their medical tasks but also can help humans to improve their quality of life such as older people. Action recognition is attracting many medical engineering researchers due to the wide range of possible applications. For example, Lea et al. [52] have proposed a system which segments and recognizes fine-grained activities in order to help clinicians to train on surgeries. We can give another example of medical applications [59], where authors have introduced a fall detection system to secure old people. Also, Dubey et al. [28] proposed a similar system. On the other hand, Fosty et al. [34] have designed a recognition system for

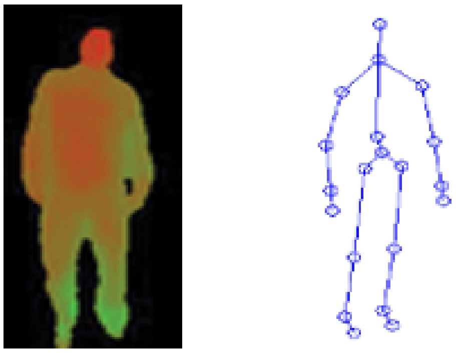


FIGURE 2.5: An example of depth (left) and skeleton (right) modalities

older people monitoring using an RGB-D camera.

Rehabilitation

Another very popular example of application is the rehabilitation. Some persons with physical disabilities can need this kind of treatment. In fact, the repetition of specific gestures can help them to reduce the negative effect of their disease on their motion. However, this therapy requires very often the presence of a specialist to ensure the correctness of the repeated gesture. The idea is then to replace this assistance by a camera which captures the gesture and returns its correctness or wrongness. We can cite many papers which has applied action recognition to rehabilitation such as [15, 1].

Video surveillance

Because of the importance of security, video surveillance represents an important field of application. This kind of technology can help to automatically detect abnormality or potentially dangerous activities, etc, allowing a faster intervention if necessary. When RGB cameras are used as sensors, the first step consists in tracking and identifying humans employing RGB images as in [56, 42]. Indeed, without this first step, the amount of image information is too important to start dealing with the task of action recognition. Segmentation and tracking processes, which represent themselves challenging issues in the field of computer vision, are therefore needed and make the issue of action recognition more and more difficult.

Recently, thanks to the emergence of RGB-D cameras, which made human segmentation easier, many researchers have proposed to develop action recognition systems for video surveillance [9, 29, 54, 48]. They make use of depth maps or skeleton data which are less sensitive to viewpoint variation, to weather changes, etc.

Gaming and entertainment
Another motivation for developing action recognition systems is their convenience for interactive games. Indeed, the Kinect camera (one of the most known RGB-D camera) has been initially produced for gaming, and more specifically for the Microsoft Xbox console.

Then, video games offer to the players the opportunity to interact spontaneously with the machine, without the need of a supplementary object (as in Nintendo Wii). For example, to virtually play tennis, it is sufficient to imitate the known gestures (like swinging, smashing, etc). In this way, many papers aim to explore this kind of industrial applications [8, 73].

As presented in this section, RGB-D cameras have allowed the improvement of many action recognition applications. Thus, the next section presents a detailed review of RGB-D human action recognition methods.

2.3 RGB-D based human action recognition

During this last decade, various methods have been proposed for human action recognition based on the novel modalities originating from RGB-D cameras. This is certainly due to numerous advantages that present this kind of devices, as described in the previous section. Recent reviews of RGB-D based action recognition approaches are presented in [3, 108].

The latter [108] has categorized RGB-D based action recognition methods into two distinct groups according to the representation of actions: hand-crafted representations and learning-based representations.

Hand-crafted methods are the most common approaches used in the literature [63, 91, 94]. They are based on the classical schema of action recognition, where low-level features are first extracted, then the final descriptor is modeled using low level features and finally a classifier is used to train a classification model such as Support Vector Machine (SVM) or k Nearest Neighbors (kNN).

With the recent advances in deep learning, learning-based representations have been proposed. Instead of selecting specific features, this category of methods learn itself the appropriate ones as in [44, 61, 70].

Since this work focuses on human action description and more specifically on hand-crafted methods, the categorization is carried out according to the used modality and therefore according to the nature of the human action descriptor. In this way, we distinguish

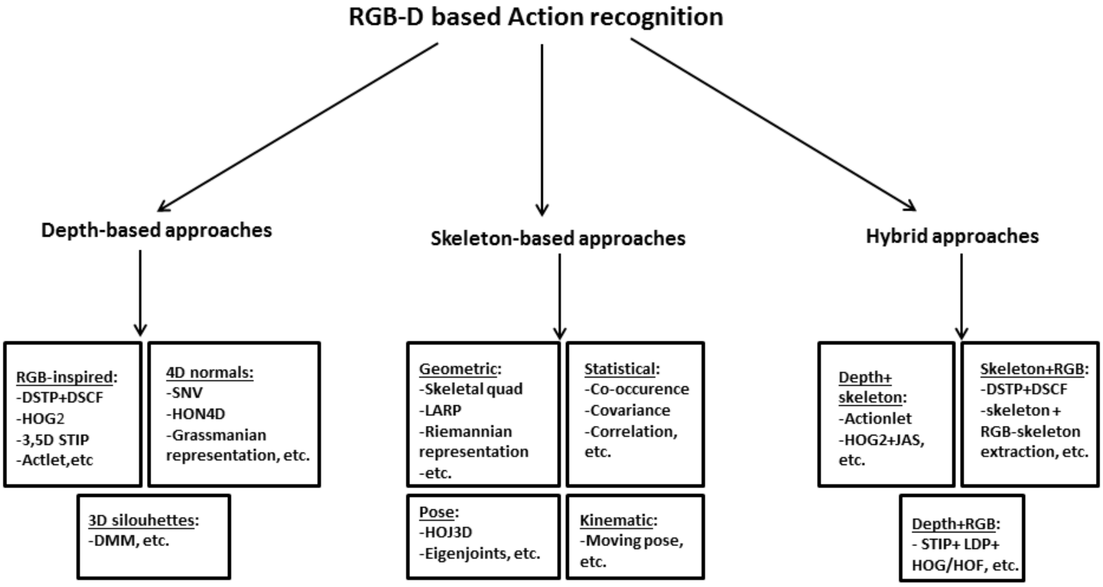

RGB-D based Action recognition
Depth-based approaches
Skeleton-based approaches
Hybrid approaches
RGB-inspired: -DSTP+DSCF -HOG2 -3,5D STIP -Actlet,etc
4D normals: -SNV -HON4D -Grassmanian representation, etc.
3D silouhettes: -DMM, etc.
Geometric: -Skeletal quad -LARP -Riemannian representation -etc.
Statistical: -Co-occurence -Covariance -Correlation, etc.
Pose: -HOJ3D -Eigenjoints, etc.
Kinematic: -Moving pose, etc.
Depth+ skeleton: -Actionlet -HOG2+JAS, etc.
Skeleton+RGB: -DSTP+DSCF -skeleton+ RGB-skeleton extraction, etc.
Depth+RGB: - STIP+ LDP+ HOG/HOF, etc.

FIGURE 2.6: Overview of RGB-D based human action descriptors divided into three main groups: depth-based, skeleton-based and hybrid descriptors

between 3 different categories of approaches: depth-based approaches, skeleton-based approaches and hybrid approaches. Figure 2.6 presents an overview of RGB-D based action recognition state-of-the-art.

2.3.1 Depth-based human action recognition

This kind of approaches uses depth images providing from RGB-D cameras in order to describe human actions.

The first generation of methods tried to adapt descriptors initially designed for RGB images to depth images. Many recent papers have proposed to extend Spatio-Temporal Invariant Points [6] to depth maps.

Xia and Aggarwal [99] have proposed a novel algorithm to detect STIP in depth images and called them Depth Spatio-Temporal Interset Points (DSTIP). Figure 2.7 shows an example of these DSTIPs. Also, they have adapted the cuboid descriptor [27] to describe the

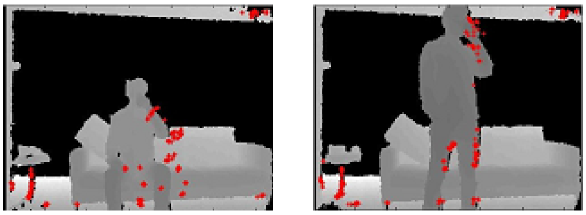


FIGURE 2.7: An example of extracted DSTIPs from two depth images in [99] containing the two following activities: drink-sit (left image) and drink-stand (right image)

DSTIPs, named Depth-Similarity Cuboid Features. The classification has been done using a Support Vector Machine classifier with a histogram intersection kernel.

At the same time, Hadfield and Bowden [40] introduced an extension of STIP to recognize actions in an unconstrained dataset. They called them 3.5D STIP since the depth maps have 2.5 dimensions other than time (2.5D+t). In this paper, different STIPs such as Harris Corners and Hessian points are compared. For the classification, an SVM model based on a Radial Basis Functions (RBF) kernel has been used.

Because of the popularity of the Bag-Of-Words approach and its variants in the field of RGB-based action recognition [74, 7], many attempts have been made to extend this kind of methods to depth modality. For instance, Foggia et al. [32] use a bag-of-words approach. They represent actions using a histogram which contains the occurrence information of feature vectors in a temporal window as in text retrieval and object recognition. They made use of SVM to carry out the classification. In [82], videos are divided in several temporal bins in order to include the full temporal information. To overcome the execution rate variability, it is also proposed to partition the video using bins of different lengths. In [12], actions are represented as strings, which are composed of symbol series. Each symbol corresponds to an *actlet*, defined as an atomic action. Figure 2.8 gives an overview of this approach. These two latter methods also used SVM for the classification step.

It may be noted in the literature, the extension of methods based on Histograms of Gradients (HOG) [23]. For example, Klaser et al. [49] proposed the HOG3D as a descriptor which represents a 3D dimensional Histogram. Few years later, Ohn-Bar and Trivedi [63] had the idea of representing actions using the combination of two histograms: one considers the spatial information, while the other analyzes the temporal information. A linear SVM has been used for the classification. However, some researchers rapidly pointed out the limitations of these adapted methods justifying their claims by the fact that depth modality has different properties and a different structure from color modality [65, 102].

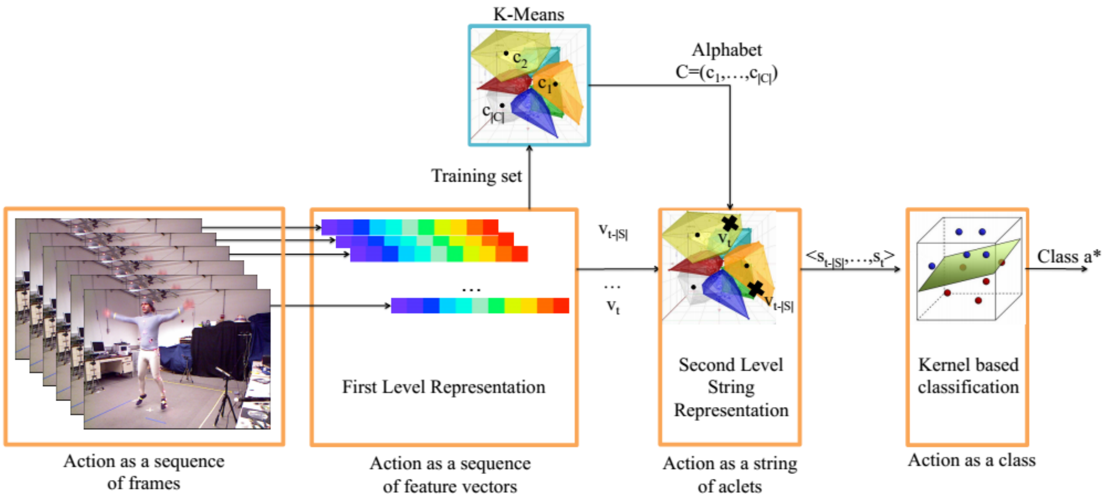

K-Means
c2
cj
c|C|
Alphabet
C=(c1,…,c|C|)
Training set
Vt-|S|
…
Vt
<st-|S|,…,st>
Class a*
First Level Representation
Second Level String Representation
Kernel based classification
Action as a sequence of frames
Action as a sequence of feature vectors
Action as a string of aclets
Action as a class

FIGURE 2.8: Overview of the method proposed in [12] where the notion of "actlet" is introduced: Feature vectors are first extracted from images. Then, each feature vector V_t is associated to a high level representation s_t thanks to the alphabet C learned during the training phase. Finally, the action is recognized using a kernel-based classification

On the other hand, some methods have been developed based on 3D silhouettes representations. For instance, Yang et al. [103] introduced Depth Motion Maps (DMM) features. It represents 3D human silhouettes projected on three orthogonal planes and accumulated over time. Then, an HOG is applied to DMM in order to obtain a compact descriptor. Finally, a linear SVM algorithm is trained for the classification. Chen et al. [16] also made use of this action representation. The main drawback of this kind of approaches is its important sensitivity to viewpoint variation.

Recently, less conventional depth-based descriptors have been also proposed. One of the most efficient representations that we can cite is the 4D normals of the human body. Figure 2.9 illustrates an example of human body normals.

This original idea has been introduced in the work of Oreifej and Liu [65] where the motion of the human body is considered as a hyper-surface of $\mathbb{R}^4$ (three dimensions for the spatial components and one dimension for the temporal component). In differential geometry, one of the well-known ways to characterize a hyper-surface is to find its normals. Thus, Oreifej and Liu chose these features and built the descriptors by quantifying a 4D histogram of normals using polychrons (a 4D extension of simple polygons [19]). As the previous cited paper, they make use of a linear SVM approach in order to classify actions. Figure 2.10 illustrates the proposed framework in [65].

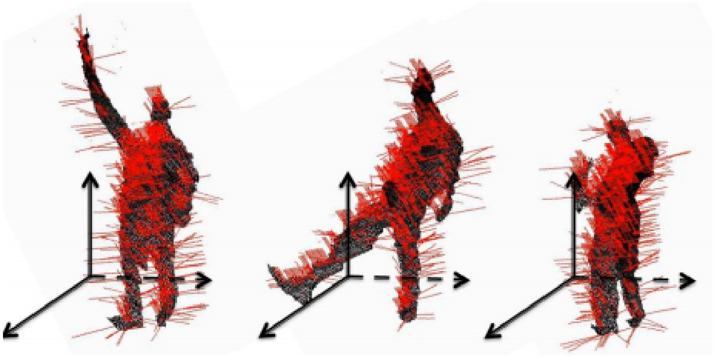
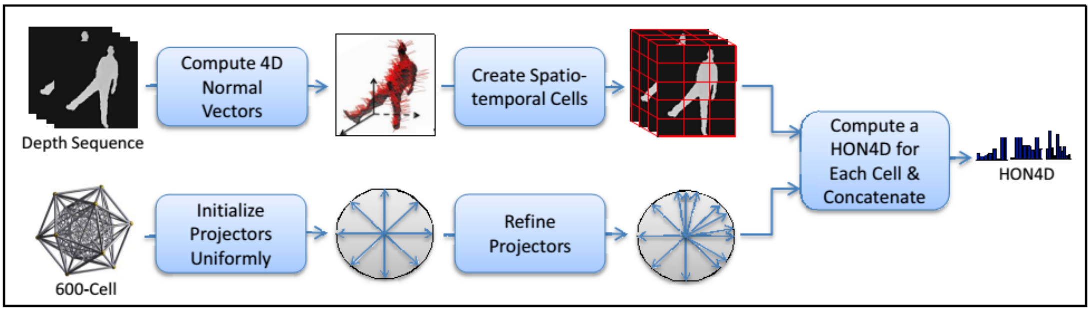


FIGURE 2.9: Example of human body normals in [65]: This figure illustrates 3D normals and not 4D normals (since they are hard to represent).



Depth Sequence
Compute 4D Normal Vectors
Create Spatio-temporal Cells
Compute a HON4D for Each Cell & Concatenate
HON4D
600-Cell
Initialize Projectors Uniformly
Refine Projectors

FIGURE 2.10: The framework proposed in [65] to compute the descriptor HON4D

Many authors have been inspired by this relevant representation, but changed the quantization step (because the neighboring information is lost with the use of HON4D). Slama et al. [83] have modeled 4D normals as subspaces lying on a Grassmann manifold. We refer to this method as "Grassmannian representation" in Figure 2.6.

At the same time, Yang and Tian [102] have used the same representation by learning a sparse dictionary and then by coding each action as a word of the learned dictionary, as described by Figure 2.11. The obtained descriptor was called the Super Normal Vector (SNV). These two papers have also used an SVM classifier. This new generation of features based on depth modality has shown its high accuracy compared to other representations. Notwithstanding this important advantage, they generally require considerable computation time.

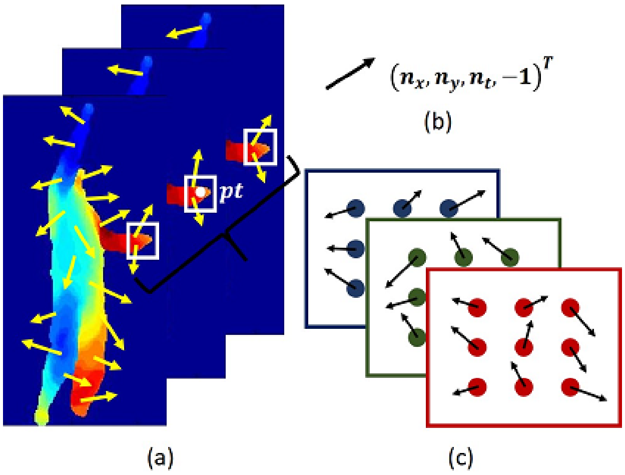


FIGURE 2.11: Illustration of SNV descriptor computation in [102]

2.3.2 Skeleton-based human action recognition

Thanks to the development of skeleton extraction algorithms such as Shotton et al.'s algorithm [81], a relatively accurate and real-time human skeleton extraction from depth maps has become possible. This fact has motivated researchers to explore skeleton sequences in order to recognize actions.

Since skeleton modality represents in reality a set of n 3D points with the knowledge of their position, first attempts have directly based their features on the position of these 3D points (on the pose of the skeleton). As an example, Xia et al. [100] proposed to model skeleton posture using a Histogram of 3D joints (HOJ3D). To select the dominant features, a Linear Discriminant Analysis is applied to HOJ3D. After that, the features are clustered into k words using k-means. Finally, the classification is done using Hidden Markov Models (HMM) which models the dynamic evolution. However, this method is sensitive to viewpoint variation because of the influence of the anthropometric variability.

Thus, Yang and Tian [101] introduced a novel descriptor based on the relative position of joints. As shown in Figure 2.12, three values are calculated (f_{cc}), (f_{cp}) and (f_{ci}) which respectively are the distance between the joints of the current pose, the distance between the current pose and the previous one and the distance between the current pose and the initial one. Then, these values are concatenated and normalized. A Principal Component Analysis (PCA) is applied to the feature vector to reduce its important dimension. The obtained descriptors, called eigenjoints, are therefore used for the classification step based on a Naive-Bayesian classifier.

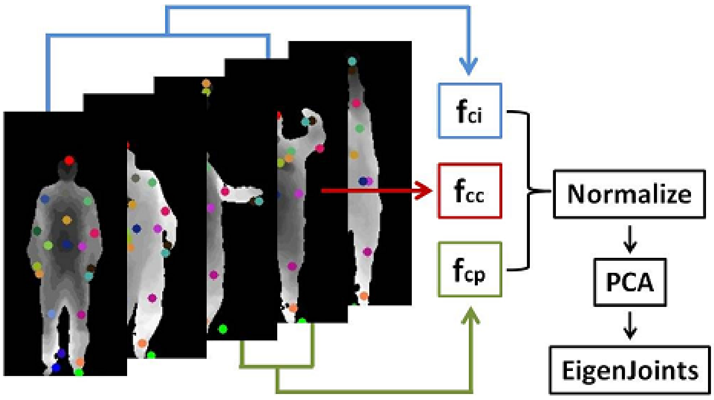


FIGURE 2.12: Illustration of the action classification method of [101]

Pose-based approaches are interesting but are clearly less accurate than more recent ones. Over the last years, it has been noted the emergence of more sophisticated descriptors.

A second class of skeleton-based approaches are the geometric methods. This kind of descriptor is designed by representing skeleton motions using geometric concepts.

In [30], skeletal quads are introduced. As illustrated in Figure 2.13, these features represent quadruples composed of four adjacent joints which contain the information of similarity between segments. A Gaussian Mixture Model (GMM) is learned to encode skeletal quads into Fisher Vectors. The classification is achieved by a linear SVM.

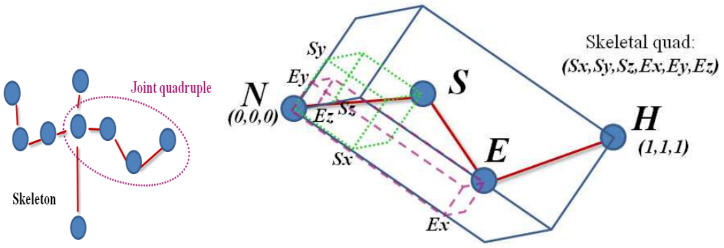


FIGURE 2.13: Skeletal quad representation calculated based on 4 joints introduced in [30]

Other works manipulate differential geometric notions to build human action descriptors such as [26, 91].

In [26], the main idea is to calculate a similarity shape measure between joint trajectories. To realize that, these curves are reparametrized using the Square-Root Velocity Function (SRVF), leading to a representation in a Riemannian manifold. A k-Nearest Neighbours

(kNN) algorithm is used to recognize actions. In Figure 2.6, this method is referred to as Riemannian representation.

In [91], actions have been represented by associating to each couple of neighbour segments a transformation matrix T (with T an element of the Special Euclidean group $SE(3)$). Each skeleton (composed of n joint connections) of the sequence is represented by a point of $SE(3)^n$, as illustrated in Figure 2.14.

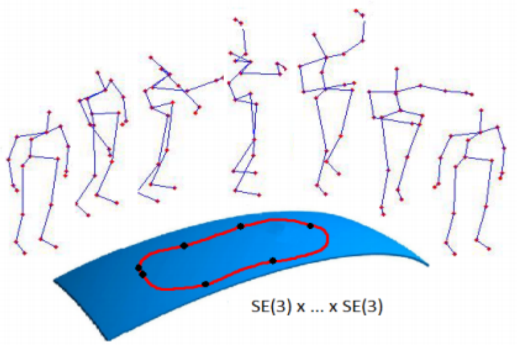


FIGURE 2.14: Action representation in the Lie algebra [91] of $SE(3)^n$

These points evolving over time are interpolated in order to describe the motion using trajectories. Since $SE(3)^n$ is not a vector space and represents a Lie group, the data are mapped to the Lie algebra of $SE(3)^n$ named $se(3)^n$. The classification is done using a combination of Dynamic Time Warping algorithm (DTW), Fourier Temporal Pyramid (FTP) and SVM. This algorithm has shown its efficiency. However the high amount of approximation can lead to low accuracy if the data are noisy and to an important calculation time.

It can be noted the emergence of methods based on kinematic aspects. Since the skeleton representation has been very often used in bio-mechanic studies [47], many papers have based their work on kinematic entities such as position, velocity and acceleration of joints. Zanfir et al. [105] proposed to concatenate these features and to weight each term by an empirical value. They classified actions using a kNN algorithm.

Finally, some papers proposed statistical-based approaches. This kind of approaches uses statistical tools in order to propose a discriminative action representation. It can be noted in the state-of-the-art an important interest for covariance descriptors. In [109], they use the notion of co-occurrence of joints. To learn these features, they made use of Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM). In [45], joint positions are used to build a covariance descriptor, as illustrated in Figure 2.15. Because covariance matrices are symmetric, only the upper triangle of the matrix is considered and converted into a vector.

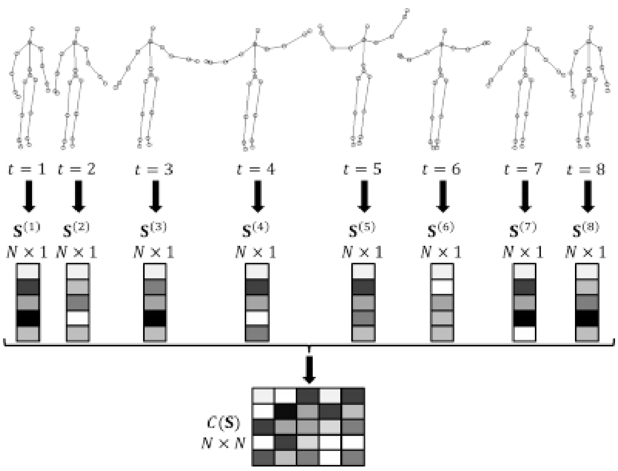

$t=1$
$t=2$
$t=3$
$t=4$
$t=5$
$t=6$
$t=7$
$t=8$
$\mathbf{s}^{(1)}$ $N \times 1$
$\mathbf{s}^{(2)}$ $N \times 1$
$\mathbf{s}^{(3)}$ $N \times 1$
$\mathbf{s}^{(4)}$ $N \times 1$
$\mathbf{s}^{(5)}$ $N \times 1$
$\mathbf{s}^{(6)}$ $N \times 1$
$\mathbf{s}^{(7)}$ $N \times 1$
$\mathbf{s}^{(8)}$ $N \times 1$
$C(\mathbf{S})$ $N \times N$

FIGURE 2.15: Covariance descriptor calculation based on joint positions presented in [45]

This vector is then used to train a linear SVM model. This work did not take into account the particular geometry of the covariance space assuming a vector space structure.

In [85], where an action recognition framework is proposed, covariance matrices are assumed to be symmetric positive definite. If the matrix is singular, the nearest symmetric positive definite matrix is found which can be very expensive in terms of computational cost. A kNN algorithm is trained to classify actions using a geodesic distance defined on the space of Symmetric Positive Definite matrices (the Log-Euclidean distance instead of using the Euclidean distance).

2.3.3 Hybrid human action recognition

This kind of approaches combines two or three modalities in order to make the recognition more efficient. Indeed, in real-world scenarios, the quality of depth maps (and therefore of the skeletons) can be affected by various facts such as the distance between the subject and the camera or the outdoor environment.

To resolve that, one solution is to use, further to depth maps or skeletons, RGB images [24, 107, 110]. For example, Das et al. [24] proposed to extract skeletons from depth maps and also from RGB images using CNN methods. Then, they merged the two skeletons to improve the accuracy of recognition and to reduce the impact of noisy depth maps. In Figure 2.6, this method is referred to as "skeleton+RGB-skeleton extraction".

Also, in [110], two modalities are used: RGB images and skeleton sequences. They made use of Spatio-Temporal Interest Points (STIP) extracted from RGB images, as well as pairwise distance between skeleton joints. Random Forests are used to fuse the features but also to classify the actions.

Other hybrid methods proposed to fuse the skeleton and the depth modalities. These methods are very useful in the task of activity recognition, when we observe interactions with objects. Generally, skeleton alone is not sufficient to model activity implying the use of objects.

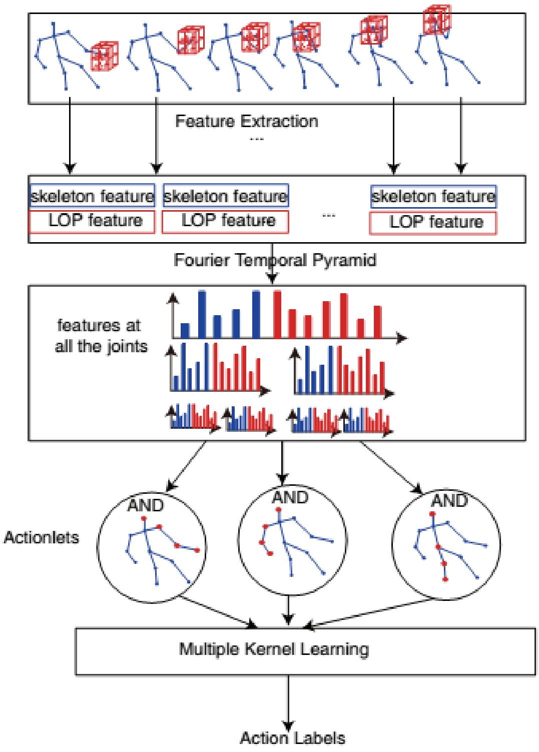


FIGURE 2.16: Overview of the framework proposed in [94] based on Actionlet computed using skeleton features and LOP

Wang et al. [94] proposed a novel Actionlet model. To do that, two different kinds of features are employed: skeleton-based features and depth-based features. Indeed, pairwise-distance between joints as well as Local Occupancy Patterns (LOP) are respectively extracted from skeleton sequences and depth maps. To capture the temporal dynamics of features, they propose the Fourier Temporal Pyramid (FTP). Actionlets are therefore defined as a conjunctive structure on the base features while each base feature is defined as a Fourier Pyramid feature of one joint. To select the discriminative actionlets, a data mining algorithm is also proposed. The recognition is done using an SVM classifier based on a Multiple Kernel Learning (MKL) strategy. Figure 2.16 presents an overview of this method.

Another work [63] proposed to extract HOG2 from depth images and Joint Angle Similarities (JAS) from skeletons as illustrated in Figure 2.17. HOG2 features have been already presented in depth-based approaches, while JAS are calculated by measuring the similarity between joint angles. To perform classification, both features are combined and a linear SVM model is used.

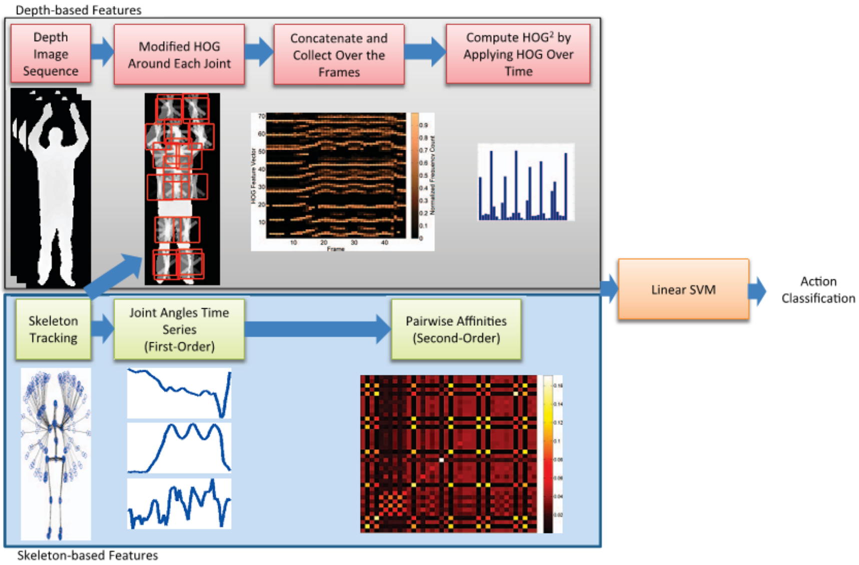

Depth-based Features
Depth Image Sequence
Modified HOG Around Each Joint
Concatenate and Collect Over the Frames
Compute HOG² by Applying HOG Over Time
Linear SVM
Action Classification
Skeleton Tracking
Joint Angles Time Series (First-Order)
Pairwise Affinities (Second-Order)
Skeleton-based Features

FIGURE 2.17: Overview of the framework based on HOG2 and JAS descriptors proposed in [63]

The methods presented here have shown their efficiency in terms of accuracy. However, in real-world applications, other criteria related to the latency are also important. For this reason, the next section is dedicated to RGB-based online and fast action recognition which study in addition to the accuracy, the latency.

2.4 RGB-D based online and fast action recognition

Since a real-time action recognition is required in the majority of real-world applications, some attempts start to be made in this direction. In [58], the main challenge of real-time systems is presented as the decrease of latency. They defined the latency as the sum of the observational latency and the computational latency. The computational latency represents the time needed for computation, while the observational latency represents the time of observation required to make a good decision. The majority of papers dealing with real-time systems has focused on reducing this observational latency and has called this task online action recognition or online recognition (when the videos are not segmented). In this way, we propose to define fast action recognition as the task of recognition aiming to reduce the computational latency. In this part, the two different notions related to real-time action recognition are presented, namely fast action recognition and online action recognition. The section dedicated to online recognition is called untrimmed videos, since the main goal of these methods is to deal with non segmented videos.

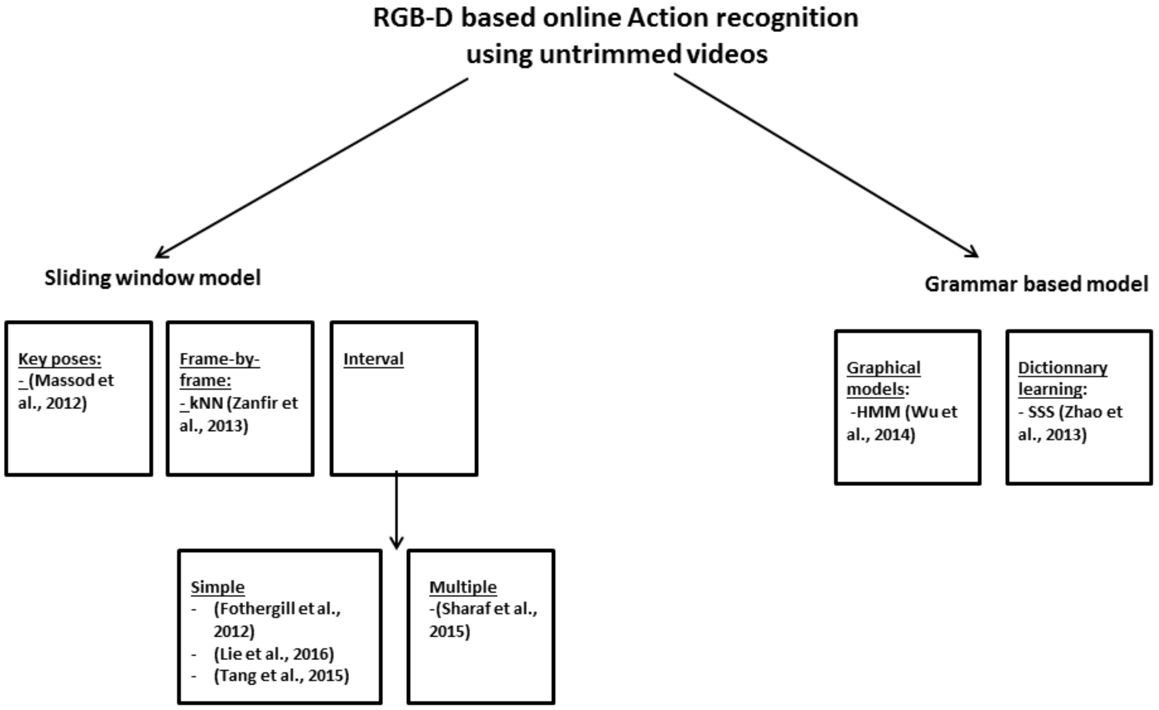

RGB-D based online Action recognition using untrimmed videos
Sliding window model
Grammar based model
Key poses: - (Massod et al., 2012)
Frame-by-frame: - kNN (Zanfir et al., 2013)
Interval
Simple - (Fothergill et al., 2012) - (Lie et al., 2016) - (Tang et al., 2015)
Multiple -(Sharaf et al., 2015)
Graphical models: -HMM (Wu et al., 2014)
Dictionnary learning: - SSS (Zhao et al., 2013)

FIGURE 2.18: Overview of RGB-D based online action recognition state-of-the-art

2.4.1 Fast action recognition

As mentioned before, fast action recognition represents the ability to recognize segmented actions with a low computational latency. It is very useful for many applications. Indeed, many off-line applications still require a fast answer for a good functioning such as rehabilitation, coaching, etc. Fast action recognition has been studied by the past with the use of

RGB cameras. We can cite in particular the following papers [17, 38, 104]. Some works making use of RGB-D cameras have outlined the computational latency aspect by reporting the complexity or the execution time of their algorithm, as in [58, 13, 26]. However, to the best of our knowledge, there do not exist a comparison between state-of-the-art methods in terms of computational latency.

2.4.2 Untrimmed videos

As explained before, online action recognition aims to reduce the observational latency. Furthermore, this kind of system has to be able to deal with unsegmented actions by detecting the beginning and the end of actions. Thanks to the success of RGB-D based action recognition methods, it could be noted a novel interest for online RGB-D based action recognition. However, this very challenging task is still an open problem because of its difficulty. Indeed, it includes: accurate action recognition, early action recognition and real-time action detection. Existing methods can be divided as follows: grammar based models and sliding window based models. In this part, an overview of these cited methods is given.

Sliding window based models

These approaches are based on a sliding window allowing the recognition at each instant based on the information contained in this window. This latter advances over time by integrating new frames and deleting others.

Some of these approaches are based on key frames. Indeed, some researches have shown that it is possible to recognize some actions based only on key frames which discriminate actions. For example, for the action raising two hands, the key frame is certainly produced at the instant where the subject has his two hands raised. Masood et al. [58] proposed to detect in a sliding window the key pose, which is called in their paper canonical pose. To do that, they introduced a max-response for each class and then based on all the max-responses obtained, the probability of belonging to a specific class is calculated by following a Multiple Instance Learning strategy.

On the other hand, some methods have been designed in order to follow a frame-by-frame strategy. At each instant, an action is assigned to every frame contained in the window. After that, the final label is obtained by fusing all the labels attributed to each frame. As an example, we can give the work of [105]: they proposed to assign to every frame a label using a kNN classifier. Then, a majority vote is applied in order to fuse the label information.

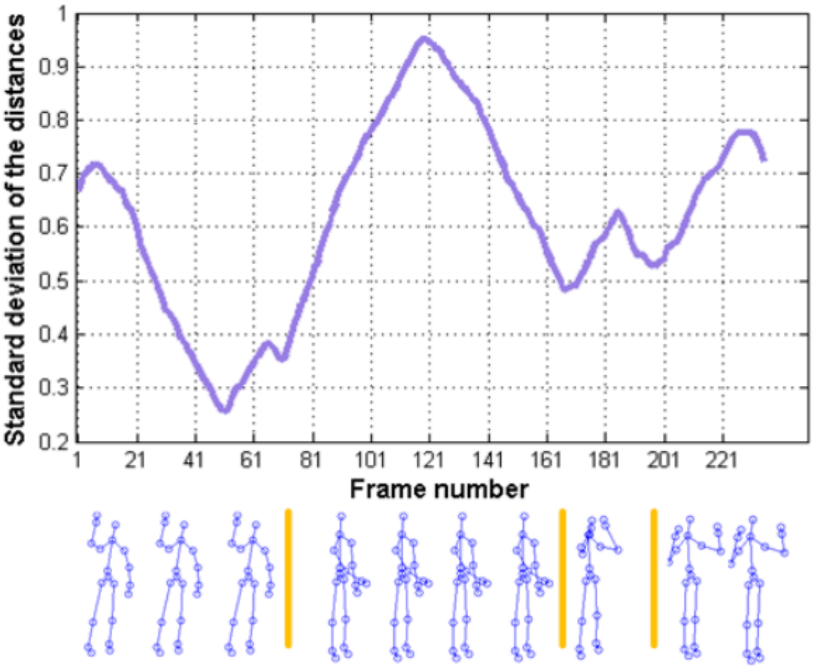


FIGURE 2.19: The standard deviation of the covariance distances calculated to detect action transitions in [85]

Another work proposed to use the entire information contained in the sliding window by calculating a descriptor and carrying out a direct classification. In [85], a covariance descriptor is calculated using all the frames in the sliding window. Then, to classify actions, the minimal distances between the input data and the training data are used. To detect the changing of actions, the standard deviation of minimal distances are used, as shown in Figure 2.19. Indeed, the closer the minimum distances are, the lower the standard deviation is and the more a transition from an action to another is possible.

However, the previous cited works which used a fixed length window remain sensitive to execution rate variability. In fact, the length of an action is variable due to the action execution rate variability. In this way, Sharaf et al. [78] proposed to use a window with multiple scales. The extracted descriptors are mined using a feature selection. Finally, they are classified with the use of an SVM algorithm. Figure 2.20 summarizes the different steps of this approach.

Grammar based models

This kind of representations aims to model actions as a sequence of states or words, depending on the used technique. For example, Zhao et al. [106] have introduced Structured Streaming Skeleton (SSS) which is built using a dictionary learning. An overview of this

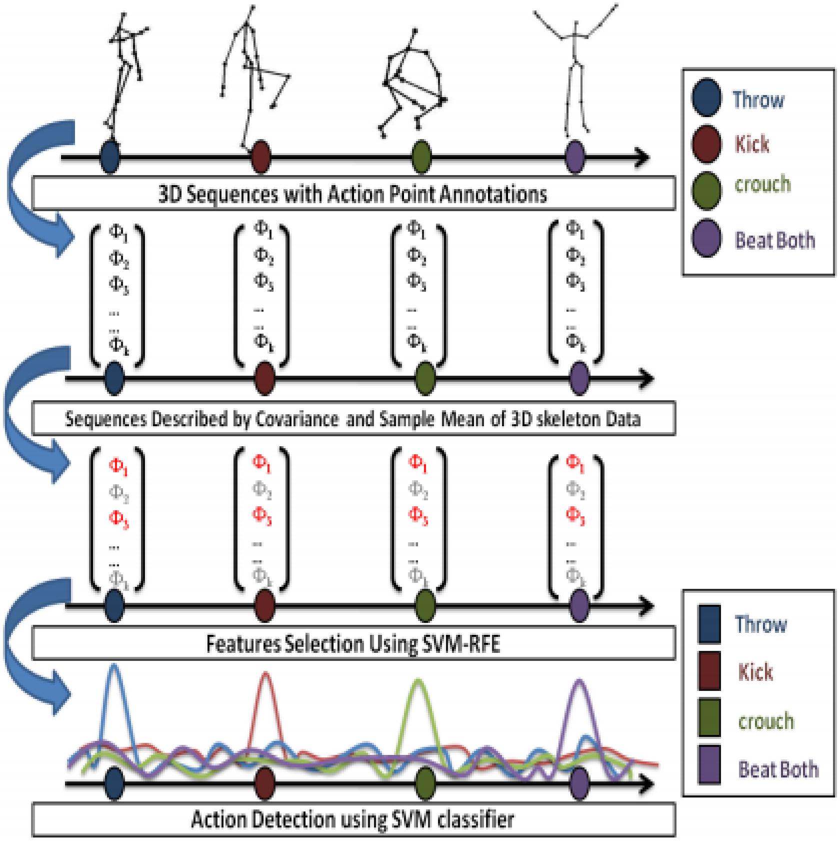

Throw
Kick
crouch
Beat Both
3D Sequences with Action Point Annotations
Φ_1
Φ_2
Φ_3
Φ_k
Sequences Described by Covariance and Sample Mean of 3D skeleton Data
Features Selection Using SVM-RFE
Action Detection using SVM classifier

FIGURE 2.20: Overview of the approach proposed in [78] based on covariance descriptors

method is illustrated in Figure 2.21.

In [98], a hierarchical dynamic framework is introduced. In their paper, they proposed a Hidden Markov Model (HMM) and estimated the emission probability based on Deep Neural Networks instead of Gaussian Mixture Model (GMM), which is more often used. As shown in Figure 2.22, the inclusion of an Ergodic State (ES) allows the detection of action transitions.

RGB-D based online and fast action recognition are still open issues. However, the constant need of real-world application improvements have motivated researchers to continue to study this challenge.

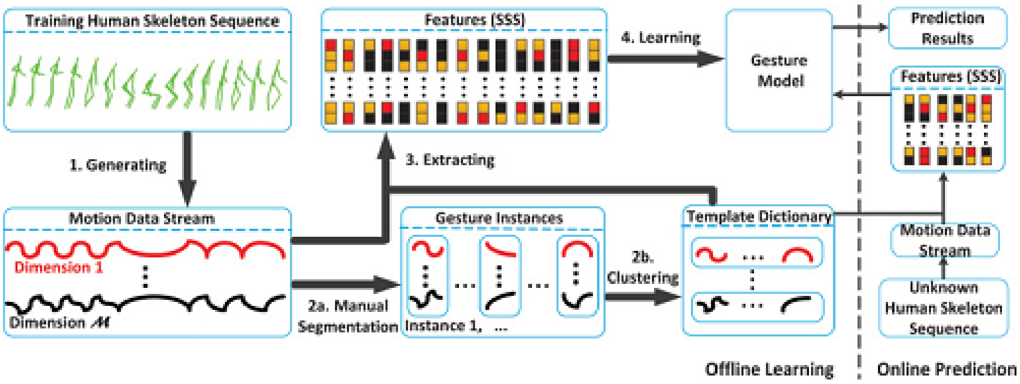


FIGURE 2.21: Overview of the method of [106] based on Structured Streaming Skeleton (SSS)

2.5 Conclusion

As presented in this chapter, since RGB-based methods present some limitations, the emergence of RGB-D cameras has renewed the interest of researchers in the task of action recognition. Using the two novel available modalities provided by RGB-D descriptors (depth maps and skeleton sequences), a huge number of novel descriptors has been proposed. As mentioned before, the classification of these human action descriptors can be done according to the nature of the used modality. In this way, RGB-D based descriptors have been separated into three groups: depth-based methods, skeleton-based methods and hybrid methods. Because hybrid methods are especially adapted to human-object interaction and are not in link with the topic of this thesis, we focus our interest on what we call simple descriptors which are depth-based descriptors and skeleton-based descriptors.

With the success of RGB-D based action recognition, some methods dealing with more complex scenarios have been initiated because of the importance of these latter in real-world applications. One of the most challenging scenario is the real-time applications requiring the reduction of latency. As presented in this chapter, these kind of applications could be divided into two different classes: online action recognition and fast action recognition which respectively aim to reduce the observational latency and the computational latency.

Over these five last years, online action recognition has increasingly attracted researchers despite the complexity of this task. Nevertheless, fast action recognition has been rarely mentioned. Indeed, the majority of offline methods such as [91, 63, 102] has only shown their efficiency in terms of accuracy and not in terms of computational latency. Furthermore, to the best of our knowledge, there is no work in the literature which proposes to compare depth-based methods and skeleton-based methods.

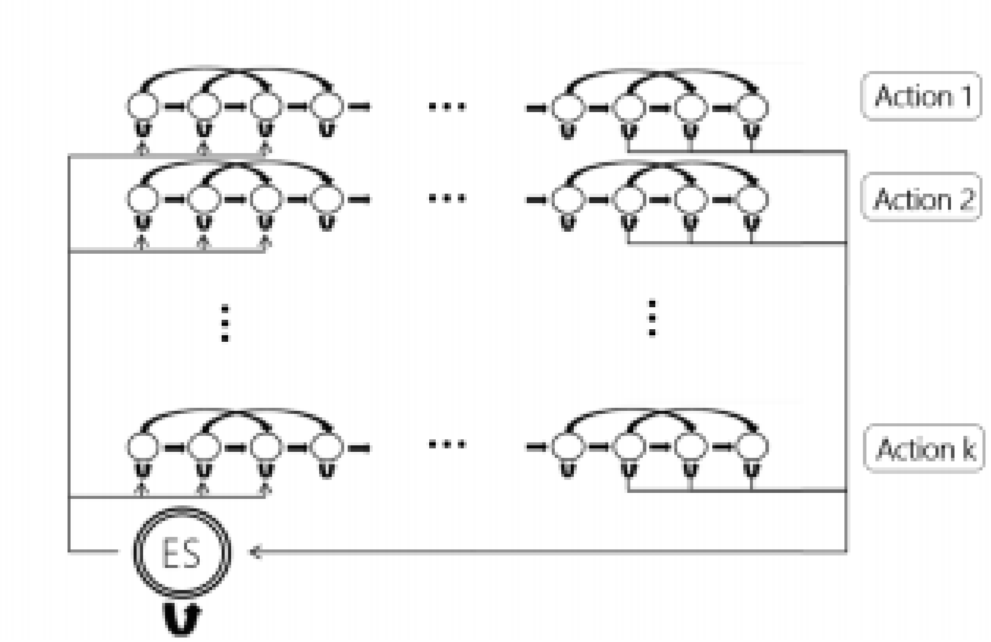


FIGURE 2.22: Overview of the ES-HMM architecture proposed in [98]

As a first step, we would study essentially fast action recognition. To do this and to orient our researches, the evaluation of RGB-D based methods is proposed in the next chapter. This evaluation aims at comparing the properties of each modality by analyzing different descriptors.

Chapter 3

Evaluation of RGB-D descriptors

3.1 Introduction

As presented in the previous chapter, different RGB-D-based approaches have been proposed in order to solve the issue of action recognition. The goal of this chapter is first to analyze with experimental results the differences between depth-based approaches and skeleton-based approaches and second to understand the weaknesses and the strengths of each method. This evaluation is a very important step in this thesis since it has oriented all our research works. We take into account only simple descriptors which are the depth approaches and the skeleton ones. In fact, this thesis focuses essentially on recognition of simple actions. As mentioned in Chapter 2, hybrid descriptors are especially useful and relevant for activity recognition. Moreover, simple descriptors can be fused with other descriptors if there is a need and this represents a different challenge. Classically, RGB-D based action recognition methods are evaluated in terms of accuracy of recognition on a specific dataset. The used dataset is mainly divided into two parts: one is used for training and the other part is devoted for testing. It exists different settings of experimentation such as leave-one-out, cross-splitting, etc. The accuracy is generally the only criterion of evaluation. To prove the effectiveness of a method, researchers generally compare its accuracy to the reported accuracy of earlier methods. However, there are two main drawbacks in the use of this protocol:

1) The experimental settings are sometimes different from a paper to another, making a fair comparison difficult. Padilla et al. [66] have outlined this issue of experimentation

on one of the most popular dataset for RGB-D based action recognition, the MSRAction3D dataset. For example, some papers use the data generated by subjects 1,2,3,4,5 for testing and the rest of data for training as in [65], while others calculate the training model using the actions performed by subjects 1,3,5,7,9 and the rest of data for testing [63]. Furthermore, some researchers divide the dataset into three subsets, while others use the whole dataset. More details about these protocols will be given in Section 3.7.

2) The criterion of accuracy is in most cases not sufficient to analyze the performance of a method. For various scenarios, the computational latency is very important. Indeed, even offline human action applications mainly need a fast answer from the action recognition system. Thus, if a method is very accurate, but needs a considerable execution time to work, its utility in real-world applications becomes very limited.

In this way, we propose to evaluate some available state-of-the-art methods as reliably as possible. Also, since we are trying to evaluate descriptors more than classification methods, we only test methods making use of a same classifier. Therefore, we select the linear SVM classifier, since it is the most used for RGB-D based action recognition. In the following, we will start by presenting an overview of the proposed evaluation protocol. Then, we will present the tested descriptors followed by a brief recall of the linear SVM algorithm. After that the chosen criteria of evaluation will be introduced. And lastly, we will present, the experiments and discuss about the results.

3.2 An overview of the proposed evaluation protocol

Figure 3.1 illustrates an overview of the proposed evaluation protocol. First of all, a specific dataset containing annotated segmented actions composed at least of the two modalities of interest (depth images and skeleton sequences) has to be chosen. In this way, according to the nature of the tested descriptor, skeleton sequences or depth sequences are used as input. Then, the data are divided into two groups: testing data and training data. The first group is used to train a classification model (training phase), while the other group is reserved to test the classification model (testing phase). For both phases, the descriptor computation has to be done. Since we are working with segmented videos, each sequence will return a descriptor. Therefore, the descriptors extracted from the training data are used to estimate a linear multi-class SVM classifier. Using this latter, the labels of the descriptors extracted from testing data are estimated. Finally, the evaluation criteria which are the recognition accuracy and the Mean Execution Time (MET) per descriptor are reported. As mentioned before, this protocol is repeated for all descriptors by exactly respecting the same experimental settings.

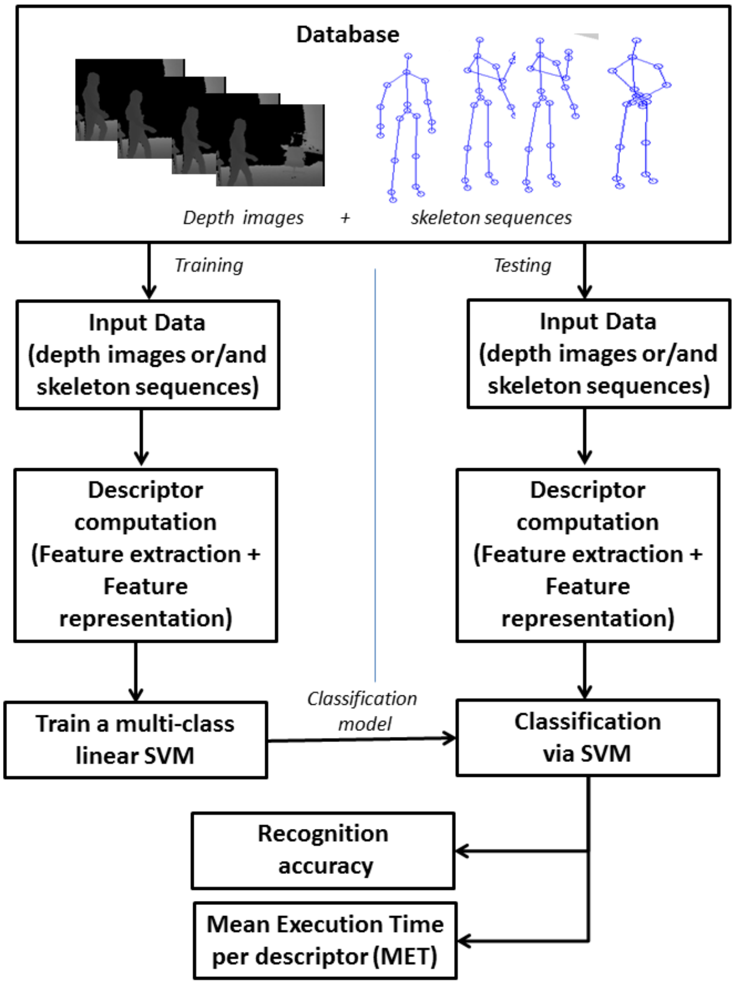

Database
Depth images + skeleton sequences
Training
Testing
Input Data (depth images or/and skeleton sequences)
Input Data (depth images or/and skeleton sequences)
Descriptor computation (Feature extraction + Feature representation)
Descriptor computation (Feature extraction + Feature representation)
Train a multi-class linear SVM
Classification model
Classification via SVM
Recognition accuracy
Mean Execution Time per descriptor (MET)

FIGURE 3.1: An overview of the proposed evaluation protocol

3.3 Tested descriptors

In this subsection, we present the tested RGB-D based descriptors. We download recent available codes of the state-of-the-art which make use of linear SVM for the classification (to realize a fair comparison of descriptors). We selected, according to the availability of the codes and the coherence with the topic of our study, three depth based methods and five skeleton-based methods. The depth-based approaches are HOG2 [63], HON4D [65] and SNV [102], while the skeleton-based approaches are proposed in [91] and gathers Joint Position (JP), Relative Joint Position (RJP), Joint Angles (JA) and Lie Algebra Relative Positions (LARP).

In what follows, we present briefly what present the different descriptors.

HOG2: At each instant, a histogram is calculated in each cell over a predefined number of bins. Finally, the histograms are combined in a whole vector that symbolizes the spatial feature vector. Then, the spatial descriptors are collected around the joints and histograms are calculated a second time by considering the time axis.

HON4D: The depth sequence is seen as a function $\mathbb{R}^3 \mapsto \mathbb{R}$ with $z = f(x, y, t)$. It can be viewed as a surface immersed in the space $\mathbb{R}^4$, defined by the points (x, y, z, t) such as $S(x, y, z, t) = f(x, y, t) - z = 0$. The normal to this surface is calculated as follows: $\vec{n} = \nabla S = (\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}, \frac{\partial z}{\partial t}, -1)$. Since only the orientation is relevant, normals are normalized. Then, the distribution of 4D normals is computed. To perform the quantization, polychrons which represent the 4D extension of polygons are used.

SNV: As described for HON4D, the normals are first computed. Then, to retain the neighboring information, they propose to define the notion of polynormals which represent a set of normals in a neighbourhood L. A sparse coding is finally used to code polynormals.

JP: As its name is indicating, this descriptor contains the information of the 3D absolute position on of skeleton joints.

RJP: This descriptor is computed using all the 3D vectors delimited by each couple of joints ans is thus called Relative Joint Position.

JA or Q: This descriptor contains the different quaternions corresponding to joint angles.

LARP: This descriptor has been described in the state-of-the-art chapter and represents the comination of transformations matrices $SE(3)$ defined foe each couple of adjacent segments.

All the representations proposed in [91] are reparametrized thanks to Dynamic Time Warping (DTW) algorithm.

3.4 Classification via Linear Support Vector Machine

In the previous chapter, it can be noted the important use of Support Vector Machines (SVM) as classifier for RGB-D based action recognition methods. More details about this method can be found in Appendix A.

For the descriptor evaluation, we propose to use a One-against-all approach proposed by Crammer and Singer [20]. As the classifier has been presented, the next section is dedicated to the specification of evaluation criteria.

3.5 Criteria of evaluation

As mentioned previously, the accuracy is not sufficient to evaluate an action recognition method method since the latency aspect is also important. In this way, instead of using only the accuracy as an evaluation criterion, we propose to use a second criterion that we call Mean Execution Time (MET) per descriptor. In this section, we present these two criteria by specifying how both are calculated. Let $\mathbf{B} = (\mathbf{I}_k, l_k)_{1 \leq k \leq N_{dataset}}$ be a dataset composed of $N_{dataset}$ instances containing each one a segmented human action, with $\mathbf{I}_k$ the k^{th} instance and l_k its label.

3.5.1 Accuracy of recognition

Let us suppose that B contains N_{action} different kind of actions. The dataset is divided into two parts, one part for training $\mathbf{B}_{train} = (\mathbf{I}_k^{train}, l_k)_{1 \leq k \leq N_{train}}$ and the rest for testing $\mathbf{B}_{test} = (\mathbf{I}_k^{test}, l_k)_{1 \leq k \leq N_{test}}$, with N_{train} the number of training instances and N_{test} the number of testing instances. With the use of training data $\mathbf{B}_{train}$, a classification model is learned allowing the prediction of the action label $l_k^{predicted}$ of an instance $\mathbf{I}_k$, such as $l_k^{predicted} \in [\![1, N_{action}]\!]$.

Thus, the accuracy is calculated as follows in Equation (3.1).

$$Accuracy = \frac{\sum_{i=1}^{N_{train}} \mathbb{1}_{l_k^{predicted} = l_k}}{N_{train}} \text{ with } \mathbb{1}_{l_k^{predicted} = l_k} = \begin{cases} 1 \text{ if } l_k^{predicted} = l_k \\ 0 \text{ else} \end{cases} \tag{3.1}$$

3.5.2 Mean execution time per descriptor

As we focus especially on human action descriptors, we propose to evaluate them in terms of latency. Let $\mathbf{D}_k$ be the human action descriptor extracted from the instance I_k and let $time_k$ be the execution time required for computing the descriptor $\mathbf{D}_k$. Therefore, the Mean Execution Time per Descriptor (MET) on the dataset $\mathbf{B}$ is calculated as described by Equation (3.2).

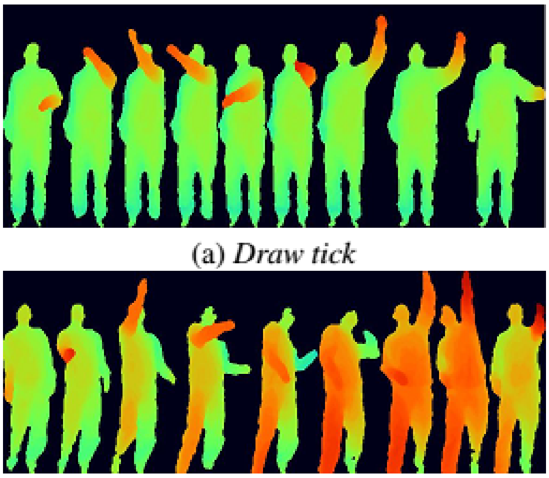
$$MET = \frac{1}{N_{dataset}} \sum_{k=1}^{N_{dataset}} time_k \tag{3.2}$$

3.6 RGB-D datasets for Action Recognition

In the literature, it exists a huge number of datasets aiming to evaluate the recognition of different human behaviour. Since we focus essentially on action recognition in this work, we choose only three datasets containing human actions (and not activities) to realize our experimentation and evaluate the descriptors. The chosen datasets are: MSRAction3D dataset, UTKinect dataset and Multiview3D dataset.

3.6.1 MSRAction3D dataset

MSRAction3D dataset [53] is one of the first dataset collected for RGB-D based action recognition. This well-known benchmark has been created by the University of Wollongong and Microsoft Research Redmond in 2010.



(a) *Draw tick*

(b) *Tennis serve*

FIGURE 3.2: Two examples of depth action sequences in MSRAction3D dataset [53]

This dataset contains 20 different actions: *high arm wave, horizontal arm wave, hammer, hand catch, forward punch, high throw, draw x, draw tick, draw circle, hand clap, two hand wave, side-boxing, bend, forward kick, side kick, jogging, tennis swing, tennis serve, golf swing, pickup throw.* Every action is performed two or three times by ten different subjects. This dataset contains two modalities: depth maps and skeleton sequences. MSRAction3D is very challenging because it contains very similar actions such as *high arm wave* and *high throw*. Figure 3.2 illustrates an example of two depth sequences contained in MSRAction3D dataset.

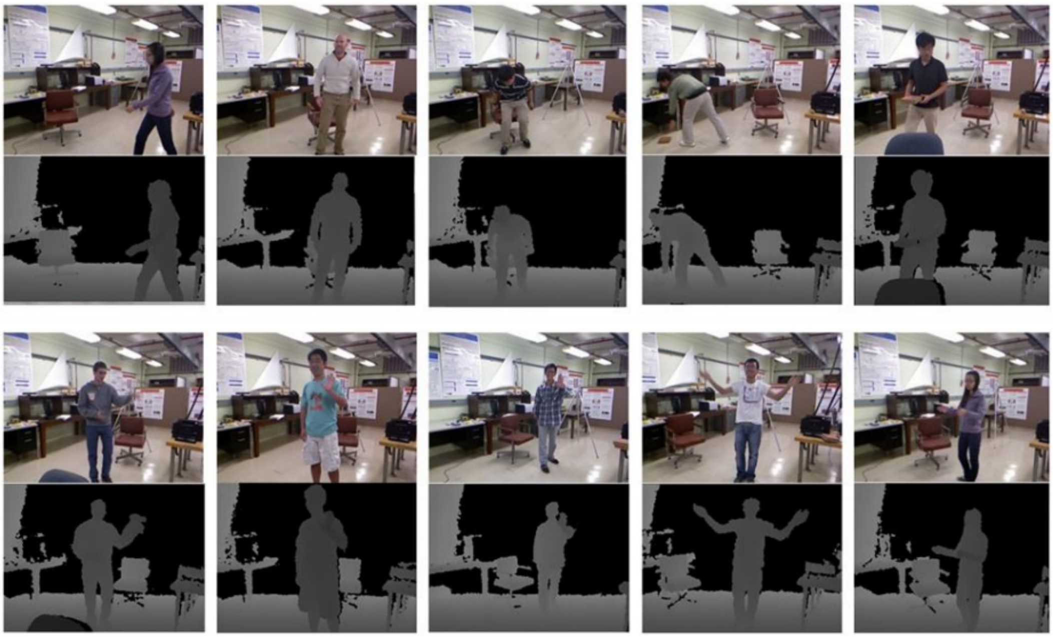


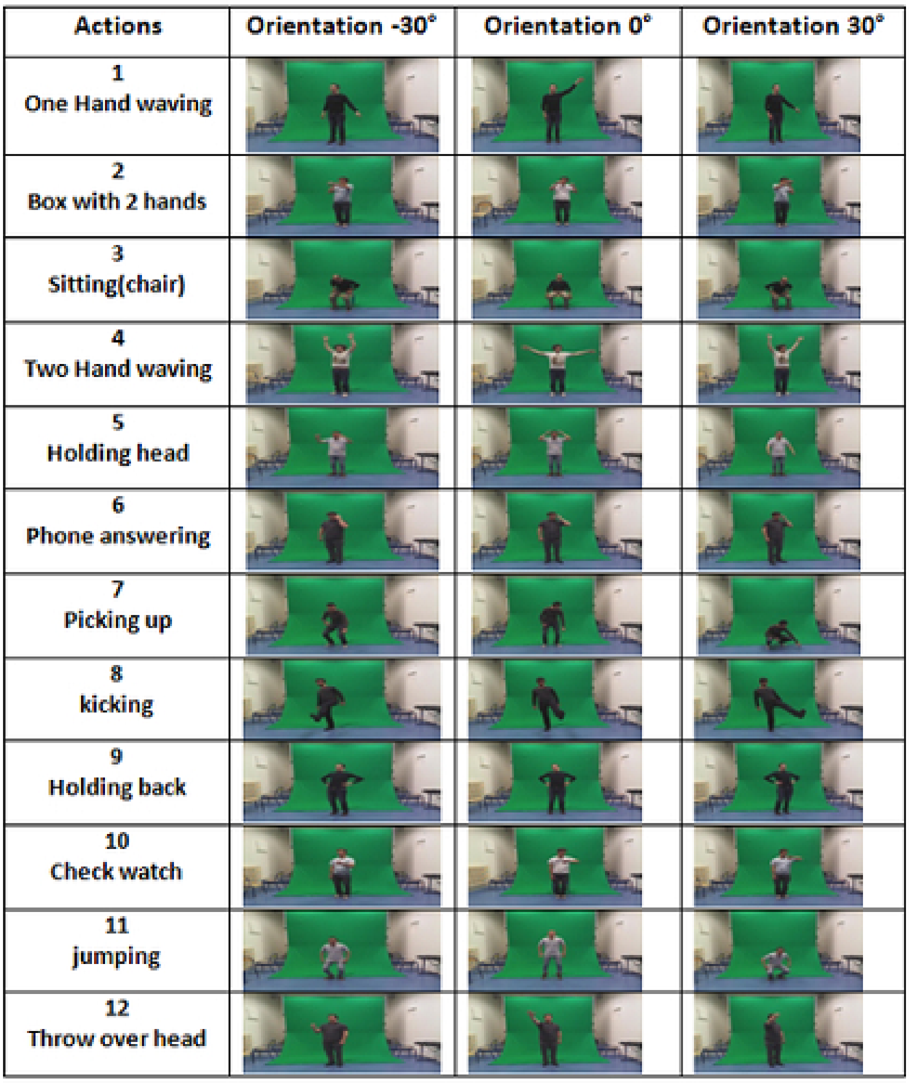
FIGURE 3.3: Examples of data in UTKinect Dataset [100]

3.6.2 UTKinect dataset

This dataset has been released by the University of Texas in 2010 [100]. UTKinect is composed of ten type of actions which are: *walk, sit down, stand up, pick up, carry, throw, push, pull, wave hands, clap hands*. Each action is performed twice by ten different subjects. RGB images, depth maps and skeleton sequences are available. The main issues in this dataset are its important subject inter-variability and its very short actions (with a very low number of frames). Figure 3.3 illustrates some images of UTKinect.

3.6.3 Multiview3D dataset

Multiview3D is a novel dataset introduced by the Engineering school of Mines Douai in 2016 [41]. This dataset contains 12 different actions: *One hand waving, box with two hands, sitting (Chair), two hand waving, holding head, phone answering, picking up, kicking, holding back, checking watch jumping, throwing over head*. Each action is performed by eight different subjects. Every subject performs every action twice with the respect of three different orientation angles (0°, 30°,-30°). These angle orientations are calculated based on the relative position of the camera. It contains three channels namely RGB images, depth images and skeleton sequences. It is easy to guess that the main challenge in this dataset is its viewpoint variation, in contrast to the two other ones, where it is assumed that the subject faces the camera. Figure 3.4 illustrates the structure of this dataset.



Actions	Orientation -30°	Orientation 0°	Orientation 30°
1 One Hand waving			
2 Box with 2 hands			
3 Sitting(chair)			
4 Two Hand waving			
5 Holding head			
6 Phone answering			
7 Picking up			
8 kicking			
9 Holding back			
10 Check watch			
11 jumping			
12 Throw over head			

FIGURE 3.4: Structure of the dataset Mines Douai Multiview3D dataset [41]

3.7 Experimental settings

The used experimental settings have to be carefully chosen and respected. As explained before, the choice of a unique protocol allows a fair comparison between different methods.

For MSRAction3D, we propose to follow the protocol proposed in [53], where the dataset is separated into three groups according to the action labels: AS1, AS2 and AS3. Every

AS1	AS2	AS3
horizontal arm wave	high arm wave	high throw
hammer	hand catch	forward kick
forward punch	draw x	side kick
high throw	draw tick	jogging
hand clap	draw circle	tennis serve
bend	Two arm wave	tennis swing
tennis serve	Forward kick	golf swing
Pick up and throw	side boxing	pick up and throw

TABLE 3.1: AS1, AS2 and AS3: the subsets of MSRAction3D [53]

subset contains 8 different actions. AS1 and AS2 contain simple actions that are very similar, while AS3 contains complex actions. Thus, the training and recognition are independently done on each subset. Table 3.1 presents these three groups. For the classification, a cross-splitting protocol is followed. As in [102], the data collected thanks to the subjects 1,3,5,7,9 are used for the training, and the rest of data are used for the testing.

For UTKinect dataset, a cross-splitting protocol is also followed where the actions performed by the half of the subjects are used for training and the other half is used for testing as in [91]. More precisely, the data generated by the subjects 1,3,5,7,9 are used for training.

For Multiview3D, we followed the experimental settings proposed in [41]. A cross-splitting is also done dividing the training and the testing samples. Then, the classification is done several times taking each time a specific orientation for the training and another specific orientation for the testing. This procedure allows us to analyze the effect of viewpoint changes on the accuracy of recognition. Finally, two values evaluating the accuracy are reported called Same View accuracy (SV) and Different View accuracy (DV). The first one represents the average accuracy value when the orientation of the training data and the orientation of the testing data is the same: it represents the easiest scenario. However, the second value reports the average accuracy when the testing data and the training data have different orientations.

All calculations were run on the same machine, a *Dell Inspiron N5010* laptop computer with *intel Core i5* processor, *Windows 7* operating system and *4GB RAM*.

Descriptor	AS1(%)	AS2(%)	AS3(%)	Overall(%)	MET (s)
HOG2 [63]	90.47	84.82	**98.20**	91.16	**6.44**
HON4D [65]	94.28	91.71	**98.20**	94.47	27.33
SNV [102]	**95.25**	**94.69**	96.43	**95.46**	146.57
JP [91]	82.86	68.75	83.73	78.44	**0.58**
RJP [91]	81.90	71.43	88.29	80.53	2.15
Q [91]	66.67	59.82	71.48	67.99	1.33
LARP [91]	83.81	84.82	92.73	**87.14**	17.61

TABLE 3.2: Accuracy of recognition and MET per descriptor on MSRAction3D: AS1, AS2 and AS3 represents the three groups proposed in the experimentation protocol of [53]

3.8 Results and discussion

In this section, the first part is devoted to the presentation of results, while a discussion about these results is presented in the second part.

3.8.1 Results

Table 3.2, Table 3.3 and Table 3.4 present respectively the obtained accuracy as well as the MET per descriptor on MSRAction3D, UTKinect and Multiview3D datasets. To realize a fair comparison, a penalty should be added to the MET of skeleton-based methods. Indeed, these methods needs an additional execution time for skeleton extraction. Generally, this process lasts approximately 45ms per frame knowing that actions are generally contained in a window composed of 12 to 50 frames in the chosen datasets. Thus, a fair comparison of execution time would have consisted in adding a penalty of 250 ms for skeleton method.

Results on MSRAction3D. On MSRAction3D dataset, depth-based descriptors including HOG2, HON4D and SNV are the most accurate with respectively 91.16%, 94.47%, 95.46% of recognition accuracy. On the other hand, skeleton-based descriptors are globally faster to compute with an MET of 0.58, 1.91 and 7.33 per descriptor for JP, RJP and Q, even with adding a penalty. However, LARP which is the most accurate skeleton-based method presents a relatively high MET (17.62 s per descriptor). This is maybe due to the complexity of the approach employed to calculate LARP (interpolation in the Lie algebra of $SE(3)$).

Results on UTKinect. On UTKinect dataset, skeleton-based descriptors including JP, LARP and RJP are the most accurate with respectively 100%, 97.08%, 97.98% of recognition accuracy. Furthermore, skeleton-based descriptors are globally faster to compute with an MET of

Descriptor	Overall(%)	MET (s)
HOG2 [63]	74.15	**5.03**
HON4D [65]	**90.92**	25.39
SNV [102]	79.80	1365.33
JP [91]	**100**	**0.43**
RJP [91]	97.98	1.91
Q [91]	88.89	1.40
LARP [91]	97.08	42.00

TABLE 3.3: Accuracy of recognition and MET per descriptor on UTKinect dataset

0.43, 1.91 and 1.40 per descriptor for JP, RJP and Q, even with adding a penalty. However, LARP (as for MSRAction dataset) presents a relatively high MET (42.00 s per descriptor). On this dataset, depth-based descriptors present a very low accuracy compared to skeleton-based descriptors with 74.15% for HOG2, 79.80% for SNV and 90.92% for HON4D. This could be explained by the fact that UTKinect contains some videos with a very small number of frames (less than 9 frames). Furthermore, depth-based descriptors need an important execution time (with an MET of 9.06s for HOG2 , 17.51s for HON4D and 1365.33s for SNV per descriptor).

Results on Multiview3D dataset. On Multiview3D dataset, skeleton-based descriptors including JP, RJP and LARP present the best score of accuracy for both tests SV and DV with respectively 96.00(SV)%-88.10%(DV), 97.70(SV)%-92.70%(DV) and 96.00(SV)%-88.10%(DV) of recognition accuracy. At the same time, they also register a better performance in terms of computational latency, (except for LARP) with respectively an MET of 1.22s, 4.58s and 10.51s. Depth-based descriptors are globally less accurate and less fast to compute with an accuracy of 87.80(SV)%-74.20%(DV) for HOG2, 89.30(SV)%-76.60%(DV) for HON4D and 94.27(SV)%-76.65%(DV) for SNV, and with respectively an MET of 9.06s, 17.51s and 271.72s per descriptor.

3.8.2 Discussion

The goal of this section is to analyze the obtained results. We propose to analyze different axes such as the trade-off between computational latency and accuracy, the robustness to view-point variation, the robustness to dataset changing and finally the differences between depth-based and skeleton-based descriptors.

Descriptor	SV(%)	DV(%)	MET (s)
HOG2 [63]	87.8	74.2	9.06
HON4D [65]	89.3	76.6	17.51
SNV [102]	94.27	76.65	271.72
JP [91]	96.00	88.10	1.22
RJP [91]	97.70	92.7	4.58
Q [91]	91.30	72.10	2.52
LARP [91]	96.00	88.1	10.51

TABLE 3.4: Accuracy of recognition and MET per descriptor on Multiview3D dataset

Trade-off between computational latency and accuracy. As mentioned before, the accuracy is an important criterion. However, it cannot be the only one in many real-world applications. In this part, we propose to evaluate descriptors at the same moment in terms of computational latency and recognition accuracy. To illustrate simultaneously the information of computational latency and the accuracy, we propose a novel kind of graphic where we can read the accuracy provided by a descriptor as well as its MET on a specific dataset. The idea of this illustration is to represent every descriptor with a ball where the surface area of the ball represents the MET of this descriptor and the center of the ball returns its accuracy. We can see this kind of graph realized for MSRAction3D dataset and UTKinect dataset in respectively Figure 3.5 and Figure 3.6. On MSRAction3D, the best descriptor in terms of accuracy and latency depends widely from the requirements of the application. In fact, it is not easy to classify them, since depth-based descriptors give better results in terms of accuracy, while skeleton-based descriptors (except LARP) give more interesting MET. However, we can easily see that for example SNV gives a relatively very bad trade-off between accuracy and latency despite the fact that it gives the best accuracy on MSRAction3D dataset. On UTKinect, it is easier to see that the performances of skeleton-based descriptors are higher. Globally, if we take into consideration the results obtained on the three datasets, we can tell that skeleton-based descriptors present generally a better trade-off between accuracy and computational latency. As we said before, the only skeleton-based descriptor that is not giving low computational latency is LARP, since the used mathematical tools lead to a huge number of approximations and calculations.

Robustness to view-point variation. As explained before, Multiview3D dataset has been designed to study the effect of viewpoint variation. Therefore, it has been shown thanks to our experimentation that JP, RJP and LARP gives the best results in terms of accuracy for both tests SV and DV, while depth-based descriptors gives the lowest accuracies. To analyze

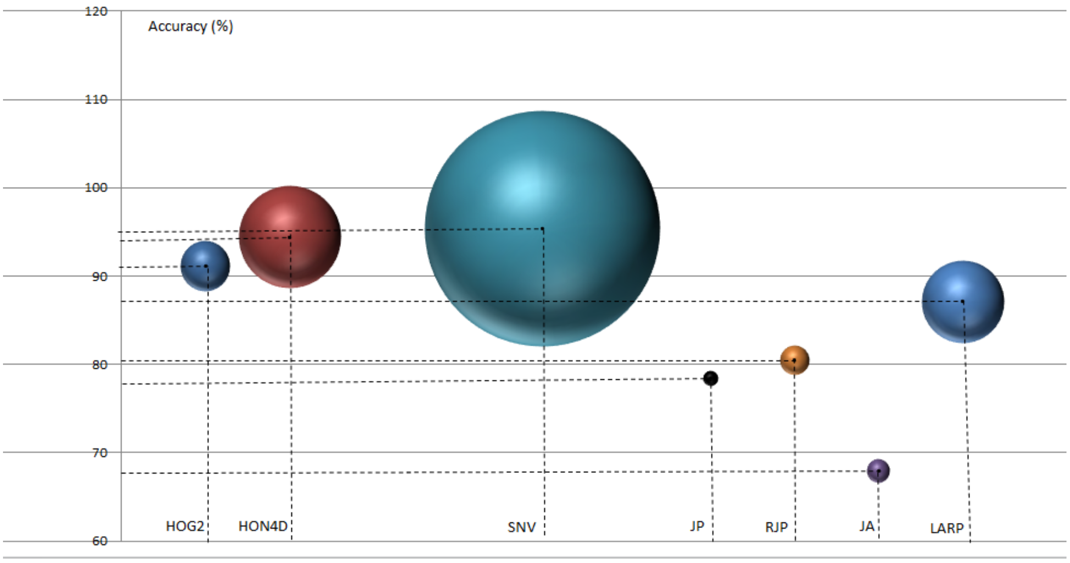


FIGURE 3.5: Illustration of the accuracy and the MET of different descriptors on MSRAction3D dataset: The center of the ball represents the accuracy of the correspoding method, while the surface of the occupied circle represents the MET per descriptor of each method.

the effect of the view-point variation, we propose to calculate the difference between the accuracies obtained for the tests SV and DV. The more important the difference is, the more the descriptor is sensitive to viewpoint variation. Figure 3.7 illustrates this difference for each descriptor. It can be seen that the descriptor Q is the most sensitive to viewpoint variation. However, generally, it can be noted that skeleton-based descriptors are less sensitive to viewpoint variation. Indeed, all of them except Q present a difference between the accuracies of SV and DV inferior to 10 %, unlike depth-based descriptors.

Robustness to dataset changing. Some descriptors are very accurate in some datasets, while they are not in others. For this reason, based on the given results, we propose to study the robustness of the different descriptors to dataset changing. To do that, we focus on the recognition accuracy values on MSRAction3D dataset, UTKinect Dataset and on Multiview3D datset. However, for Multiview3D dataset, only the test SV is taking into account, since the results of the DV test is mainly affected by the viewpoint variation. Figure 3.8 shows the recognition accuracy obtained on every dataset for each descriptor, while Figure 3.9 illustrates the amount of standard deviation of the accuracy of each descriptor. Indeed, the higher the standard deviation is, the more the accuracy is not stable. In this way, we can see that HON4D and LARP are the most robust to the changing of dataset according to our experiments.

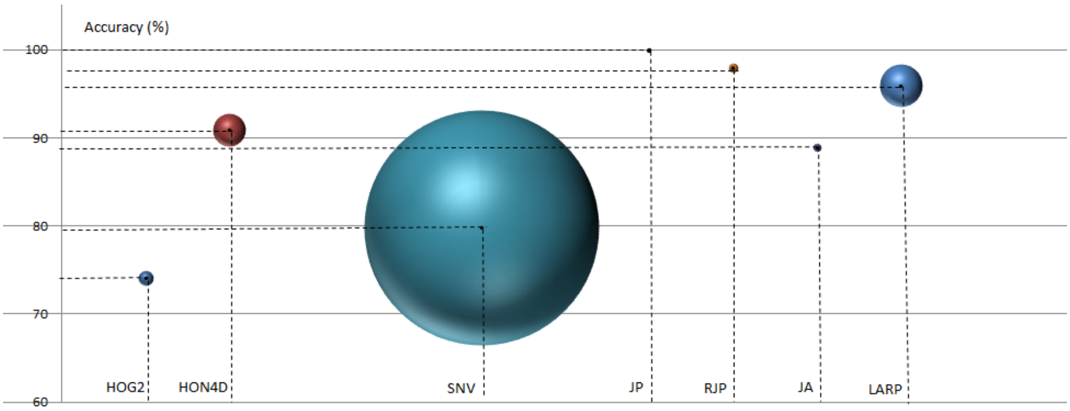


FIGURE 3.6: Illustration of the accuracy and the MET of different descriptors on UTKinect dataset

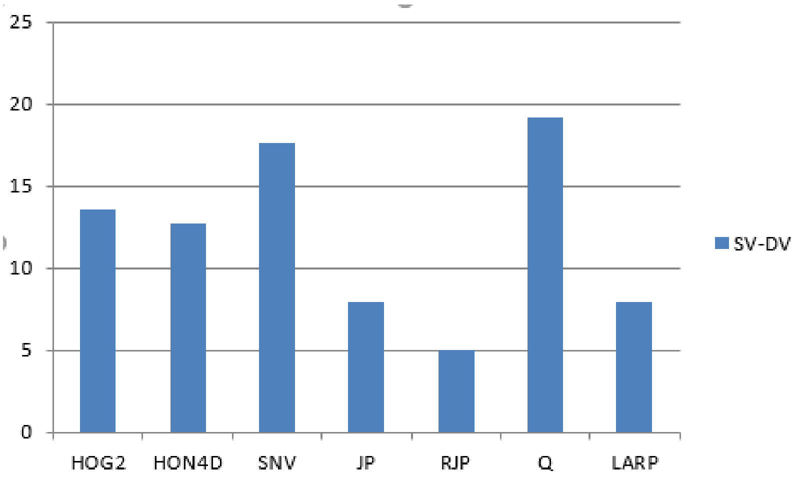


FIGURE 3.7: The difference between SV and DV accuracies for every descriptor

Depth-based descriptors vs skeleton-based descriptors. Thanks to the results of the experiments, it can be noted some differences between depth-based descriptors and skeleton-based descriptors. Table 3.5 reports different properties of descriptors by opposing depth-based descriptors to skeleton-based descriptors. As shown by the experiments, the accuracy depends widely from the tested dataset. For example, depth-based descriptors are generally more accurate on MSRAction3D dataset, while skeleton-based descriptors are globally more accurate on UTKinect and Multiview3D datsets.

However, experiments have shown the superiority of skeleton-based descriptors compared to depth-based descriptors in terms of computational latency. This fact could be explained by the less important dimension of skeleton-based descriptors, since the skeleton modality itself presents a lower dimension ($number\ of\ joints\ \times 3$) than the depth map

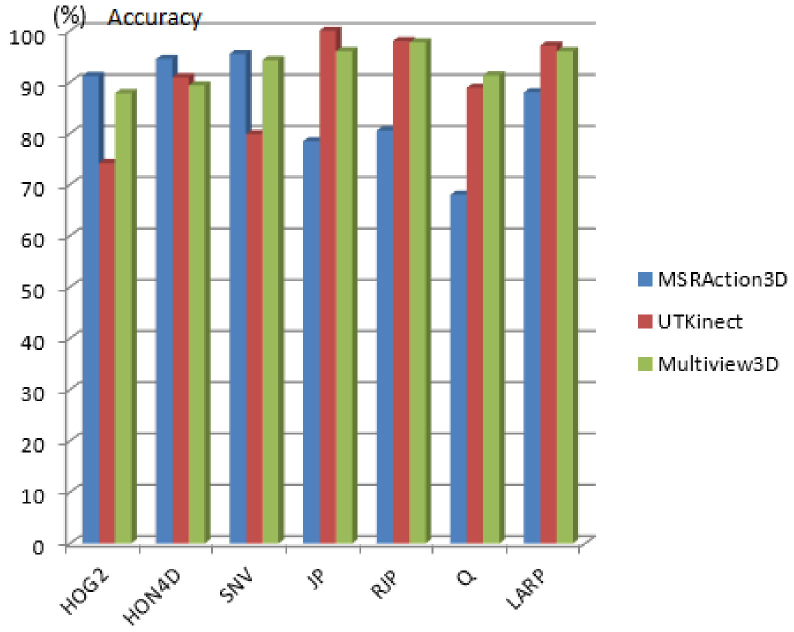


FIGURE 3.8: The accuracy on different datasets of each descriptor

descriptor	depth-based descriptors	skeleton-based descriptors
Accuracy (A)	nothing specific	
Descriptor Dimension	generally higher	generally lower
Computational Latency (CL)	generally higher	generally lower
Trade-off between CL and A	generally less good	generally better
Robustness to dataset changing	nothing specific	
Robustness to view-point variation	generally less robust	generally more robust

TABLE 3.5: Depth-based descriptors vs skeleton-based descriptors

(generally $240\ (lines) \times 320\ (columns)$).

In this way, we can conclude that skeleton-depth based descriptors present generally a better trade-off between computational latency and accuracy.

Finally, the experiments have also shown that the nature of descriptors does not influence the robustness to dataset changing. Nevertheless, skeleton-based descriptors have been proved to be more robust to view-point variation.

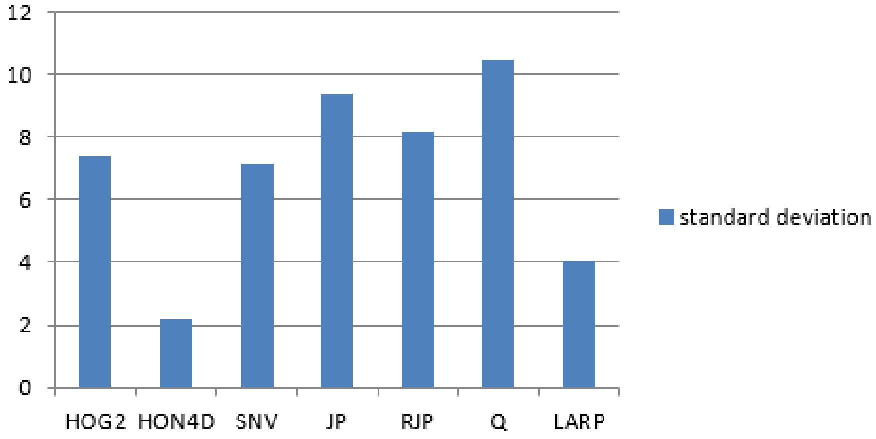


FIGURE 3.9: The standard deviation of the accuracy of different descriptors

3.9 Conclusion

In this chapter, we have proposed an evaluation of existing RGB-D based descriptors. For this purpose, recent depth-based descriptors as well as, recent skeleton-based ones have been tested on three benchmarks containing simple actions. The results have shown the better properties of skeleton-based descriptors in terms of computational latency (even with adding a penalty to take the skeleton extraction into account) and robustness to view-point variation. Since we are interested by fast action recognition, we focus in the rest of this manuscript on the skeleton modality which is more adapted to our case. However, we have to be careful with the choice of the descriptor computation algorithms because the complexity of some algorithms could increase the computational latency even with the use of the skeleton modality (such as LARP descriptor). Moreover, even if skeleton-based descriptors are faster to compute, the tested ones still present a high computation latency (in terms of seconds). The goal of this thesis is to propose a novel descriptor which presents a lower computational latency (applicable in real-world scenarios) with an acceptable accuracy. In the next chapter, a novel fast and accurate skeleton-based descriptor for action recognition is introduced.

Chapter 4

Kinematic Spline Curves descriptor: a novel fast and accurate descriptor for RGB-D based action recognition

4.1 Introduction

In the last chapter, we have presented an evaluation of RGB-D based descriptors in terms of two main criteria which are accuracy and computational latency. This evaluation showed the suitability of skeleton-based descriptors for fast action recognition (in comparison to depth-based descriptors which require a more important execution time). It was also outlined that despite the high accuracy of the tested state-of-the-art skeleton-based descriptors, these latter remain slow to compute and hardly adaptable to fast real-world applications. In this way, we propose in this chapter a novel, fast and relatively accurate skeleton-based descriptor for fast action recognition that we call Kinematic Spline Curves (KSC). The main idea of this descriptor is based on a cubic spline interpolation of skeleton kinematic features (joint position, joint velocity and joint acceleration) and the uniform resampling of the obtained curves. This procedure allows the calculation of same-size descriptors independently from the video length which is variable from an instance to another. To ensure the rapidity of calculation as well as the accuracy of our descriptor, we propose to carefully select the chosen algorithms by prioritizing simple and efficient algorithms and by avoiding complex models which can lead to an important increase of computational latency. On the other hand, to ensure the accuracy of the descriptor, two main facts have to be taken into account: the anthropometric variability and the execution rate variability. The anthropometric variability is due to the different proportions of human bodies. This is caused by the intervariability of human skeletons. The execution rate variability is mainly due to a different distribution of motion over time. Indeed, each person executes a gesture in a particular way. Furthermore, even a same person, executes every time the same gesture differently. It can be noted the presence of an inter and intra-variability at the same time. To overcome anthropometric and execution rate variability, two processes are proposed: an extension of the skeleton normalization algorithm proposed in [105] as well as a novel temporal normalization called Time Variable Replacement (TVR).

Therefore, the main contributions in this chapter are the KSC descriptor itself but also the TVR algorithm. In the second section, we present a brief review of cubic spline interpolation algorithm which is used to interpolate the kinematic features. Then, the third section details the proposed KSC descriptor, as well as the different processes used to compute it. In the fourth section, the experimental evaluation show the effectiveness of our method.

4.2 Cubic spline interpolation: a review

As presented in Section 4.1 , a cubic spline interpolation will be used to interpolate kinematic features. We reject the idea of using approximation instead of interpolation because of the low number of frames (generally between 20 and 50 per action in well-known datasets) favoring an inaccurate approximation. This method is a well-known numerical method which connects coherently a set of N points defined by their coordinates $(\alpha_k, y_k)_{k \in [\![1,N]\!]}$ with $\alpha_1 < \alpha_2 < ... < \alpha_k < ... < \alpha_N$. In this section, we recall the cubic spline interpolation approach and explain also the reason of this choice.

4.2.1 Formulation of the cubic spline interpolation

The goal is to estimate the function f which is assumed to be a continuous piecewise function of $N-1$ cubic third degree polynomials f_k as described by Equation (4.1).

$$f(t) = (f_k(t)) \quad for \quad \alpha_k < t < \alpha_{k+1} \tag{4.1}$$

Each function f_k is defined on the slice $[\alpha_k, \alpha_k + 1]$ (4.2). In order to respect the continuity of f, each f_k is joined at α_k by f_{k-1} (the first derivative $y_k\prime = \frac{y_{k+1}-y_k}{\alpha_{k+1}-\alpha_k}$ and the second derivative $y_k{\prime\prime} = \frac{y_{k+1}{\prime} - y_k{\prime}}{\alpha_{k+1} - \alpha_k}$ of y_k are also assumed to be continuous). The coefficients a_k, b_k, c_k, d_k of each polynomial f_k are computed by following Equations (4.3).

$$f_k(\alpha) = a_k(\alpha - \alpha_k)^3 + b_k(\alpha - \alpha_k)^2 + c_k(\alpha - \alpha_k) + d_k \tag{4.2}$$

where

$$\begin{cases} a_k = \dfrac{1}{(\alpha_{k+1} - \alpha_k)^2}(-2\dfrac{y_{k+1} - y_k}{\alpha_{k+1} - \alpha_k} + y_k{\prime} + y_{k+1}{\prime}) \\ b_k = \dfrac{1}{\alpha_{k+1} - \alpha_k}(3\dfrac{y_{k+1} - y_k}{\alpha_{k+1} - \alpha_k} - 2y_k{\prime} - y_{k+1}{\prime}) \\ c_k = y_k{\prime} = \dfrac{y_{k+1} - y_k}{\alpha_{k+1} - \alpha_k} \\ d_k = y_k \end{cases} \tag{4.3}$$

We choose this kind of interpolation because of the particular behavior of third degree polynomials which present a maximum of one inflexion point. Therefore, this kind of curve is at the same time realistic (in our case) and does not present undesired oscillations. The next experimental section will deepen these facts. Furthermore, this algorithm presents low computational latency.

To carry out the cubic spline interpolation, a matlab toolbox implementing the algorithm proposed by [25] is directly used.

4.2.2 Experiments related to interpolation

In this part, we present small experiments to explain the choice of the cubic spline interpolation. These experiments aim just to give concrete examples of interpolation and are not proofs. The goal is just to visualize the different interpolation methods. However, we will confirm the suitability of cubic spline interpolation compared to the other classical interpolation methods presented in Section 4.4.2.

Thus, we propose to compare the different interpolation methods provided by the matlab toolbox [25] that we denote by:

- **spline:** it corresponds to the cubic spline interpolation presented in the last section.

- **linear:** it corresponds to a linear interpolation. Instead of estimating a cubic function between every two consecutive points, this method aims to calculate a linear function.

- **nearest:** it corresponds to a nearest neighbour interpolation. The idea is to estimate the intermediate points by attributing them the value of the closest point.

- **pchip:** it corresponds to a shape-preserving piecewise Hermite cubic spline interpolation. This kind of interpolation is based on the same idea of the cubic spline interpolation. However, the interpolated curve has to follow the monotony of the data (shape-preserving).

- **v5cubic:** it corresponds to the cubic spline interpolation implemented in the version 5 of matlab, which does not extrapolate data.

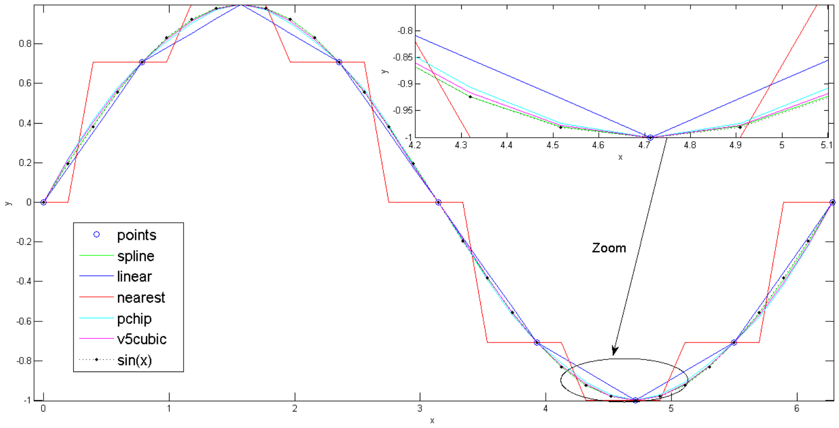


FIGURE 4.1: The interpolation of sampled points from the function $f(x) = sin(x)$ using the different methods

To compare these approaches, we use four known functions f, g, h and r with $f(x) = sin(x)$, $g(x) = cos(x)$, $h(x) = sinc(x)$ and $r(x) = arctan(x)$. We sample them, interpolate them, re-sample them with a smaller step and compare them to the real values. To evaluate the accuracy of the tested interpolation methods, the mean quadratic error is computed. Table 4.1 reports the calculated quadratic errors for each interpolation method using several usual functions. We specify that the initial number of samples taken for the interpolation is $s_i = 9$ and the number of samples recovered to calculate the error after the interpolation is $s_f = 33$. For a better visualization of different interpolation techniques, we propose the illustration of Figure 4.1 which shows the selected points, the interpolated functions as well as the real function $f(x)$.

It can be noticed that the nearest neighbour interpolation as well as the linear interpolation interpolation present very bad results. The shape of the obtained curves for these two algorithms is quite different from the original one, leading to an important quadratic error. The best results for these four functions are given by the cubic spline interpolation denoted by "spline".

Spline vs pchip. The two algorithms are very similar. Nevertheless, the fact that "pchip" is preserving the monotony of points can lead to non continuous second derivatives, contrary

function/type of interpolation	spline	linear	nearest	pchip	v5cubic
$f(x)$	**0.017**	0.2218	0.9573	0.093	0.0547
$g(x)$	**0.0153**	0.2218	0.9573	0.0811	0.0461
$h(x)$	**0.6191**	1.0447	1.5774	1.0419	0.8292
$r(x)$	**0.0975**	0.2614	1.1563	0.1032	0.1372

TABLE 4.1: Mean quadratic error of the different interpolation methods of the functions $f(x)$, $g(x)$, $h(x)$ and $r(x)$

to "spline". However, the efficiency of these two algorithms depends on the nature of the data, the number of samples, etc.

It exists other interpolation algorithms such as polynomial interpolation for which the order of the piecewise functions are not fixed. However, despite its accuracy, this kind of approach is very expensive in terms of calculation.

The different interpolation approaches presented here will be tested in Section 4.4.2.

4.3 Kinematic Spline Curves (KSC)

In this section, we describe the different procedures allowing the calculation of the proposed human action descriptor KSC. Figure 4.2 illustrates these different processes. As specified in Section 4.1, skeleton normalization compensates for anthropometric variability. Kinematic Features (KF) are computed using the discrete information of normalized joint positions. To reduce the effect of execution rate variability, a change of variables is made (replacing time by another variable called Normalized Action Time (NAT)). This temporal normalization is called as specified earlier, Time Variable Replacement (TVR). This step is followed by a cubic spline interpolation in order to make these features continuous. Finally, a uniform sampling is done on the continuous Kinematic Features KF^c to build the final descriptor KSC. Each subsection below details one of these five steps.

4.3.1 Spatial Normalization (S.N.) via skeleton normalization

In this study, an action is represented by a skeleton sequence varying over time. At each instant t, the skeleton is represented by the pose $\mathbf{P}(t)$, which is composed of n joints with the knowledge of their 3D position $\mathrm{p_j}(t) = [x_j(t), y_j(t), z_j(t)]$ with $j \in [\![1, n]\!]$.

$$\mathbf{P}(t) = [\mathbf{p_1}(t), ..., \mathbf{p_j}(t), ..., \mathbf{p_n}(t)] \tag{4.4}$$

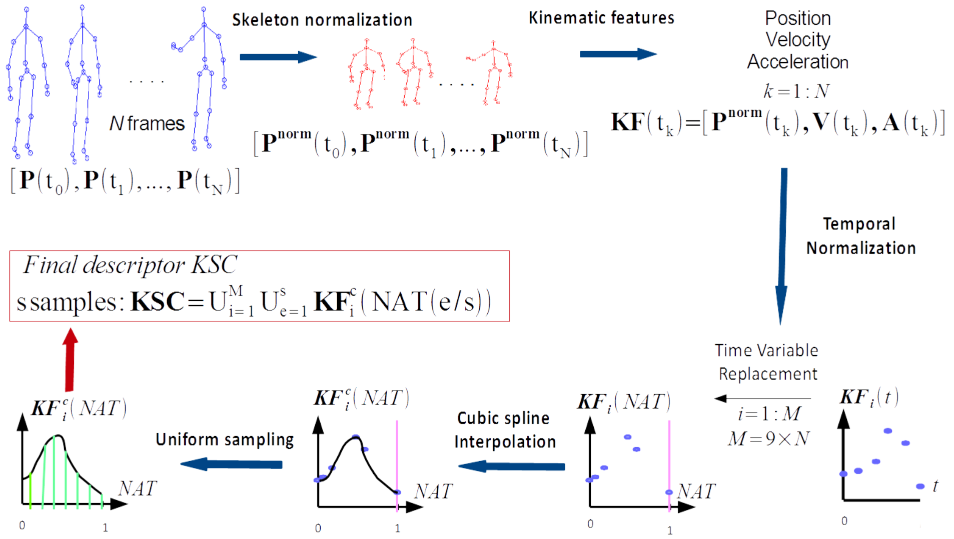

Skeleton normalization
Kinematic features
Position
Velocity
Acceleration
$k=1:N$
N frames
$[\mathbf{P}(t_0),\mathbf{P}(t_1),...,\mathbf{P}(t_N)]$
$[\mathbf{P}^{norm}(t_0),\mathbf{P}^{norm}(t_1),...,\mathbf{P}^{norm}(t_N)]$
$\mathbf{KF}(t_k)=[\mathbf{P}^{norm}(t_k),\mathbf{V}(t_k),\mathbf{A}(t_k)]$
Temporal
Normalization
Final descriptor KSC
s samples: $\mathbf{KSC}=\mathrm{U}_{i=1}^{M}\mathrm{U}_{e=1}^{s}\mathbf{KF}_i^c(\mathrm{NAT}(e/s))$
Time Variable
Replacement
$i=1:M$
$M=9\times N$
$KF_i(t)$
t
$KF_i(NAT)$
Cubic spline
Interpolation
$KF_i^c(NAT)$
Uniform sampling
$KF_i^c(NAT)$
NAT
0
1

FIGURE 4.2: An overview of our approach: this figure describes the different steps used to compute Kinematic Spline Curves (KSC). The first step represents the skeleton normalization which re-size the skeleton to make it invariant to anthropometric variability. Based on the information of normalized joint positions, the joint velocities and joint accelerations are computed. Therefore, the three values are concatenated to obtain KF. To palliate execution rate variability, t is replaced by the NAT variable by using a Time Variable Replacement Function (TVRF). This step represents a novel temporal normalization method, referred to as TVR. Then, each component KF is interpolated using a cubic spline algorithm in order to obtain M functions KF^c. Finally, The uniform sampling of KF^c allows us to build the final descriptor KSC.

Thus, a skeleton sequence, which represents a specific action (segmented videos), can be seen as a multidimensional time series. As in the majority of bio-mechanical studies, the initial position of human hip joint is assumed to be the origin. This is why we subtract the hip joint coordinates $\mathbf{p_{hip}}$ from each joint coordinates. Figure 4.3 gives an example of skeleton and indicates hip joint location.

$$\mathbf{P}(t) = [\mathbf{p}_1(t) - \mathbf{p}_{\mathrm{hip}}, ..., \mathbf{p}_{\mathrm{j}}(t) - \mathbf{p}_{\mathrm{hip}}, ..., \mathbf{p}_{\mathrm{n}}(t) - \mathbf{p}_{\mathrm{hip}}] \tag{4.5}$$

Although we consider that the hip joint is the absolute origin, an important spatial variability remains present, mainly due to anthropometric variability. To overcome this variability, we propose a skeleton normalization that is very similar to the normalization used in [105]. However, authors of this latter chose to learn an average skeleton for each dataset and then to constrain each skeleton of the dataset to have the same limb sizes. The problem is

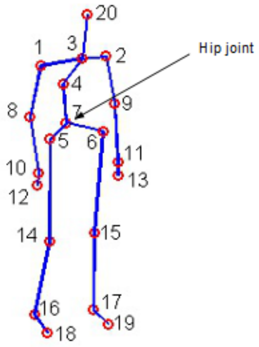


FIGURE 4.3: Illustration of the hip joint in the skeleton modality

that in real-world applications, the average skeleton of a specific dataset could be not representative (the average of a shape is not necessary a shape). Thus, we propose the euclidean normalization of each segment length without imposing a specific length. Hence, we obtain skeletons with unitary segments. Each skeleton is normalized successively starting with the root (hip joint) and moving gradually to the connected segments. This approach preserves the shape of the skeleton and its performance will be discussed in the Section 4.4. Algorithm 1 describes the skeleton normalization used. All joint positions are normalized except the hip joint position which is assumed to be the root and is therefore unchanged ($\mathbf{p}^{norm}_{hip} = \mathbf{p}_{hip}$). The experiments 4.4.5 show the good performance of this normalization comparing to the one based on the average skeleton.

We use the normalized skeleton sequence $\mathbf{P}^{\mathrm{norm}}$ obtained after applying Algorithm 1 to $\mathbf{p}^{\mathbf{hip}}$ to design our descriptor as depicted by Equation (4.6). Since all joints are interconnected, we obtain a normalized position for each joint i, ($\mathbf{p}^{\mathbf{norm}}_{\mathbf{a_i}}$, $\mathbf{p}^{\mathbf{norm}}_{\mathbf{b_i}}$ with $a_i, b_i \in [\![1, n]\!]$).

$$\mathbf{P}^{\mathrm{norm}} = [\mathbf{p}^{\mathrm{norm}}_{1}, \mathbf{p}^{\mathrm{norm}}_{2}, ..., \mathbf{p}^{\mathrm{norm}}_{\mathrm{n}}] \tag{4.6}$$

4.3.2 Kinematic Features (KF)

As in [105], we assume that human motion can be described by three important mechanical values: joint positions (normalized) $\mathbf{P}^{\mathrm{norm}}$, joint velocities $\mathbf{V}$ and finally joint accelerations $\mathbf{A}$. These values are concatenated to obtain what we call Kinematic Features (KF), denoted by $\mathbf{KF}$ in Equation (4.7).

$$\mathbf{KF}(t) = [\mathbf{P}^{\mathrm{norm}}(t), \mathbf{V}(t), \mathbf{A}(t))] \tag{4.7}$$

Algorithm 5: Skeleton normalization at an instant t

```
Input  : $(\mathbf{p}_{\mathbf{a_i}}(t), \mathbf{p}_{\mathbf{b_i}}(t))_{1\leq i\leq C}$ represents the segment extremities ordered and $C$
         represents the number of connections with $a_i$ the root extremity and $b_i$ the
         other extremity of the segment $i$
Output: $(\mathbf{p}^{\mathbf{norm}}_{\mathbf{a_i}}(\mathbf{t}), \mathbf{p}^{\mathbf{norm}}_{\mathbf{b_i}}(\mathbf{t}))_{1\leq i\leq C}$ with $a_i, b_i \in [\![1, n]\!]$
$\mathbf{p}^{\mathbf{norm}}_{\mathbf{a_1}}(t) := \mathbf{p}_{\mathbf{a_1}}(t)(\mathbf{p}_{\mathbf{a_1}}(t) = \mathbf{p}_{hip}(t)$ represents the position of the hip joint)
for $i \leftarrow 1$ to $C$ (C:Number of segments) do
    $\mathbf{S_i} := \mathbf{p}_{\mathbf{a_i}}(t) - \mathbf{p}_{\mathbf{b_i}}(t)$
    $\mathbf{s'_i} : = \frac{\mathbf{S}_i}{\|(\mathbf{p}_{\mathbf{a_i}}(t) - \mathbf{p}_{\mathbf{b_i}}(t))\|_2}$
    $\mathbf{p}^{\mathbf{norm}}_{\mathbf{b_i}}(t) := \mathbf{s'_i} + \mathbf{p}^{\mathbf{norm}}_{\mathbf{a_i}}(t)$
end
```

The velocity and the acceleration are known to be respectively the first and the second derivative of the position with respect to the time ($\mathbf{V}(t) = \frac{d\mathbf{P}^{\mathbf{norm}}(t)}{dt}$ and $\mathbf{A}(t) = \frac{d^2\mathbf{P}^{\mathbf{norm}}(t)}{dt}$).

Numerical derivative calculation Since we are using discrete data, we need to use a numerical derivative calculation method in order to calculate velocity and acceleration. We propose to use one of the most common derivative approximation called the central derivative approximation.

Let us suppose that we are looking for the derivative of $y(x)$ denoted by $y'(x)$. We assume that the function $y(x)$ can be developed in Taylor series considering the point x_m. The central derivative approximation is computed as depicted by Equation (4.8).

$$y'(x_m) = \frac{y(x_m + h) - y(x_m - h)}{2h} + O(h^2) \tag{4.8}$$

Numerical Kinematic features calculation In this context, the discrete data are uniformly separated. In this way, the term $2h$ is constant and can be simplified, giving Equation (4.9).

$$y'(x_m) = y(x_m + h) - y(x_m - h) + O(h^2) \tag{4.9}$$

In this part, we describe how velocity (4.10) and acceleration (4.11) are numerically computed from the discrete data of joint positions as in [105] by using the central approximation presented in Equation (4.8). In reality, the skeleton is composed of a skeleton sequence of N frames. $k \in [\![1, N]\!]$ denotes the frame index and t_k represents its associated instant.

$$\mathbf{V}(t_k) = \mathbf{P}^{\text{norm}}(t_{k+1}) - \mathbf{P}^{\text{norm}}(t_{k-1}) \tag{4.10}$$

$$\mathbf{A}(\mathbf{t_k}) = \mathbf{P}^{\text{norm}}(t_{k+2}) + \mathbf{P}^{\text{norm}}(t_{k-2}) - 2 \times \mathbf{P}^{\text{norm}}(t_k) \tag{4.11}$$

The dimension of the KF extracted from each frame is equal to $M = 9 \times n$. We recall that n is the number of joints. The velocity of the first and the last frames as well as the acceleration of the two first and the two last frames are considered null, since they could not be calculated.

4.3.3 Time Variable Replacement (TVR): a new approach for Temporal Normalization (TN)

Temporal variability is principally caused by execution rate variability (the temporal differences that exist when a subject repeats a same action or when another subject performs a same action), which implies changeable action duration and different distribution of motion. This means that actions are not performed in the same time slice as illustrated on the top panel of Figure 4.4. Consequently, recognizing actions with features varying in different time slices proves to be difficult. For this reason, Temporal Normalization (T.N.) is needed to improve the efficiency of our method. We propose to interpolate the components of the features, according to a novel variable called Normalized Action Time (NAT), instead of time. This change of variable is done thanks to a function TVRF which places the actions in a space that is invariant to execution rate variability as shown in Figure 4.4. Therefore, the KF can be expressed as a variable depending on NAT which is rate-invariant (4.12).

$$\mathbf{KF}(NAT) = [\mathbf{P}^{\text{norm}}(NAT), \mathbf{V}(NAT), \mathbf{A}(NAT)] \tag{4.12}$$

The choice of the function TVFR is crucial for the good functioning of the normalization.

1) First, this function should have a physical meaning which makes features invariant to the execution rate variability.

2) Second, the used function should be increasing and has to realize a one-to-one correspondence with time.

3) Finally, to compare actions with different temporal length, the latter function should vary in a fixed range independently from the length of the skeleton sequence as described by Figure 4.5.

In this way, let us consider that the $TVRF_{\mathscr{I}}$ ($\mathscr{I}$ represents the index of the action instance) as an increasing one-to-one correspondence between the variable interval of time and a fixed interval where NAT varies:

$$\forall \mathscr{S}, TVRF_{\mathscr{S}} : \left| \begin{array}{rcl} [0, N_{\mathscr{S}}] & \longrightarrow & [a, b] \\ t & \longmapsto & TVRF_{\mathscr{S}}(t) = NAT \end{array} \right.$$

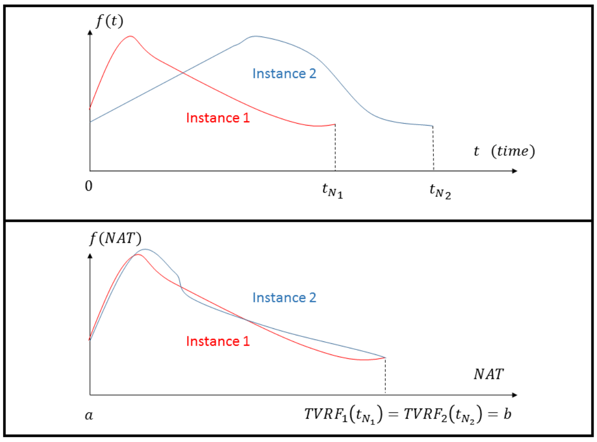


FIGURE 4.4: Example of the effect of temporal normalization on a joint component trajectory $f(t)$: In this example, $f(t)$ refers to the x-coordinates of a joint. We consider two instances of the same type of action (Instance1 and Instance2). Top: the trajectories are plotted as functions of time. We can observe that the time slices of the two trajectories are different ($t_{\mathrm{N1}} \neq t_{\mathrm{N2}}$). Bottom: After the substitution of time by an NAT variable, we can observe that the trajectories vary in the same range [a,b]. An important remark can also be made: the two trajectories representing the same action type are more similar.

a and b represent constant values such as $[a, b]$ is the range where NAT varies and $N_{\mathscr{S}}$ is the length of the skeleton sequence $\mathscr{S}$. We present hereafter two different TVRF respecting the mentioned conditions: the Normalized Accumulated kinetic Energy (NAE) of the skeleton and the Normalized Pose Motion Signal Energy (NPMSE).

Normalized Accumulated kinetic Energy of the skeleton (NAE) as a Time Replacement Variable Function (TVRF). The Normalized Accumulated kinetic Energy (NAE) of the skeleton is proposed as a Time Replacement Variable Function (TVRF). At an instant t and for each $\mathscr{S}$, it refers to the ratio between the kinetic energy $E_{\text{acc}}^{kinetic}(t)$ consumed by the human body until t and the total kinetic energy $E_{\text{total}}^{kinetic}$ consumed by the human body on the whole skeleton sequence composed of N frames. Equation (4.13) describes this new term where $E(t)$ represents the kinetic energy consumed by the human body at an instant t.

$$NAT = TVRF_{\mathscr{S}}(t) = NAE(t) = \frac{E_{\text{acc}}^{kinetic}(t)}{E_{\text{total}}^{kinetic}} \tag{4.13}$$

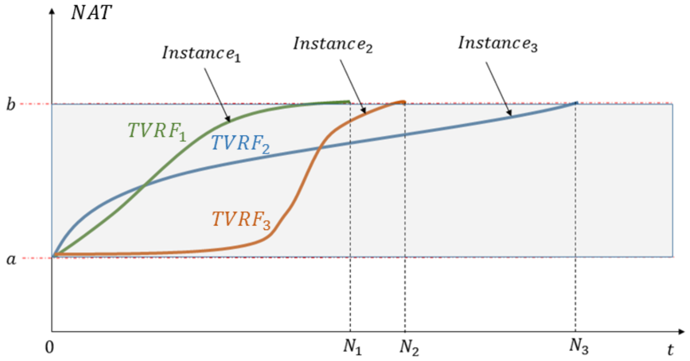


FIGURE 4.5: Example of TVRFs: Let $Instance_1$, $Instance_2$ and $Instance_3$ be three skeleton sequences with a temporal length respectively equal to N_1, N_2 and N_3. This figure shows how the TVR allows the expression of different instances with different temporal range in a same fixed range $[a, b]$

Since the data structure is discrete, for each punctual instant t_k, NAE is calculated as follows in Equation (4.14).

$$NAT = TVRF_{\mathscr{I}}(t_k) = NAE(t_k) = \frac{E^{kinetic}_{\text{acc}}(t_k)}{E^{kinetic}_{\text{total}}} = \frac{\sum_{c_1=1}^{t_k} E^{kinetic}(c_1)}{\sum_{c_2=1}^{N} E^{kinetic}(c_2)} \tag{4.14}$$

We recall that N represents the number of frames contained in the skeleton sequence.

Indeed, the NAE variable increases with the velocity of joints and consequently with the displacement quantity as well. If there is no motion, NAE does not increase. We use NAE for two essential reasons. First, all actions must be expressed in the same space which is guaranteed by the normalization of the energy (varying between 0 and 1). Secondly, to perform a coherent interpolation, a growing variable is necessary. This is why accumulation is used.

Kinetic Energy Calculation: Instead of considering the skeleton as a body, it can be assimilated to a set of n points, where each one represents a skeleton joint. In many earlier papers [64, 77, 80], the kinetic energy of the human skeleton $E^{kinetic}(t_k)$, at an instant t_k, is expressed by Equation (5.5), where n represents the number of joints, m_i and V_i, respectively, the mass and the velocity of the joint i. Since the skeleton joints are fictitious, we assume that they have a unitary mass ($m_i = 1, \forall i$).

$$E^{kinetic}(t_k) = \sum_{i=1}^{n} \frac{1}{2} m_\mathrm{i} V_\mathrm{i}^2(t_k) = \sum_{i=1}^{n} \frac{1}{2} V_\mathrm{i}^2(t_k) \tag{4.15}$$

$NAE(t_k)$, at an instant t_k is calculated, thanks to the kinetic energy term $E^{kinetic}(t_k)$ as depicted in Equation (4.13). After a change of variables, the KF can be expressed as a variable depending on NAE (see Equation (4.12)). Hence, Kinematic Features are expressed in the same range $[0, 1]$ and become therefore less sensitive to execution rate variability.

Normalized Pose Motion Signal Energy (NPMSE) as a Time Variable Replacement Function (TVRF). A second TVRF called Normalized Pose Motion Signal Energy (NPMSE) is proposed to replace the time variable.

Before defining the notion of Pose Motion, we need to define the rest pose. The rest pose represents the position of the spatially normalized skeleton when any gesture can be visually detected as in Figure 4.3.

To obtain this rest pose, a statistical model could be learned using the training data. For the moment, since we are still working on temporally segmented data and since we know that all actions in the used datasets start with the rest pose, we can use the first skeleton of the sequence to define the rest pose.

We define the Pose Motion Signal $s(t)$ as the distance d between the current pose $\mathbf{P^{norm}}(t)$ and the rest pose $\mathbf{P^{norm}}(rest)$ ($rest$ is the first pose in our experimental conditions), as described by Equation (4.16).

$$s(t) = d(\mathbf{P^{norm}}(t), \mathbf{P^{norm}}(rest)) \tag{4.16}$$

This used distance can be l_2, l_1, etc. More details about the nature of the distance will be given in the experimental section. Since the available data has a discrete nature, the energy of the discrete signal $s(t_k)$ at an instant t_k is calculated as follows in Equation (4.17). This term is called Pose Motion Signal Energy.

$$E^{signal}(t_k) = \sum_{i=1}^{k} |s(t_i)|^2 \tag{4.17}$$

It is important to specify that in this paragraph the energy calculated is the signal energy and not the kinetic one, which are two different notions.

The Pose Motion Signal Energy is a growing value because of the sum term in the energy calculation. However, to ensure that all the actions are expressed in the same range, a normalization of this term has to be done. For an instance $\mathscr{I}$, the TVRF is therefore the Normalized Pose Motion Signal Energy (NPMSE) which is calculated by dividing the Pose Motion Signal Energy at an instant t by the Pose Motion Signal Energy at the final instant of the sequence t_f as described by Equation (4.18).

$$NAT = TVRF_{\mathscr{I}}(t) = NPSME(t) = \frac{E^{signal}(t)}{E^{signal}(t_f)} \tag{4.18}$$

Since the time data are discrete, the NPMSE at an instant t_k is calculated as follows (4.19).

$$NAT = TVRF_{\mathscr{I}}(t_k) = NPMSE(t_k) = \frac{E^{signal}(t_k)}{E^{signal}(t_N)} = \frac{\sum_{i=1}^{k} |s(t_i)|^2}{\sum_{i=1}^{N} |s(t_i)|^2} \tag{4.19}$$

Relation with key poses: The idea of using the distance between the current pose and the rest pose comes from the fact that the more the poses are different from the rest state, the more they can discriminate actions. It is in a certain way related to the notion of key poses presented in many papers [60, 22, 72], where the key pose is defined as the extreme pose of an action. In [60], authors show that this kind of pose is sufficient to obtain interesting results. However, this kind of method does not include the spatio-temporal aspect which is also important. For this reason, we propose this TVRF function which exploits the notion of keypose without ignoring the spatio-temporal aspect.

A comparison between the two TVRFs is presented in Section 4.4.4. This method of normalization is simple to apply in offline applications needing a fast answer. However, in its actual form, it is hardly applicable in unsegmented scenario since the calculation of whole motion energy is needed for the normalization.

4.3.4 Cubic spline interpolation of Kinematic Features (KF)

Based on the continuous nature of human actions, we assume that the kinematic values (position, velocity, acceleration) also represent continuous functions of time. To switch from a numerical space to a continuous one, we propose to interpolate KF components depending on NAT as described in Section 4.3. For the interpolation, we use the cubic spline interpolation described in Section 4.2. Using the discrete information of each KF component, we obtain continuous functions $\mathbf{KF^c}$ depending on a chosen TVRF, as described in Equation (4.20), where $Spline$ refers to the cubic spline operator. We recall that M represents the dimension of $\mathbf{KF}$.

$$\mathbf{KF^c_i}(NAT) = \mathbf{KF^c_i}(TVRF_{\mathscr{I}}(t)) = Spline(\mathbf{KF_i}(TVRF_{\mathscr{I}}(t_k)))_{k\in[\![0,N]\!]}, \forall i \in [\![1, M]\!] \tag{4.20}$$

4.3.5 Uniform sampling

To build the final descriptor KSC of an instance $\mathscr{I}$, the obtained functions are uniformly sampled, choosing a fixed number of samples s. The choice of s will be discussed in Section 4.4.10. This process allows us not only to obtain same-size descriptors regardless of the sequence length but also to temporally normalize the actions and obtain values according to the same amount of NAT. The step used for the sampling is proportional to the sequence length in order to obtain a fixed size of $9 \times n \times s$ for all human motion descriptors. The final descriptor KSC used for the classification is depicted by Equation (4.21).

$$\mathbf{KSC} = \cup_{i=1..M} \cup_{e=0..s-1} \mathbf{KF^c_i}(NAT(a + \frac{e(b-a)}{s-1})) \tag{4.21}$$

Figure 4.6 illustrates the interest of uniform sampling after interpolation of curves expressed as functions of a TVRF. The idea is to maximize the Euclidean distance between two feature vectors when the represented actions are different and to maximize it when they represent the same action. We summarize the different processes that allow us to compute a KSC descriptor from a skeleton sequence in Algorithm 2.

4.3.6 Action recognition via linear Support Vector Machine

To perform action recognition, the KSC is used with a linear Support Vector Machine (SVM) classifier provided by the SVM library LibSVM [14]. The multi-class SVM proposed by [20] is carried out with a cost which is equal to 10. The main advantage of the linear kernel classifier is its low computational latency compared to non-linear kernel ones [31].

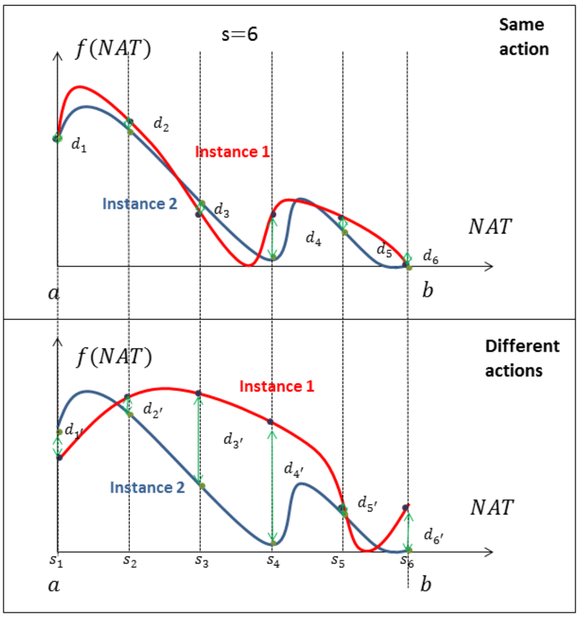


FIGURE 4.6: Uniform sampling Interest: In this figure, we propose the visualization of a component curve sampling, after the temporal normalization. We consider two instances of the same type of action on the top and two instances containing different actions on the bottom. These samples are used to design the final descriptor. The more similar the curves are , the smaller the distances d_i and $d_i^{'}$ between their samples are, for $i = 0...5$ (because the number of samples s is equal to 6 in this example). The classification will be based on this distance.

4.4 Experimental evaluation

In this section, we propose to evaluate the proposed method for action recognition on four benchmarks: MSRAction3D [53], UTKinect [100], Multiview3D Action [41] and MSRC12 [35] datasets. The first three datasets have already been presented in Chapter 3.

MSRC12 dataset: This dataset is generally used for skeleton-based action detection. It contains 594 sequences including 12 different types of gestures performed by 30 subjects. In total, there are 6244 annotated gesture instances. MSRC12 dataset only provides skeleton joints and action points which correspond to the beginning of an action. Despite the fact that MSRC12 dataset has been initially proposed for action detection, it could be very useful for the action recognition task because of its large volume.

For MSRAction, UTKinect and Multiview3D datasets, we follow the same experimental settings used in Chapter 3.

Algorithm 6: Computation of KSC from an instance $\mathscr{I}$

```
Input  : skeleton sequence $(\mathbf{P_j}(t_k))_{1 \le j \le n, 1 \le k \le N}$
Output: KSC
Normalize Skeleton $(\mathbf{P_j^{norm}}(t_k))_{1 \le j \le n, 1 \le k \le N}$ (4.6)
Compute Kinematic Features $\mathbf{KF_i}(t_k))_{1 \le i \le M}$ (4.12)
Compute $(TVRF_{\mathscr{I}}(t_k))_{1 \le k \le N}$ (4.13)
for $i \leftarrow 1$ to $M$ do
    Interpolation:
      $\mathbf{KF_i^c}(NAT) = \mathbf{KF_i^c}(TVRF_{\mathscr{I}}(t)) := Spline(\mathbf{KF_i}(TVRF_{\mathscr{I}}(t_k)))_{1 \le k \le N}$
end
Uniform sampling with $s$ the number of samples:
  $\mathbf{KSC} := \cup_{i=1..M} \cup_{e=0..s-1} \mathbf{KF_i^c}(NAT(a + \frac{(b-a)e}{s-1}))$
```

Descriptor	AS1(%)	AS2(%)	AS3(%)	Overall(%)	MET (s)
HOG2 [63]	90.47	84.82	**98.20**	91.16	**6.44**
HON4D [65]	94.28	91.71	**98.20**	94.47	27.33
SNV [102]	**95.25**	**94.69**	96.43	**95.46**	146.57
JP [91]	82.86	68.75	83.73	78.44	0.58
RJP [91]	81.90	71.43	88.29	80.53	2.15
Q [91]	66.67	59.82	71.48	67.99	1.33
LARP [91]	83.81	84.82	92.73	87.14	17.61
KSC-NAE (ours)	86.92	72.32	94.59	84.61	**0.092**
KSC-NPMSE-l_1 (ours)	85.05	85.71	96.4	89.05	**0.091**
KSC-NPMSE-l_∞(ours)	**85.71**	86.61	93.69	88.69	**0.091**
KSC-NPMSE-l_2(ours)	83.81	**87.50**	**97.30**	**89.54**	**0.092**

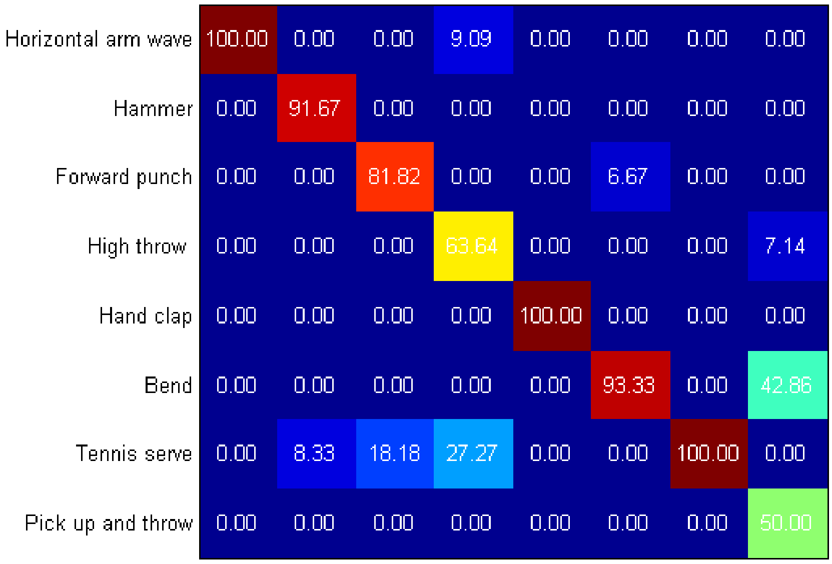
TABLE 4.2: Accuracy of recognition and MET per descriptor on MSRAction3D: AS1, AS2 and AS3 represent the three groups proposed in the experimentation protocol of [53].

For MSRC12 dataset, we follow the same protocol used in the paper of Hussein et al. [45] where the end points of actions have been added to the dataset. Thanks to these points, it became possible to benefit from the large amount of data in order to test an action recognition method without detecting actions. Thus, a cross-splitting is done where the actions performed by half of the subjects are used for training, while the rest of data is used for testing.

In the experiments, the two TVRF presented in this chapter are tested: the NAE and the NPMSE. When the KSC is calculated based on the temporal normalization using the NAE, our descriptor is denoted KSC-NAE. To calculate NPMSE, three kind of distances are used l_1 l_2, and l_∞. When the function NMPSE is employed (using the distances l_1, l_2 and l_∞) for the temporal normalization, our descriptor is respectively denoted by KSC-NMPSE-l_1, KSC-NMPSE-l_2 and KSC-NMPSE-l_∞ .

As in Chapter 3, the accuracy, as well as the MET are reported and compared to the state-of-the art algorithms.

Skeleton alignment for Multiview 3D dataset: Finally, since Multiview3D dataset contains skeletons with variable orientations, we propose to add in the pre-processing step, a simple skeleton alignment algorithm. We align the data as follows: we consider that for each action the first skeleton of the sequence is in the rest state. To do that, we assume that we work in a specific scenario where the actions are already segmented and where each first skeleton is in the rest state. We choose one of the first skeletons as a reference and we optimize the transformation matrix between the first pose of each sequence and the reference skeleton using a least square optimization. Thus, we apply the obtained transformation matrix to the rest of the sequence.



Horizontal arm wave	100.00	0.00	0.00	9.09	0.00	0.00	0.00	0.00
Hammer	0.00	91.67	0.00	0.00	0.00	0.00	0.00	0.00
Forward punch	0.00	0.00	81.82	0.00	0.00	6.67	0.00	0.00
High throw	0.00	0.00	0.00	63.64	0.00	0.00	0.00	7.14
Hand clap	0.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00
Bend	0.00	0.00	0.00	0.00	0.00	93.33	0.00	42.86
Tennis serve	0.00	8.33	18.18	27.27	0.00	0.00	100.00	0.00
Pick up and throw	0.00	0.00	0.00	0.00	0.00	0.00	0.00	50.00

FIGURE 4.7: Confusion matrix obtained using the descriptor KSC-NPMSE-l_2 on the group AS1 of the dataset MSRAction3D

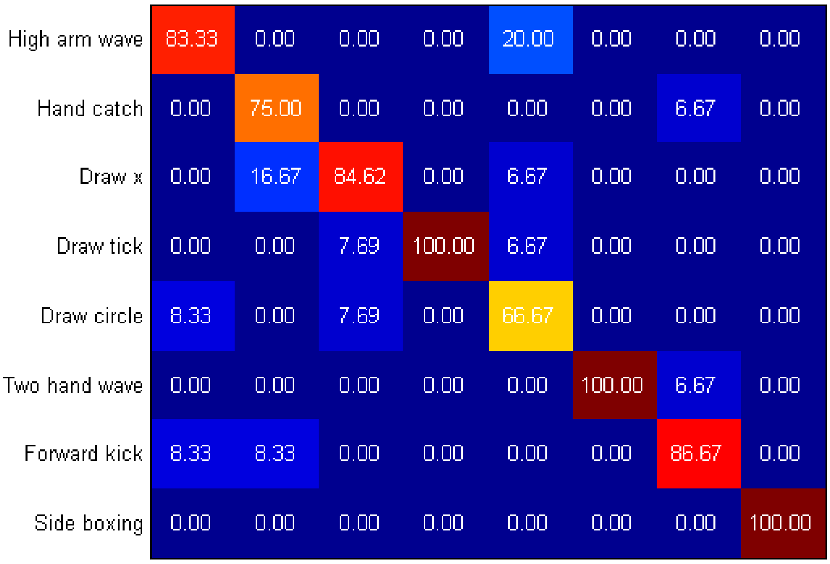
4.4.1 Low computational latency

Table 4.2, Table 4.4, Table 4.5 and Table 4.6 compare our approach with state-of-the-art methods in terms of computational latency by reporting the MET per descriptor in respectively MSRAction3D, UTKinect, MSRC12 and Multiview3D datasets. As mentioned in Chapter 3, it has been shown that the skeleton extraction process lasts approximately 45ms

Process	MSRAction3D	UTKinect	Multiview3D	MSRC12
Spatial Normalization	0.022	0.016	0.016	0.06
Descriptor computing	0.07	0.064	0.073	0.077
Classification	0.008	0.002	0.010	0.015
Total	0.1	0.082	0.099	0.152

TABLE 4.3: MET of each process of KSC-NPMSE-l_2 descriptor on the four benchmarks

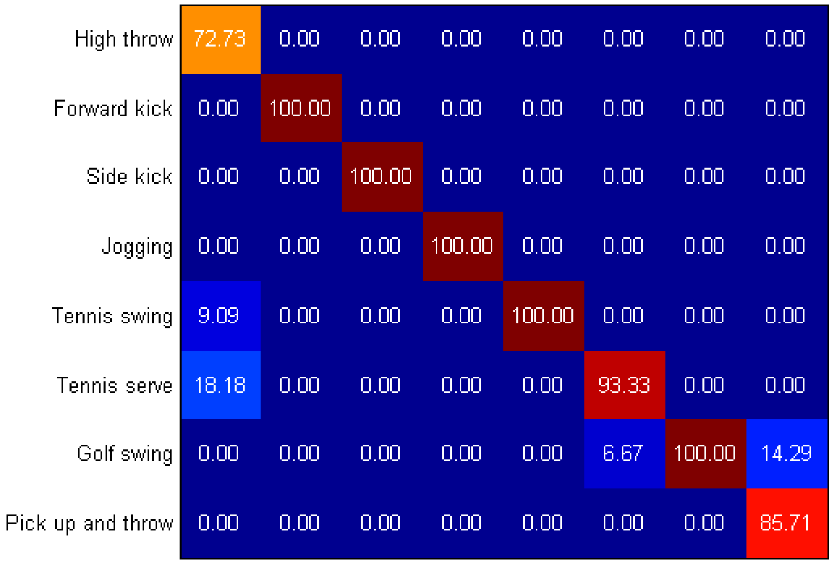
per frame knowing that actions are generally contained in a window composed of 12 to 50 frames. Thus, a fair comparison of execution time would have consisted in adding a penalty of 250 ms for skeleton methods. But, this would have no influence on the fact that our approach remains more suited to real-time applications, despite an additional extraction process. Indeed, we show that our descriptors KSC-NAE, KSC-NPMSE-l_1, KSC-NPMSE-l_2 and KSC-NPMSE-l_∞ are faster in terms of computational latency with a maximum of only 0.092s of MET on MSRAction3D dataset, 0.10 on UTKinect dataset, 0.137 on MSRC12 dataset and 0.092 on Multiview3D dataset.



High arm wave	83.33	0.00	0.00	0.00	20.00	0.00	0.00	0.00
Hand catch	0.00	75.00	0.00	0.00	0.00	0.00	6.67	0.00
Draw x	0.00	16.67	84.62	0.00	6.67	0.00	0.00	0.00
Draw tick	0.00	0.00	7.69	100.00	6.67	0.00	0.00	0.00
Draw circle	8.33	0.00	7.69	0.00	66.67	0.00	0.00	0.00
Two hand wave	0.00	0.00	0.00	0.00	0.00	100.00	6.67	0.00
Forward kick	8.33	8.33	0.00	0.00	0.00	0.00	86.67	0.00
Side boxing	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00

FIGURE 4.8: Confusion matrix obtained using the descriptor KSC-NPMSE-l_2 on the group AS2 of the dataset MSRAction3D

Also, Table 4.3, which presents the execution time per descriptor of each process on the four benchmarks, shows that the classification step can be neglected compared to the descriptor computation step in terms of computational latency. Thus, with these results,



High throw	72.73	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Forward kick	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
Side kick	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00
Jogging	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00
Tennis swing	9.09	0.00	0.00	0.00	100.00	0.00	0.00	0.00
Tennis serve	18.18	0.00	0.00	0.00	0.00	93.33	0.00	0.00
Golf swing	0.00	0.00	0.00	0.00	0.00	6.67	100.00	14.29
Pick up and throw	0.00	0.00	0.00	0.00	0.00	0.00	0.00	85.71

FIGURE 4.9: Confusion matrix obtained using the descriptor KSC-NPMSE-l_2 on the group AS3 of the dataset MSRAction3D

we can assume that the constraint of low computational latency is respected. However, it is important to notice that the computational latency also depends from a lot of parameters such as the image resolution (from the number of joints, in this case) as well as the number of classes (20 for MSRAction3D, 10 for UTKinect, 12 for MSRC12 and 12 for Multiview3D datasets). Our experiments aim just at comparing execution time of different techniques applied to a similar situation.

4.4.2 Good accuracy and Robustness

Computational latency is an important criterion of evaluation, but is not sufficient. In fact, the accuracy of recognition should be acceptable. Table 4.2, Table 4.4, Table 4.5 and Table 4.6 respectively report the accuracy of recognition of our method on MSRAction3D, UTKinect, MSRC12 and Multiview3D datasets and compare it to state-of-the-art methods.

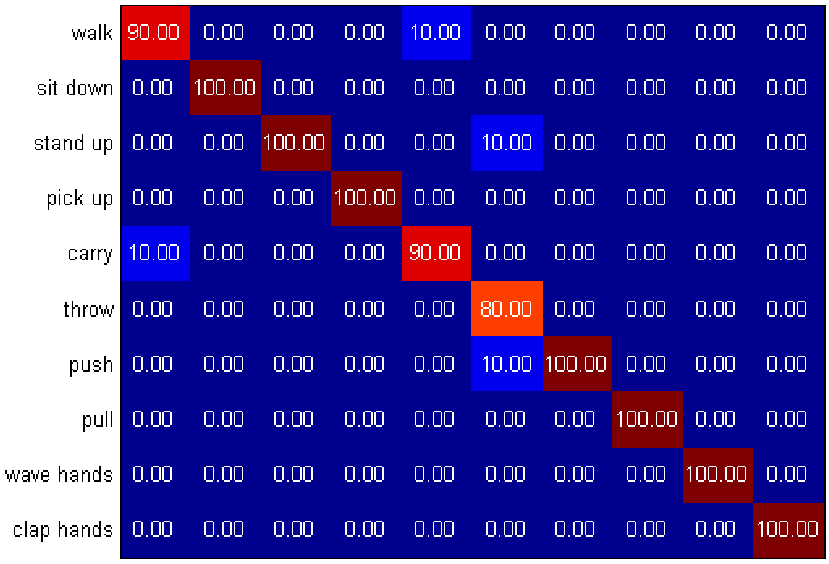
On MSRAction3D dataset, our method allows us to recognize 89.54% of the actions correctly using KSC-NPMSE-l_2 descriptor, which is a better score than the ones given by other skeleton representations. On MSRAction3D dataset, some depth-based descriptors give better results in terms of accuracy of recognition. However, their execution times per

Descriptor	Accuracy (%)	MET (s)
HOG2 [63]	74.15	**5.03**
HON4D [65]	**90.92**	25.39
SNV [102]	79.80	1365.33
JP [91]	**100**	0.43
RJP [91]	97.98	1.91
Q [91]	88.89	1.40
LARP [91]	97.08	42.00
Random Forrest [110]	87.90	-
KSC-NAE (ours)	84.00	**0.08**
KSC-NPMSE-l_1 (ours)	94.00	0.09
KSC-NPMSE-l_∞ (ours)	94.00	**0.08**
KSC-NPMSE-l_2 (ours)	**96.00**	0.10

TABLE 4.4: Accuracy of recognition and MET per descriptor on UTKinect dataset

Descriptor	Accuracy (%)	MET (s)
Logistic Regression [58]	91.2	-
Covariance descriptor [45]	91.7	-
KSC-NAE (ours)	84.00	0.137
KSC-NPMSE-l_1 (ours)	93.22	**0.134**
KSC-NPMSE-l_∞ (ours)	94.17	**0.132**
KSC-NPMSE-l_2 (ours)	**94.27**	0.134

TABLE 4.5: Accuracy of recognition and MET per descriptor on MSRC12 dataset



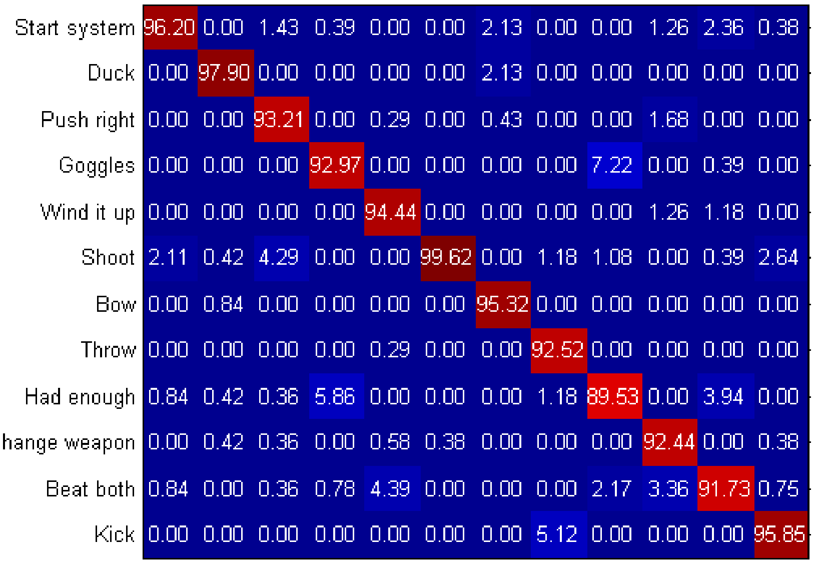
walk	90.00	0.00	0.00	0.00	10.00	0.00	0.00	0.00	0.00	0.00
sit down	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
stand up	0.00	0.00	100.00	0.00	0.00	10.00	0.00	0.00	0.00	0.00
pick up	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
carry	10.00	0.00	0.00	0.00	90.00	0.00	0.00	0.00	0.00	0.00
throw	0.00	0.00	0.00	0.00	0.00	80.00	0.00	0.00	0.00	0.00
push	0.00	0.00	0.00	0.00	0.00	10.00	100.00	0.00	0.00	0.00
pull	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00	0.00
wave hands	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00
clap hands	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00

FIGURE 4.10: Confusion matrix obtained using the descriptor KSC-NPMSE-l_2 on the dataset UTKinect

descriptor remain too high and seem to be unsuitable for real-time applications (6.44s for HOG2, 27.333s for HON4D, 146.57s for SNV against 0,092s for KSC).

On the other hand, some skeleton-based descriptors such as JP, RJP and LARP give slightly better accuracy on UTKinect and Multiview3D datasets for the Same View test (100% for JP, 97.98% for RJP and 97.08% for LARP against 96% KSC-NPMSE-l_2), but higher MET making them unsuitable for real-time applications (MET of 0.43s for JP, 1.91 for RJP, 42.00 for LARP against 0.10s for KSC-NPMSE-l_2). Furthermore, these descriptors are not robust to the dataset changing. For example, JP gives an accuracy of 78.44% on MSRAction3D, while it gives 96.00% on Multiview3D dataset for the Same View test. On MSRC12 dataset, KSC-NPMSE-l_2 descriptor outperforms the other representations in terms of accuracy with a score of 94.27% of good recognition.

Furthermore, Figure 4.7, Figure 4.8, Figure 4.9, Figure 4.10 and Figure 4.11, which respectively represent the confusion matrices obtained on MSRAction3D, UTKinect, and MSRC12 datasets show that the majority of actions are well recognized and that the confusion is especially present for very close actions such as *high throw* and *tennis serve* on the group AS3 of MSRAction3D (see Figure 4.9). Thus, we can conclude that the accuracy of our descriptor is acceptable on the four benchmarks and that our method is robust to the dataset changing.



Start system	96.20	0.00	1.43	0.39	0.00	0.00	2.13	0.00	0.00	1.26	2.36	0.38
Duck	0.00	97.90	0.00	0.00	0.00	0.00	2.13	0.00	0.00	0.00	0.00	0.00
Push right	0.00	0.00	93.21	0.00	0.29	0.00	0.43	0.00	0.00	1.68	0.00	0.00
Goggles	0.00	0.00	0.00	92.97	0.00	0.00	0.00	0.00	7.22	0.00	0.39	0.00
Wind it up	0.00	0.00	0.00	0.00	94.44	0.00	0.00	0.00	0.00	1.26	1.18	0.00
Shoot	2.11	0.42	4.29	0.00	0.00	99.62	0.00	1.18	1.08	0.00	0.39	2.64
Bow	0.00	0.84	0.00	0.00	0.00	0.00	95.32	0.00	0.00	0.00	0.00	0.00
Throw	0.00	0.00	0.00	0.00	0.29	0.00	0.00	92.52	0.00	0.00	0.00	0.00
Had enough	0.84	0.42	0.36	5.86	0.00	0.00	0.00	1.18	89.53	0.00	3.94	0.00
Change weapon	0.00	0.42	0.36	0.00	0.58	0.38	0.00	0.00	0.00	92.44	0.00	0.38
Beat both	0.84	0.00	0.36	0.78	4.39	0.00	0.00	0.00	2.17	3.36	91.73	0.75
Kick	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.12	0.00	0.00	0.00	95.85

FIGURE 4.11: Confusion matrix obtained using the descriptor KSC-NPMSE-l_2 on the dataset MSRC12

4.4.3 A trade-off between computational latency and accuracy

The low-latency and the good accuracy of our descriptor KSC-NPMSE show that this latter realizes a trade-off between computational latency and accuracy. However, it is difficult to fuse these two information. Thus, we propose to use the same graph, used to illustrate the accuracy and the MET at the same time, proposed in Chapter 3 (see Figure 4.12). We can notice that KSC-NPMSE-l_2 compared to other descriptors is accurate and presents a very low computational latency.

4.4.4 NAE vs NPMSE

By analyzing the experiments, we can conclude that the NMPSE function gives better results than the NAE function on the four datasets. Indeed, we can notice a difference of around 5% on MSRACtion3D dataset, 12% of UTKinect dataset, 10% on MSRC12 dataset and 8%(SV)-11%(DV) on Multiview3D dataset between the descriptors KSC-NAE and KSC-NPMSE-l_2, while the execution time per descriptor remains the same. Also, compared to the distance l_1 and l_∞, the distance l_2 gives slightly more accurate results. For this reason, only the KSC-NPMSE-l_2 will be taken into account in the rest of the experiments. In future

Descriptor	Same view (%)	Different view (%)	MET (s)
Actionlet [94]	87.1	69.7	-
HOG2 [63]	87.8	74.2	9.06
HON4D [65]	89.3	76.6	17.51
SNV [102]	94.27	76.65	271.72
JP [91]	96.00	88.10	1.22
RJP [91]	97.70	92.7	4.58
Q [91]	91.30	72.10	2.52
LARP [91]	96.00	88.1	10.51
KSC-NAE (ours)	82.12	79.43	0.09
KSC-NPMSE-l_1 (ours)	**90.63**	89.67	0.091
KSC-NPMSE-l_{infty} (ours)	89.93	89.06	**0.09**
KSC-NPMSE-l_2 (ours)	90.45	**90.10**	0.092

TABLE 4.6: Accuracy of recognition and MET per descriptor on Multiview3D dataset

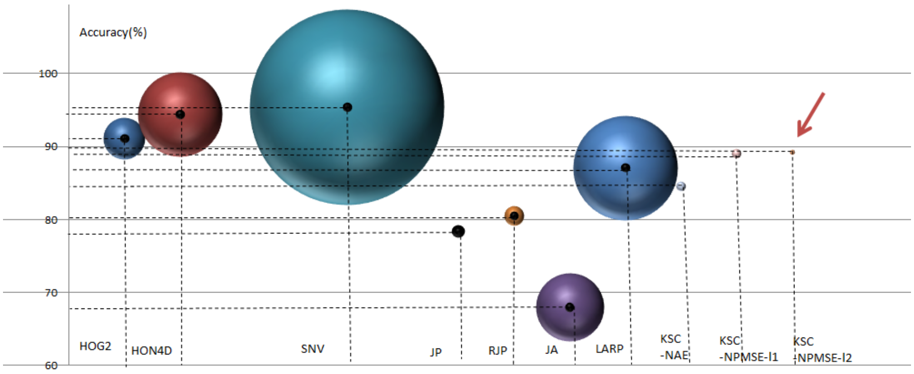


FIGURE 4.12: Illustration of the accuracy and the MET of different descriptors on MSRAction3D dataset: We recall that the center of every ball represents the accuracy of each corresponding method, while the surface area of the ball returns the MET.

work, it could be interesting to compare a wider range of more sophisticated distances. By observing the shape of the two curves (NAE and NPMSE-l_2) as shown in Figure 4.13, it can be noticed that the NMPSE is smoother than NAE and that NMPSE increases importantly only in the presence of key poses, while NAE increases more uniformly. The superiority of NMPSE as a TVRF function could be explained by the fact that the notion of key frames is more exploited with the use of NMPSE.

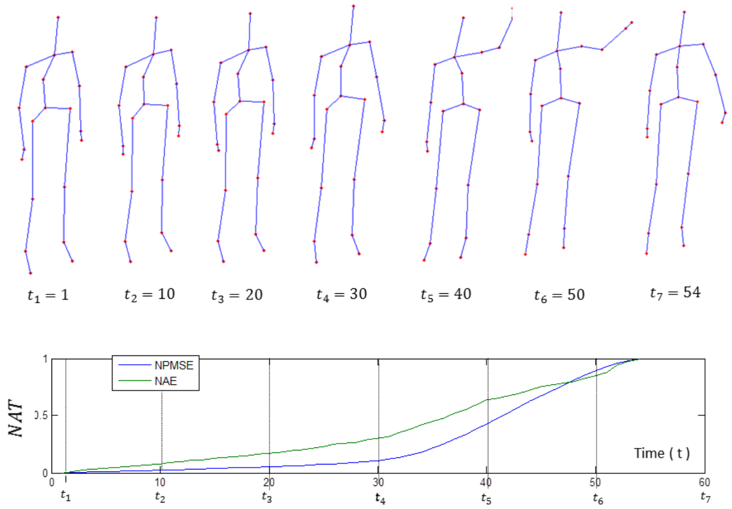


FIGURE 4.13: The visualization of the NAE (green) and the NPMSE (blue) varying over time. Skeletons at some instants t_k are also visualized.

Deleted Process	MSRAction3D (%)	UTKinect (%)	Multiview3D SV (%)	Multiview3D DV (%)	MSRC12 (%)
Nothing	**89.54**	**96.00**	**90.45**	**90.10**	**94.27**
without S.N.	84.49	87.00	88.27	87.76	92.71
without T.N.	76.73	87.00	78.82	78.60	81.43

TABLE 4.7: Effect of each process on the accuracy of recognition using KSC-NPMSE-l_2

4.4.5 Benefits of Spatial Normalization (SN)

Table 4.7 reports the role of the different processes of normalization. Each column of the table presents the obtained accuracy of recognition, after deleting one of the proposed processes contributing to spatial and temporal normalization. We can observe that SN contributes considerably to enhancing the accuracy on the three datasets. We observe an increase of around 5% for MSRAction3D dataset, 9% for UTKinect, dataset approximately 2% for MSRC12 dataset and around 2 % for Multiview3D dataset (for both SV and DV tests). To explain the use of a unitary euclidean normalization instead of the use of an average skeleton as in [105], we report the results of our method combined with an average skeleton-based normalization on the four used datasets. The accuracy of recognition is very low compared with the values obtained using our spatial normalization algorithm with only 81.92% on MSRAction3D, 84% on UTKinect and 87.5%(SV) - 86.95% (DV) on Multiview3D datasets.

Kinematics	MSRAction3D (%)	UTKinect (%)	Multiview3D SV (%)	Multiview3D DV (%)	MSRC12 (%)
P+V+A	**89.54**	**96.00**	90.45	**90.10**	**94.27**
P+V	86.06	92.00	**91.15**	88.98	93.92
P	86.04	90.00	85.07	85.24	93.50
V	83.01	90.00	81.77	80.99	89.68
A	82.39	81.00	78.99	75.78	86.46

TABLE 4.8: Effect of each kinematic component on the accuracy of recognition using KSC-NPMSE-l_2

orientation	0°	30°	$-30°$
0°	88.54	95.83	91.67
30°	91.67	94.79	85.42
-30°	89.58	91.67	93.75

orientation	0°	30°	$-30°$
0	90.63	91.67	90.63
30	88.54	87.50	88.54
-30°	88.54	87.50	87.50

TABLE 4.9: Accuracy of every test on Multiview3D dataset: We detail here the accuracy obtained for every test. The table on the left represents the results given when the training data are performed by subjects 1,2,3 and 4, while the table on the right represents the results given when the training data are performed by subjects 5,6,7 and 8 as proposed in [41]. The orientation specified in the columns represents the orientation of the data used for the training, while the orientation specified in the lines represents the orientation of the data used for testing.

4.4.6 Benefits of Temporal Normalization (TN)

In Table 4.7, the removal of the process of TN shows its important role in our method. Indeed, without TN, the accuracy of recognition decreases from 89.54% to 76.73% on MSRAction3D, from 96% to 87% on UTKinect, from 90.45%(SV)-90.10%(DV) to 78.82%(SV)-78.65%(DV) on Multiview3D and from 94.27% to 81.43% on MSRC12.

4.4.7 Benefits of kinematic features

In Table 4.8, we evaluate the importance of each kinematic term. This table shows that position is generally the most discriminative value followed by velocity and acceleration. This may be due to the increase of the approximation error caused by the derivations. Nevertheless, the combined information of the three kinematic values give us the best amount of accuracy on the four datasets.

Kinematics	MSRAction3D (%)	UTKinect (%)	Multiview3D SV (%)	Multiview3D DV (%)	MSRC12 (%)
linear	10.68	95.00	91.61	91.15	7.57
nearest	10.68	95.00	**92.53**	**91.41**	7.57
pchip	88.98	**96.00**	91.67	91.32	94.17
v5cubic	88.98	95.00	90.45	90.10	7.58
spline	**89.54**	**96.00**	90.10	90.10	**94.27**

TABLE 4.10: Effect of the different interpolation approaches on the accuracy of recognition using KSC-NPMSE-l_2

4.4.8 Robustness to view-point variation

Table 4.5 gives the results obtained on Multiview3D dataset. The results prove that our method is the most robust to view-point variation. Indeed, KSC-NAE and KSC-NPMSE present a very slight difference between the accuracies registered for SV and DV tests, with less than 2% (superior to 5% for other state-of-the-methods). The most robust descriptor to view-point variation is the KSC-NPMSE-l_2, with only 0.35% of difference between the two tests. To detail the different tests, Table 4.9 is presented. We can see that the SV and DV tests register accuracies in the same range (between 87.50% and 95.83%).

4.4.9 Cubic spline interpolation role

This paragraph discusses the choice of the interpolation method. To show the benefits of cubic spline interpolation, Table 4.10 reports the accuracy when the interpolation strategy is changed. The different interpolation techniques cited in Section 4.2.2 are tested combined with our method: "linear", "nearest", "pchip","V5cubic" and "spline" (our choice). The results show that the cubic spline interpolation used does not give always the best results in terms of accuracy. However, this kind of interpolation is the most stable and always gives relatively good results. For example, on Multiview3D dataset, the best results are given by the interpolation "nearest" (with 92.53%(SV)-91.41%(DV) against 90.45%(SV)-90.10%(DV) for "spline"). Nevertheless, this technique gives very bad results on MSRAction3D dataset (with 10.62% against 89.54% for "spline").

4.4.10 Parameter s influence

The parameter s, which represents the number of samples affects the accuracy of recognition. As shown in the Table 4.11, with varying the parameter s, the accuracy of recognition will also vary. The parameter s is therefore fixed according to the best score obtained ($s = 20$

number of samples s	5 (%)	10 (%)	15 (%)	20 (%)	25 (%)	30 (%)	35 (%)	40 (%)	45 (%)	50 (%)
MSRAction3D	87.21	88.04	89.27	**89.54**	88.95	88.67	88.08	88.08	88.35	88.37
UTKinect	94.00	95.00	**96.00**	93.00	95.00	95.00	95.00	94.00	94.00	94.00
Multiview3D (SV)	88.37	90.10	**90.45**	89.76	90.28	90.10	90.45	90.28	**90.45**	**90.45**
Multiview3D (DV)	87.59	89.15	**90.10**	89.76	89.50	89.67	89.93	89.67	89.50	89.06
MSC12	92.20	92.96	93.91	93.92	93.69	94.04	94.11	**94.27**	94.08	93.98

TABLE 4.11: Effect of the number of samples s on the accuracy of recognition using KSC-NPMSE-l_2

for MSRAction3D, $s = 15$ for UTKinect and Multiview3D and $s = 40$ for MSRC12). It is important to notice that the choice of s does not affect the results considerably. For the tested values, we observe a decrease of up to 3% compared with the highest score of accuracy for a number of samples varying between 5 and 50.

4.5 Conclusion

In this chapter, we have presented a novel descriptor for fast action recognition, called Kinematic Spline Curves (KSC). It is based on the cubic spline interpolation of kinematic values of joints, more precisely, position, velocity and acceleration. To make our descriptor invariant to anthropometric variability and execution rate variation, we perform a skeleton normalization as well as a temporal normalization. For this reason, a novel method of temporal normalization is proposed called TVR. The proposed method has shown its efficiency in terms of accuracy and computational latency in four different datasets. In this way, this technique could be applied in real-world applications requiring fast calculation. In this thesis, actions are assumed to be already segmented. In future work, the issue of temporal segmentation will be studied. Some techniques developed for dynamical switched models can be used to detect different modes in an unsegmented sequence, allowing the detection of particular points of transition from an action to another or from an action to the rest state. First attempts have been made to adapt these methods in [10, 62, 92, 55]. However, this method has some limitations: the need of energy calculation on the whole sequence to perform the temporal normalization represents an issue for its online extension. Some techniques of prediction could be interesting, but that remains a large topic to study. For this reason, we investigate another additional fast and accurate action descriptor, which could be more suitable for an online use.

Chapter 5

Symmetric Positive semi Definite matrices as descriptors

5.1 Introduction

In the last chapter, we have presented a novel fast and accurate descriptor for action recognition called Kinematic Spline Curves (KSC). However, as explained earlier, this descriptor is unsuitable for online action recognition.

In this way, to make an online recognition possible, we suggest to make use of covariance descriptors. Indeed, in contrast to the KSC descriptor, this representation does not need the knowledge of the whole sequence in order to be computed: it could be calculated on subranges of the sequence. Thus, we propose a novel descriptor called Kinematic Covariance (KC) descriptor by integrating the Kinematic Features (KF) presented in the last chapter in a covariance matrix. Over the past decade, non singular covariance matrices have been proven to be very efficient descriptors in the field of pattern recognition thanks to their various advantages. Furthermore, they are very suitable to online recognition since they can be incrementally computed.

Nevertheless, non singular covariance descriptors do not form a vector and are elements of the space of SPD matrices. Thus, researchers have extended kernel learning methods to the Riemannian space of SPD matrices in order to exploit machine learning techniques. However, in the case of action recognition, singular covariance matrices are hardly avoided because the dimension of features could be higher than the number of samples. More details can be found in Section 5.4.3. Such covariance matrices (non singular and singular) belong to the space of Symmetric Positive semi-Definite (SPsD) matrices.

Thus, in order to classify actions using KC descriptors, we propose to adapt kernel methods and more particularly Support Vector Machines (SVM) to the space of SPsD matrices by introducing a novel and simple formulation of the distance on the space of SPsD matrices. This new distance represents an extension of the Log-Euclidean distance for the space of SPD matrices [5] and is called Modified-Log-Euclidean (MLE). This approach is static since it does not include the temporal ordering information.

Hence, to overcome this limitation, a second descriptor called Hierarchical Kinematic Covariance (HKC) is introduced. This latter is composed of KC descriptors calculated on sub-ranges of the whole sequence. The recognition is done with the use of a Multiple Kernel Learning (MKL) classifier based on the extended version of SVM for the space of SPsD matrices.

In the following, we start by a brief review of methods making use of covariance matrices in computer vision. For a better understanding, we recall the mathematical background related to this work. Then, the Kinematic Covariance (KC) descriptor is introduced and the issue of classification based on this representation is formulated. After that, the kernel-based methods are extended to the space of SPsD matrices, by proposing a novel distance for this space, the Modified Log-Euclidean distance. Using this extension, KC descriptors are classified based on an SVM model. Then, the Hierarchical Kinematic Covariance (HKC) descriptor, as well as the extended version of SVM-based Multiple Kernel Learning (MKL) classifier are introduced. Finally, the results obtained on the three different benchmarks are reported in an experimental section, showing the effectiveness of the proposed approaches in terms of accuracy, computational latency and observational latency.

5.2 Covariance matrices in computer vision

In this section, we present methods in computer vision which make use of covariance matrices. As action recognition is composed of two parts which are action description and action classification, we divide this state-of-the-art review into two parts: covariance descriptors and classification of covariance matrices. The first part exposes how covariance matrices have been used as descriptors in computer vision, while the second part recalls methods proposed to classify covariance matrices.

5.2.1 Covariance descriptor

In this chapter, we focus on covariance descriptors which have attracted great interest of researchers. In 2006, they have been introduced as descriptors for the first time in the field of computer vision [88]. Then, these features have been applied to object recognition [88], classification of image sets [96], pedestrian detection [46], face recognition [67], action recognition [45], etc. This popularity is mainly due to the good properties of covariance matrices. Indeed, they can be used to fuse heterogeneous features, are robust to occlusion and partially invariant to rotation and scale. Furthermore, they contain the information of correlation between features and are therefore very informative. Also, it has been shown in two recent papers that covariance descriptors are adapted to the case of online action recognition [51, 85] due to their ability to be incrementally calculated.

5.2.2 Classification of covariance matrices

In computer vision, the need of classification is very often present. One of the most popular classification approaches are the distance-based machine learning techniques such as kNN, Neural Networks, kernel methods (SVM, MKL), etc. The distance information between descriptors is very often used by this family of methods to estimate the region of each class. Classical distance-based machine learning algorithms have been developed with the use of the Euclidean distance, assuming that descriptors are expressed in the vector space $\mathbb{R}^n$, with n representing the dimension of the descriptor.

However, informative descriptors in computer vision do not necessary form a vector space. Thus, applying distance-based approaches without paying attention to the particular geometry of the feature space can lead to bad performances.

That is the case of covariance matrices which have been assumed to be non singular in earlier papers [88, 67, 96] and, consequently to be Symmetric Positive Definite (SPD). More details can be found in Section 5.4.3. It can be noted that many recent methods using covariance descriptors have proposed to generalize distance-based algorithms to the Riemannian manifold of Symmetric Positive Definite (SPD) matrices [46, 88, 39].

As mentioned before, covariance matrices have been widely used in computer vision for tasks such as object recognition, face recognition, pedestrian recognition, etc. Given that non singular covariance matrices are elements of the Riemannian manifold of the SPD matrices, many researchers have made attempts to formulate the geodesic distance of the space of SPD matrices. Indeed, these distances are important knowing that they can be used to extend meaningful distance-based machine learning algorithms. In 2003, Förstner et al. [33] have introduced the Affine-Invariant distance by considering the Riemannian structure of the space of SPD matrices. To alleviate the excessive execution time required to calculate Affine-Invariant distance, a novel distance has been introduced by Pennec et al. [5], called the Log-Euclidean distance. Other distances for the space of SPD matrices have been proposed such as the Stein distance [84], the Cholesky distance [50], etc. Nevertheless, the most popular remain the Affine-Invariant and the Log-Euclidean distances as they take into account the Riemannian geometry of the space of SPD matrices. More theoretical details about these distances will be exposed in Section 5.4. Based on them, distance-based learning algorithms for the space of SPD matrices have been proposed in order to make use of SPD matrices as descriptors in pattern recognition applications. In [85, 51], Affine-Invariant and Log-Euclidean distances are respectively used to build a k Nearest Neighbors (kNN) model. Recently, Jayasumana et al. [46] have extended the Radial Basis Function (RBF) kernel to the space of SPD matrices. After that, this novel kernel has been combined with different kernel-based machine learning algorithms such as SVM, MKL and PCA.

In the context of action recognition, obtained covariance matrices are mostly singular because of the need of an important number of features. Therefore, the distance-learning methods become unsuitable since singular covariance matrices are not SPD, but are Symmetric Positive semi-Definite (SPsD). This issue will be described in greater details in Section 5.4.3. For this reason, an extension of kernel based methods for SPsD matrices is proposed in this chapter. For a better understanding of this extension, we start by presenting the mathematical background related to these works.

5.3 Mathematical background

First, the mathematical notations are defined. Second, we recall the principle of kernel learning methods, which have been initially designed for Euclidean spaces. Then, the Riemannian manifold of SPD matrices is presented. Finally, the recent extension of kernel based methods for the space of SPD matrices [46] is reviewed.

5.3.1 Notations

In this part, we define the different mathematical notations

$M_d(\mathbb{R})$ is the vector space of square $d \times d$ matrices.

The transpose of a matrix M is denoted by $\mathrm{M}^{\mathbf{T}}$.

The inverse of an invertible matrix M is denoted by $\mathrm{M}^{-\mathbf{1}}$.

Suppose that $\mathbf{A}$ is a diagonalizable matrix of $M_d(\mathbb{R})$ and that the diagonal matrix in Equation (5.5), denoted by $\mathbf{D_A}$ contains ordered eigenvalues. $\mathbf{P}$ represents the transformation matrix.

$$\mathbf{A} = \mathbf{P}\mathbf{D_A}\mathbf{P^{-1}} \tag{5.1}$$

$Sym_d(\mathbb{R})$ is the vector space of $d \times d$ symmetric matrices.

$$Sym_d(\mathbb{R}) = \left\{\mathbf{M} \in M_d(\mathbb{R}),\ \mathbf{M}^T = \mathbf{M}\right\} \tag{5.2}$$

$GL_d(\mathbb{R})$ is called General Linear group and represents the group of $d \times d$ invertible matrices.

$$GL_d(\mathbb{R}) = \{\mathbf{M} \in M_d(\mathbb{R}),\ det(\mathbf{M}) \neq 0\} \tag{5.3}$$

$Sym_d^{+*}(\mathbb{R})$ is the space of $d \times d$ Symmetric Positive Definite (SPD) matrices.

$$Sym_d^{+*} = \left\{\mathbf{M} \in M_d(\mathbb{R}),\ \mathbf{M}^T = \mathbf{M} \text{ and } \forall \mathbf{X} \in \mathbb{R}^d \text{ and} X > 0,\ \mathbf{X^T M X} > 0\right\} \tag{5.4}$$

$Sym_d^{+}(\mathbb{R})$ is the space of Symmetric Positive semi-Definite (SPsD) $d \times d$ matrices.

$$Sym_d^{+} = \left\{\mathbf{M} \in M_d(\mathbb{R}),\ \mathbf{M}^T = \mathbf{M} \text{ and } \forall \mathbf{X} \in \mathbb{R}^d,\ \ \mathbf{X^T M X} \geq 0\right\} \tag{5.5}$$

5.3.2 Kernel learning methods

Kernel learning methods represent distance-based machine learning algorithms which make use of kernel functions to evaluate data in a space of higher dimension.

Let us suppose that data are described in a feature space E, which represents a d dimensional vector space equipped with an inner product $\langle \cdot, \cdot \rangle_E$. Since the linear learning task is sometimes complex in the initial feature space E, the idea of non linear learning methods is to find a mapping function. This latter denoted by $\Phi : E \mapsto H$ makes the task easier, such as H is a Hilbert space (finite or infinite dimension) equipped with an inner product denoted by $\langle \cdot, \cdot \rangle_H$. Since kernel based methods use only the information of similarity between data, the explicit knowledge of the function Φ can be avoided. Thus, a kernel function $K : E \times E \mapsto \mathbb{R}$ is directly used to measure the similarity between the data in the space H, by mapping them implicitly, as depicted by Equation (5.6). $\forall \mathbf{x}, \mathbf{y} \in E$,

$$K(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle_H \tag{5.6}$$

To realize this kernel trick, Mercer's theorem which is recalled below has to be respected (Theorem 5.3.1).

Theorem 5.3.1. *A kernel function $K : E \times E \mapsto \mathbb{R}$ which is continuous, symmetric and positive semi-definite ($\forall \mathbf{x}_i, \mathbf{x}_j \in E$ and $\forall c_i, c_j \in \mathbb{R}$, $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0$) represents an inner product in a space of higher dimension .*

In this case, the kernel is called *Reproducing Kernel Hilbert Space (RKHS).*

In what follows, we give an example for which kernel functions could be useful, namely, the non linear Support Vector Machines (SVM).

Non linear Support Vector Machines

In the Appendix A, the linear Support Vector Machines method is recalled. This method of classification has been initially developed for the case of data that are linearly or almost linearly separable. In this way, kernel functions are used to extend Support Vector Machines to non linear classification. Since the SVM is based on the optimization of an equation including an inner product between data, this latter is replaced by a kernel function K as described by Equation (5.7).

$$\begin{aligned} \max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{with} \sum_{i=1}^{N} \alpha_i y_i ; \alpha_i \geq 0; \end{aligned} \tag{5.7}$$

We recall that N is the number of training data, $\mathbf{x}_i$ and $\mathbf{x}_j$ respectively the descriptors extracted from the instance i and j, y_i and y_j their associated labels and $\alpha = (\alpha_1, \alpha_2, ..., \alpha_N)$ the Lagrange multipliers.

This classification method and consequently the used kernel have been designed by assuming the vector space structure of the feature space E. To extend this method to a non linear space, the kernel function which is based on an inner product calculation should be adapted to the topology of this space. Since covariance descriptors have been supposed to be non singular and to belong to the Riemannian manifold of Symmetric Positive Definite matrices, the topology of this latter has been studied in order to propose suitable distances (that can be used for the construction of meaningful inner products).

5.3.3 Riemannian manifold of Symmetric Positive Definite matrices

In this part, we propose to present the Riemannian space of SPD matrices. However, we recall first some notions needed as the exponential and the logarithm of a matrix, as well as the logarithm calculation of digonalizable matrices.

Exponential and logarithm of a matrix

The exponential function, initially applied to real and complex numbers can be generalized to square matrices. The exponential $\exp(\mathbf{A})$ of a matrix $\mathbf{A} \in M_d(\mathbb{R})$ is defined by analogy based on the development on power series of the exponential function as depicted by Equation (5.8).

$$\exp(\mathbf{A}) = \sum_{k=0}^{\infty} \frac{\mathbf{A}^k}{k!} \text{ with } \mathbf{A}^0 = \mathbf{I}_d \tag{5.8}$$

The exponential of a square matrix always gives an invertible matrix. In other words, the exponential function represents a mapping from the space of square matrices $M_d(\mathbb{R})$ to the General Linear group $GL_d(\mathbb{R})$.

$$\begin{array}{rcccl} \exp & : & M_d(\mathbb{R}) & \to & GL_d(\mathbb{R}) \\ & & \mathbf{A} & \mapsto & \exp(\mathbf{A}) \end{array}$$

The matrix logarithm is defined as the inverse function of the matrix exponential. Let consider two matrices $\mathbf{A}, \mathbf{B} \in M_d(\mathbb{R})$. Thus,

$$\text{if } \mathbf{A} = \exp(\mathbf{B}) \text{ then } \mathbf{B} = Log(\mathbf{A}) \tag{5.9}$$

Logarithm calculation of a diagonalizable matrix

Here, we recall only the calculation of the logarithm of diagonalizable matrices because we only calculate the logarithm of SPD and SPsD matrices, which are symmetric and which are consequently diagonalizable. Let $\mathbf{A} \in M_d(\mathbb{R})$ be a diagonalizable matrix. Therefore, it exists a transformation matrix $\mathbf{P}$, which satisfies:

$$\mathbf{A} = \mathbf{P}\mathbf{D}_{\mathbf{A}}\mathbf{P}^{-1} \tag{5.10}$$

with $\mathbf{D}_A$ the diagonal matrix composed of $\mathbf{A}$'s eigenvalues which are denoted by $(\lambda_i)_{1\leq i\leq d}$.

$$\mathbf{D}_A = \begin{pmatrix} \lambda_1 & 0 & 0 \\ \vdots & \lambda_i & \vdots \\ 0 & 0 & \lambda_d \end{pmatrix}$$

Thus, the logarithm of a diagonalizable matrix is calculated thanks to this relation:

$$Log(\mathbf{A}) = \mathbf{P}Log(\mathbf{D}_A)\mathbf{P}^{-1} = \mathbf{P} \begin{pmatrix} Log(\lambda_1) & 0 & 0 \\ \vdots & Log(\lambda_i) & \vdots \\ 0 & 0 & Log(\lambda_d) \end{pmatrix} \mathbf{P}^{-1}$$

Properties of the Riemannian manifold of Symmetric Positive Definite matrices Differentiable manifolds are the base structures of differentiable geometry. They represent spaces which are locally similar to a Euclidean space. The tangent space at a given point p of a differentiable manifold M is defined as the vector space denoted by T_pM formed by all the tangent vectors to any curve of M passing through p. Differentiable manifolds are not necessary linear and measuring similarity between its elements with a Euclidean distance is very often unsuitable.

Riemannian geometry aims to analyze the properties of differential manifolds in order to propose a more representative measure. Indeed, a Riemannian manifold is a differentiable manifold characterized by a smooth inner product function on any tangent space of the manifold. The family of these inner products is referred to as Riemannian metric. This metric makes possible the definition of geometric entities such as angle between curves, geodesic distances, etc.

Thus, the geodesic distance between two points of a manifold is the length of the shortest curve of the manifold which connects them. Geodesic distance can be viewed as the extension of the Euclidean distance for non vector space. Figure 5.1 illustrates the interest of the geodesic distance in the presence of a non-linear manifold.

The space of Symmetric Positive Definite $d \times d$ matrices Sym_d^{+*} represents one of the well-known example of Riemannian manifold [4].

The exponential of a symmetric matrix is a matrix which is symmetric positive definite and vice versa the logarithm of a symmetric positive definite matrix is a symmetric matrix.

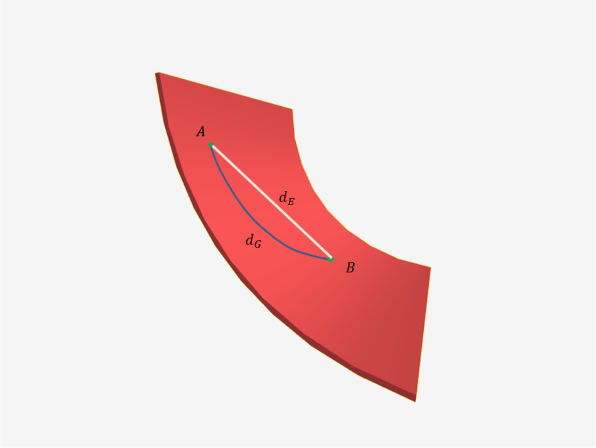


FIGURE 5.1:

Let us consider a 2D manifold (represented by the red surface) embedded in the space $\mathbb{R}^3$. The Euclidean distance d_E between A and B (in white) and the geodesic distance d_G between A and B (in blue) on a non-linear manifold are represented.

It is easy to demonstrate that $(Sym_d, +, .)$ has a vector space structure.

$$\begin{array}{rcccc} Log & : & Sym_d^{+*} & \rightarrow & Sym_d \\ & & \mathbf{A} & \mapsto & Log(\mathbf{A}) \end{array}$$

The set of non singular covariance matrices belongs to the Riemannian manifold Sym_d^{+*}. Because of the importance of covariance matrices in computer vision, many attempts have been made to propose an appropriate geodesic distance such as Affine-Invariant distance [33], Log-Euclidean distance[5], Cholesky distance[50], Root Stein Divergence distance [84], etc.

In what follows, we will only present the two distances called Affine-Invariant [33] and Log-Euclidean [5] which have several benefitial properties. To the best of our knowledge, these latter are the only real geodesic distances of the literature based on the particular geometry of Sym_d^{+*}.

Affine-Invariant distance

In [33], the proposed geodesic distance for Sym_d^{+*} that we denote by d_{AI} is described by Equation (5.11) with $\|.\|_F$ the norm of Frobenius.

$$d_{AI}(\mathbf{M}_1, \mathbf{M}_2) = \left\| Log(\mathbf{M}_1^{-1/2}\mathbf{M}_2\mathbf{M}_1^{-1/2}) \right\|_F \quad \text{with } \mathbf{M}_1, \mathbf{M}_2 \in Sym_d^{+*} \tag{5.11}$$

Note that because $\mathbf{M}_1^{-1/2}\mathbf{M}_2\mathbf{M}_1^{-1/2} \in Sym_d^{+*}$, the logarithm of this matrix is an element of Sym_d. The norm of Frobenius can be therefore used to calculate the distance in the Lie algebra Sym_d. It is demonstrated to be invariant to congruence transformation and to inversion. Although Affine-Invariant distance has very interesting specificity, it is very greedy in terms of computing time [5].

Log-Euclidean distance

In [5], the authors introduce a novel distance for Sym_d^{+*} in order to overcome the high complexity of the Affine-Invariant distance. The space $(Sym_d^{+*}, +, .,)$ equipped with a binary operation '$+$' and a scalar operation '.' does not define a vector space. In fact, $\mathbf{M}_2 = \alpha\mathbf{M}_1 + \mathbf{M}$, with $\alpha \in \mathbb{R}$ and $\mathbf{M}, \mathbf{M}_1 \in Sym_d^{+*}$ is not necessary Symmetric Positive Definite and does not systematically belong to the space Sym_d^{+*}. The authors propose to construct two new operations (binary and scalar) ensuring the vector space structure of Sym_d^{+*}. These two operations called internal logarithmic multiplication denoted by $\oplus$ and external logarithmic multiplication denoted by $\otimes$ are defined as follows. Let us suppose that $\mathbf{M}_1, \mathbf{M}_2 \in Sym_d^{+*}$. Therefore,

$$\mathbf{M}_1 \oplus \mathbf{M}_2 = exp(Log(\mathbf{M}_1) + Log(\mathbf{M}_2)) \in Sym_d^{+*} \tag{5.12}$$

$$\alpha \otimes \mathbf{M}_1 = exp(\alpha.Log(\mathbf{M}_1)) = \mathbf{M}_1^{\alpha} \in Sym_d^{+*} \tag{5.13}$$

We recall that if the matrices are invertible (which is the case of SPD matrices), its logarithm is unique [21, 11].

The Log-Euclidean distance that we denote by d_{LE} is therefore defined by Equation (5.14).

$$d_{LE}(\mathbf{M}_1, \mathbf{M}_2) = \|Log(\mathbf{M}_1) - Log(\mathbf{M}_2)\|_F \tag{5.14}$$

It is invariant to inversion, to translation in the logarithmic space but is not completely invariant to affine transformations (contrary to Affine-Invariant distance). The main advantage of this distance is its rapidity of calculation. In the next section, we will show how these distances can be very useful to extend machine learning algorithms. In particular, we will review the recent extension proposed by Jayasumana et al. [46] who make use of the Log-Euclidean distance to extend kernel leaning methods to the space of SPD matrices.

5.3.4 Kernel learning on Symmetric Positive Definite matrices

Many attempts have been made to generalize these methods to non linear spaces such as Riemannian manifolds. Recently, as specified earlier in Section 5.2.2, Jayasumana et al. [46] have adapted methods based on the Radial Basis Function (RBF) kernel to the space Sym_d^{+*}. To realize that, the Euclidean distance has been replaced by the Log-Euclidean distance.

Each point $\mathbf{x}$ of the non linear space M can be mapped to a feature vector $F(\mathbf{x})$ of a Hilbert space H. A kernel function $K : M \times M \rightarrow \mathbb{R}$ defining an inner product on H, has been used to map the data to a space of a higher dimension. However, as proved in Mercer's theorem (Theorem 5.3.1), the kernel has to represent an RKHS. The main challenge is to prove that the kernel is positive definite (the definition of a continuous and symmetric function is easier to realize).

The RBF kernel has demonstrated its efficiency for Euclidean feature space. It maps points from the feature space to a Hilbert space of infinite dimension. In $\mathbb{R}^n$, the RBF kernel denoted by K_G is expressed as follows,

$$K_G(\mathbf{x}_i, \mathbf{x}_j) = exp - \frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{2\sigma^2} \tag{5.15}$$

$\mathbf{x}_i$ and $\mathbf{x}_j$ respectively represent the descriptor extracted from the instance i and j, while σ^2 is a parameter to fix and represents the variance.

This kernel is calculated using the Euclidean distance between the points $\mathbf{x}_i$ and $\mathbf{x}_j$. To adapt this function to the Riemannian manifold Sym_d^{+*}, the Euclidean distance is replaced by a distance designed for Sym_d^{+*}. Considering only geodesic distances on Sym_d^{+*}, Jayasumana et al. have shown that the Log-Euclidean distance is the only Sym_d^{+*} distance which makes the RBF kernel positive definite. We recall the theorem demonstrated in their paper [46]:

Theorem 5.3.2. *Let (M, d) be a space equipped with a distance d and let $K_G^M : M \times M \rightarrow \mathbb{R}$ be a function with $K_G^M(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{d^2(\mathbf{x}_i, \mathbf{x}_j)}{2\sigma^2})$, and $\mathbf{x}_i, \mathbf{x}_j \in M$. Therefore, K_G^M is a positive definite kernel $\forall \sigma$ if and only if it exists a prehilbertian space V and a function $\phi : M \rightarrow V$ with $d(\mathbf{x}_i, \mathbf{x}_j) = \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|_V$.*

Based on the stated theorem, the authors have finally formulated Corollary 5.3.1.

Corollary 5.3.1. *Let suppose that $K_G^{SPD} : Sym_d^{+*} \times Sym_d^{+*} \rightarrow \mathbb{R}$ with $K_G^{SPD}(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{d_{LE}^2(\mathbf{x}_i, \mathbf{x}_j)}{2\sigma^2})$ and $d_{LE}(\mathbf{x}_i, \mathbf{x}_j) = \|Log(\mathbf{x}_i) - Log(\mathbf{x}_j)\|_F$ for $\mathbf{x}_i, \mathbf{x}_j \in Sym_d^{+*}$. Then, K_G^{SPD} is a positive definite kernel $\forall \sigma \in \mathbb{R}$.*

It is easy to notice that the distance d_{LE} is unsuitable to SPsD matrices. Let $\mathbf{x}$ be a singular SPsD matrix. Therefore, $\mathbf{x}$ is diagonalizable and a transformation matrix $\mathbf{P}$ exists such as $\mathbf{x} = \mathbf{P}^{-1}\mathbf{D_x}\mathbf{P}$. Therefore, $Log(\mathbf{x})$ is equal to $\mathbf{P}^{-1}Log(\mathbf{D_x})\mathbf{P}$. However, because $\mathbf{x}$ is positive semi-definite, the diagonal matrix $\mathbf{D_x}$ contains at least one eigenvalue which is equal to 0, leading to the calculation of $Log(0)$ which is not defined.

In next section, we present the novel Kinematic Covariance (KC) descriptor and show why this latter can be an SPsD matrix, leading to a barrier in classification.

5.4 Kinematic Covariance descriptor and problem formulation

In this section, the computation of covariance descriptors based on low dimensional features developed for object recognition is recalled. Then, the novel Kinematic Covariance (KC) descriptor for human action recognition is introduced and finally the issue of classification of this descriptor belonging to the space of SPsD matrices is detailed.

5.4.1 Pixel-based covariance descriptor

Let $\mathbf{x_i} \in \mathbb{R}^d$ be a d-dimensional feature vector $\forall i$ and let us suppose that $\mathbf{D}_{d\times N} = [\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_N}]$ represents the data matrix, N being the number of samples. In [88], the region covariance descriptor is calculated as follows,

$$\mathbf{C} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x_i} - \mu)(\mathbf{x_i} - \mu)^T \tag{5.16}$$

with μ the mean of samples. Therefore, the covariance descriptor $\mathbf{C}$ represents a $d \times d$ matrix which is assumed to be non singular and to be consequently an element of the space of Symmetric Positive Definite (SPD) matrices denoted by Sym_d^{+*}.

Assuming that covariance matrices are non singular can be reasonable for applications using descriptors for which N is largely superior to the number of features d, such as region descriptors. In these applications, the required number of features is very small compared to the number of samples (pixels in the case of [88]).

5.4.2 Kinematic covariance descriptor

As mentioned before, we propose to introduce a novel descriptor making use of statistical and kinematic tools that we call Kinematic Covariance (KC). So, the Kinematic Features (KF) presented in Section 4.3.2 are integrated in a covariance matrix. We recall that to ensure the invariance to anthropometric variability, skeletons are first normalized. Then, kinematic Features representing low-level features are calculated with the use of normalized skeletons. We remind that t_k represents the instant associated to the k^{th} frame. As presented in Chapter 4, at every instant t_k, $\mathbf{KF}(t_k) = [\mathbf{p}^{\mathbf{norm}}(t_k), \mathbf{V}(t_k), \mathbf{A}(t_k)]$ is composed of normalized joint positions $\mathbf{P}^{\mathbf{norm}}(t_k)$, joint velocities $\mathbf{V}(t_k)$ and joint accelerations $\mathbf{A}(t_k)$.

We recall that the dimension of the vector $\mathbf{KF}(t_k)$ is equal to $d_1 = 9 \times n$, with n the number of joints. We suppose that N is the number of frames in the segmented sequence. Thus, the novel Kinematic Covariance (KC) descriptor is computed as described by Equation (5.17),

$$\mathbf{KC} = \frac{1}{N}\sum_{k=1}^{N}(\mathbf{KF}(t_k) - \nu)(\mathbf{KF}(t_k) - \nu)^T \tag{5.17}$$

where $\nu = \sum_{k=1}^{N} \mathbf{KF}(t_k)$ is the mean of Kinematic Features. We can deduce that the Kinematic Covariance descriptor $\mathbf{KC}$ represents a square matrix of dimension $d_1 \times d_1$.

5.4.3 Problem formulation: Kinematic Covariance descriptor and the space of Symmetric Positive semi-Definite matrices

For Kinematic Covariance descriptor, the assumption claiming that covariance descriptors are non singular is not valid anymore since an important number of features is used.

Since the number of joints is in general equal to $n = 20$ and the number of frames in the majority of datasets is inferior to 100 frames, we can conclude that $d_1 = 180$ is generally superior to the number of frames (which represents the number of samples used to calculate the matrix of covariance). This fact leads to singular covariance matrices.

In a more formal manner, if the Kinematic Covariance matrix is full-ranked ($rank(\mathbf{KC}) = d_1$), the matrix is not singular and is therefore SPD ($\mathbf{KC} \in Sym_d^{+*}$). Nonetheless, the rank of a $d_1 \times d_1$ covariance matrix respects this inequality $rank(\mathbf{KC}) \leq min(d_1, N-1)$ and it can be noted that if $d_1 > N$, then $rank(\mathbf{KC}) < d_1$, implying the singularity of the matrix $\mathbf{KC}$. Such matrices are not positive definite since they have at least one eigenvalue equals to zero. In reality, these matrices are Symmetric Positive semi-Definite (SPsD). Indeed, $d_1 \times d_1$

covariance matrices (non singular and singular) are elements of the space of SPsD matrices denoted by Sym_d^+.

To overcome this numerical limitation, the majority of papers has chosen to work with a very restricted number of features $d_1 < n$ [45, 51]. Then, various geodesic distances have been proposed for the space Sym_d^{+*} making the classification in Sym_d^{+*} possible, as presented in Section 5.3.4. Nevertheless, if the use of a more important amount of features is needed for a better discrimination, the distance-based classification methods developed for Sym_d^{+*} become unsuitable. Therefore, the main problem would be to know how to generalize distance-based machine learning algorithms to the space Sym_d^+ in order to classify actions using KC descriptors? The next section is devoted to the resolution of this issue.

5.5 Action classification using Kinematic Covariance descriptors and an extended version of kernel learning for SPsD matrices

In order to classify actions using Kinematic Covariance descriptors, an extension of kernel learning methods and more specifically of RBF-based kernel methods able to classify SPsD matrices is proposed. As shown in Equation (5.15), when features are expressed in $\mathbb{R}^n$, the RBF kernel depends on the Euclidean distance. Thus, in order to adapt this method to the space Sym_d^+, a novel distance called Modified Log-Euclidean for SPsD matrices is first introduced. Then, the RBF kernel is extended to the space Sym_d^+ using this distance. Finally, Kinematic Covariance descriptors are classified using an SVM classifier based on the extended version of the RBF kernel.

5.5.1 Modified Log-Euclidean distance

Since our goal is to extend kernel methods to the space of SPsD matrices (the space of Kinematic Covariance matrices), a distance for Sym_d^+ is introduced. Instead of proposing a distance by analyzing the particular geometry of Sym_d^+, we modify the Log-Euclidean designed for Sym_d^{+*} which has been already used to extend kernel methods. This subsection describes step by step the formulation and the validity of the proposed distance that is called Modified Log-Euclidean distance.

First, we state a theorem showing that it exists a one-to-one correspondence between SPsD and a subset of SPD. Then, based on this theorem, we construct the distance by using a mapping function between these two spaces thanks to an extension of the Log-Euclidean distance.

Theorem 5.5.1. $\forall \epsilon > 0$, *it exists a bijective relation* ψ *between* Sym_d^+ *and a set* S *defined as:*

$$\begin{array}{ccccc} \psi & : & Sym_d^+ & \rightarrow & S \\ & & \mathbf{M} & \mapsto & \mathbf{M} + \epsilon \mathbf{I}_d \end{array}$$

with $S \subset Sym_d^{+*}$. $\mathbf{I}_d$ *represents the identity matrix of dimension* d.

Proof of Theorem 5.5.1. First of all, we show that $\forall \mathbf{M} \in Sym_d^+$, $\psi(\mathbf{M})$ is Symmetric Positive Definite (SPD):

1) Symmetric: $\mathbf{M}$ is symmetric and $\mathbf{I}_d$ is symmetric, as well as $\epsilon \mathbf{I}_d$. Therefore, $\mathbf{M} + \epsilon \mathbf{I}_d$ is symmetric.

2) Positive definite: Let $\mathbf{X} \in \mathbb{R}^d$ such as $\mathbf{X} = [x_1, x_2, ..., x_d]^T$,

$$\begin{aligned} \mathbf{X}^T \psi(\mathbf{M})\mathbf{X} &= \mathbf{X}^T(\mathbf{M} + \epsilon \mathbf{I}_d)\mathbf{X} \\ &= \mathbf{X}^T\mathbf{M}\mathbf{X} + \epsilon \mathbf{X}^T\mathbf{X} \\ &= \mathbf{X}^T\mathbf{M}\mathbf{X} + \epsilon \sum_{i=1}^{d} x_i^2 > 0 \end{aligned} \tag{5.18}$$

We can deduce that $\phi(\mathbf{M})$ is symmetric positive definite and consequently $S \subset Sym_d^{+*}$.

For a fixed $\epsilon > 0$, let us suppose that $\mathbf{Y} \in S$. Then, it exists therefore a matrix $\mathbf{M} \in Sym_d^+$ with $\mathbf{Y} = \mathbf{M} + \epsilon \mathbf{I}_d$. So, $\mathbf{M} = \mathbf{Y} - \epsilon \mathbf{I}_d$ which is unique. Thus, the relation ψ is proven to be bijective.

□

The distance on the space Sym_d^+ is computed based on the Log-Euclidean proposed for the space Sym_d^{+*}, using the function $\psi : Sym_d^+ \rightarrow Sym_d^{+*}$ defined as $\psi(\mathbf{M}) = \mathbf{M} + \epsilon \mathbf{I}_d$ for $\epsilon > 0$. Let $\mathbf{A}, \mathbf{B}$ be two elements of Sym_d^+ and let us suppose that $\mathbf{A}_1 = \mathbf{A} + \epsilon \mathbf{I}_d$ and $\mathbf{B}_1 = \mathbf{B} + \epsilon \mathbf{I}_d$. We define the Modified Log-Euclidean (MLE) distance d_{MLE} between A and B as follows:

$$\begin{aligned} d_{MLE}(\mathbf{A}, \mathbf{B}) &= \|Log(\psi(\mathbf{A})) - Log(\psi(\mathbf{B}))\|_F \\ &= \|Log(\mathbf{A} + \epsilon \mathbf{I}_d) - Log(\mathbf{B} + \epsilon \mathbf{I}_d)\|_F \\ &= \|Log(\mathbf{A}_1) - Log(\mathbf{B}_1)\|_F = d_{LE}(\mathbf{A}_1, \mathbf{B}_1) \end{aligned} \tag{5.19}$$

This extension is particularly useful for singular matrices (with a null determinant). Choosing ϵ very small compared to the eigenvalues allows the construction of a mapping function which constrains the matrices to be symmetric positive definite matrices without making them too far from their initial position. Thus, the distance is measured on the space Sym_d^{+*} and the Modified-Log-Euclidean metric inherits many properties of the Log-Euclidean distance. In the following, we show first that this measure validates all the distance conditions. Then, we show that if ϵ is chosen very small, the approximation could lead to an accurate evaluation of the distance.

Distance for Sym_d^+

Here, we show the validity of the proposed distance. Let $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ be elements of Sym_d^+. We suppose that $\mathbf{A}_1 = \mathbf{A} + \epsilon\mathbf{I}$, $\mathbf{B}_1 = \mathbf{B} + \epsilon\mathbf{I}$ and $\mathbf{C}_1 = \mathbf{C} + \epsilon\mathbf{I}$. The Modified Log-Euclidean distance d_{MLE} is proven to be a distance on Sym_d^+ because the 4 necessary conditions are respected, namely:

1) Positivity: $d_{MLE}(\mathbf{A}, \mathbf{B}) = \|Log(\mathbf{A} + \epsilon\mathbf{I}_d) - Log(\mathbf{B} + \epsilon\mathbf{I}_d)\|_F = \|Log(\mathbf{A}_1) - Log(\mathbf{B}_1)\|_F \geq 0$ because $\mathbf{A}_1, \mathbf{B}_1 \in Sym_d^{+*}$.

2) Separation: $d_{MLE}(\mathbf{A}, \mathbf{B}) = \|Log(\mathbf{A}_1) - Log(\mathbf{B}_1)\|_F \Leftrightarrow \mathbf{A}_1 = \mathbf{B}_1 \Leftrightarrow \mathbf{A} = \mathbf{B}$.

3) Symmetry: $d_{MLE}(\mathbf{B}, \mathbf{A}) = \|Log(\mathbf{B}_1) - Log(\mathbf{A}_1)\|_F = d_{LE}(\mathbf{B}_1, \mathbf{A}_1) = d_{LE}(\mathbf{A}_1, \mathbf{B}_1) = d_{MLE}(\mathbf{A}, \mathbf{B})$.

4) Triangle Inequality: $d_{LE}(\mathbf{A}_1, \mathbf{C}_1) \leq d_{LE}(\mathbf{A}_1, \mathbf{B}_1) + d_{LE}(\mathbf{B}_1, \mathbf{C}_1) \Leftrightarrow d_{MLE}(\mathbf{A}, \mathbf{C}) \leq d_{MLE}(\mathbf{A}, \mathbf{B}) + d_{MLE}(\mathbf{B}, \mathbf{C})$.

Choice of the parameter ϵ

In this subsection, we show that if ϵ is chosen very small compared to covariance matrix eigenvalues, the approximation of distance is relatively accurate even if this measure is calculated in the space of SPD matrices.

Since the matrices of the space Sym_d^+ are symmetric, they are also diagonalizable. We refer the reader to Section 5.3.3, where the calculation of the logarithm of a diagonalizable matrix is presented. Let $\mathbf{D}_M$ the diagonal matrix obtained by the diagonalization of $\mathbf{M}$ and $\mathbf{P}$ the transformation matrix satisfying $\mathbf{M} = \mathbf{P}\mathbf{D}_M\mathbf{P}^{-1}$ such as the eigenvalues in the matrix $\mathbf{D}_M$ are organized from the smallest to the highest eigenvalue (ensuring the uniqueness of P). We recall that $Log(\mathbf{M}) = \mathbf{P}Log(\mathbf{D}_M)\mathbf{P}^{-1}$. Thus, the calculation of the logarithm depends widely from the eigenvalues of $\mathbf{M}$.

Let us suppose that $\mathbf{M}_1 = \psi(\mathbf{M})$. As it belongs to Sym_d^{+*}, $\mathbf{M}_1$ is also diagonalizable. Thus, it exists a matrix $\mathbf{P}_1$ with $\mathbf{M}_1 = \mathbf{P}_1\mathbf{D}_{M_1}\mathbf{P}_1^{-1}$ such as the eigenvalues in the matrix

$\mathbf{D}_{M_1}$ are organized from the smallest to the highest eigenvalue. In fact,

$$\begin{aligned} \mathbf{D}_{\mathbf{M}_1} &= \mathbf{P}_1^{-1}\mathbf{M}_1\mathbf{P}_1 \\ &= \mathbf{P}_1^{-1}(\mathbf{M}+\epsilon\mathbf{I}_d)\mathbf{P}_1 \\ &= \mathbf{P}_1^{-1}\mathbf{M}\mathbf{P}_1+\epsilon\mathbf{I}_d \end{aligned} \tag{5.20}$$

Since $\mathbf{D}_{M_1}$ is an ordered diagonal matrix and $\mathbf{I}_d$ is an identity matrix then $\mathbf{P}_1^{-1}\mathbf{M}\mathbf{P}_1$ is a diagonal matrix containing ordered eigenvalues of M and $\mathbf{P}_1 = k\mathbf{P}$, with $k \in \mathbb{R}$. Indeed, the order ensures the uniqueness of the transformation matrix up to a scale factor.

$$\mathbf{D}_{\mathbf{M}_1} = (k\mathbf{P}^{-1})\mathbf{M}(k^{-1}\mathbf{P}) + \epsilon\mathbf{I}_d = \mathbf{P}^{-1}\mathbf{M}\mathbf{P} + \epsilon\mathbf{I}_d = \mathbf{D}_M + \epsilon\mathbf{I}_d \tag{5.21}$$

We conclude that:

$$(\lambda_1)_i = (\lambda)_i + \epsilon \tag{5.22}$$

$(\lambda)_i$ and $(\lambda_1)_i$ for $i = 1...d$ respectively represent the ordered eigenvalues of $\mathbf{M}$ and $\mathbf{M}_1$. With the analysis of this result, it can be noted that if ϵ is very small compared to $(\lambda)_i, \forall i \in [\![1, d]\!]$, the approximation is accurate enough. More details and practical experimentation concerning this parameter will be given in Section 5.7.

5.5.2 RBF-Kernel methods Symmetric Positive semi-Definite space

In this section, the RBF kernel based methods are extended to the space of SPsD matrices Sym_d^+ using the MLE distance. As the distance d_{MLE} between two matrices $\mathbf{A}, \mathbf{B}$ symmetric positive semi-definite leads to compare the distance d_{LE} between two matrices symmetric positive definite, it is easy to show that d_{MLE} is negative definite as shown for the Log-Euclidean distance in [46].

As specified in Section 5.3.2, to respect Mercer's theorem, the kernel has to be positive definite. Consequently, for an RBF kernel, the distance used should be negative definite [76]. Theorem 3.1 induces the following Corollary 5.5.1.

Corollary 5.5.1. *Let $K_G^{SPsD} : Sym_d^+ \times Sym_d^+ \to \mathbb{R}$ be a kernel such as $K_G^{SPsD}(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{d_{MLE}^2(\mathbf{x}_i,\mathbf{x}_j)}{2\sigma^2})$ and $d_{MLE}(\mathbf{x}_i, \mathbf{x}_j) = \|Log(\mathbf{x}_i + \epsilon\mathbf{I}_d) - Log(\mathbf{x}_j + \epsilon\mathbf{I}_d)\|_F$ for $\mathbf{x}_i, \mathbf{x}_j \in Sym_d^+$. Then, K_G^{SPsD} is a positive definite kernel $\forall\sigma \in \mathbb{R}$ and $\forall\epsilon > 0$.*

Proof of Corollary 5.5.1. Since $\mathbf{x}_i, \mathbf{x}_j \in Sym_d^+$ and $\epsilon > 0$, $\mathbf{x}_i + \epsilon\mathbf{I}_d$ and $\mathbf{x}_j + \epsilon\mathbf{I}_d \in Sym_d^{+*}$. Let suppose that $\mathbf{X}_i = \mathbf{x}_i + \epsilon\mathbf{I}_d$ and $\mathbf{X}_j = \mathbf{x}_j + \epsilon\mathbf{I}_d$. Therefore, $d_{MLE}(\mathbf{x}_i, \mathbf{x}_j) = \|Log(\mathbf{x}_i + \epsilon\mathbf{I}_d) - Log(\mathbf{x}_j + \epsilon\mathbf{I}_d)\|_F = \|Log(\mathbf{X}_i) - Log(\mathbf{X}_j)\|_F = d_{LE}(\mathbf{X}_i, \mathbf{X}_j)$ with $\mathbf{X}_i, \mathbf{X}_j \in Sym_d^{+*}$. Thus, $K_G^{SPsD}(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{d_{LE}^2(\mathbf{X}_i,\mathbf{X}_j)}{2\sigma^2})$ and based on Corollary 3.1, the kernel K_G^{SPsD} is positive definite. □

5.5.3 Action recognition recognition via SVM

After extracting the KC descriptors, a classification step is necessary to perform the recognition of actions. We propose to use a multi-class Support Vector Machines (SVM) as classifier which is a very popular kernel-based method. Using the extended kernel K_G^{SPsD}, the dual problem becomes:

$$
\begin{aligned}
max_\alpha \sum_{i=1}^{N_t} \alpha_i - \frac{1}{2} \sum_{i=1}^{N_t} \sum_{j=1}^{N_t} \alpha_i \alpha_j y_i y_j K_G^{SPsD}(\mathbf{KC}_i, \mathbf{KC}_j) \\
\text{with} \sum_{i=1}^{N_t} \alpha_i y_i; \alpha_i \geq 0;
\end{aligned}
\tag{5.23}
$$

$\mathbf{KC}_i$ and $\mathbf{KC}_j$ respectively represent the descriptor of the instance i, the instance j, with y_i, y_j their associated labels, N_t the number of instances used for the training and $\alpha = (\alpha_1, ..., \alpha_i, ..., \alpha_{N_t})$ the Lagrange multipliers.

The proposed approach making use of Kinematic Covariance descriptors and an extended version of SVM for SPsD matrices is illustrated in Figure 5.2. Algorithm 3 resumes how an action is recognized based on this static method.

Algorithm 7: Action recognition of an instance $\mathscr{I}$ based on the proposed static algorithm

```
Input  : skeleton sequence $(\mathbf{P_j}(t_k))_{1 \leq j \leq n, 1 \leq k \leq N}$
Output: $label$
Normalize Skeleton $(\mathbf{P_j^{norm}}(t_k))_{1 \leq j \leq n, 1 \leq k \leq N}$ (4.6)
Compute Kinematic Features $\mathbf{KF_i}(t_k))_{1 \leq i \leq M}$
Compute $(\mathbf{KC}_{\mathscr{I}})$ (5.17)
$label$ =Action recognition using the extended version of RBF-based SVM with the
    MLE distance (5.23)
```

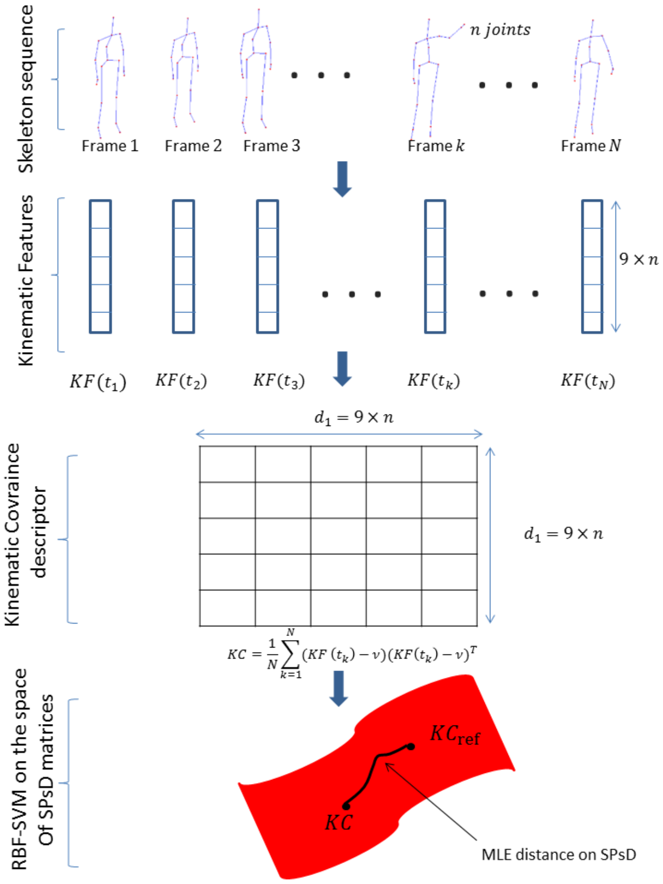


FIGURE 5.2: The proposed approach combining KC descriptors and an MLE-based SVM classifier: For every instant t_k corresponding to the time acquisition of the frame k, Kinematic Features ($KF(t_k)$) are first calculated. Then, the Kinematic Covariance descriptor is computed by integrating the KF as features in the covariance matrix. Finally, an SVM based on RBF kernel is carried out to classify actions in the space of SPsD matrices by using the MLE distance.

5.6 A dynamical approach

The limitation of KC descriptor and more generally of covariance descriptors in action recognition is mainly due to the fact that it does not contain the temporal information. Indeed, this kind of representation does not inform about the dynamical evolution over time. Thus, we propose a dynamical approach by introducing a descriptor called Hierarchical Kinematic Covariance (HKC) descriptor. More details will be given below, including the classification stage.

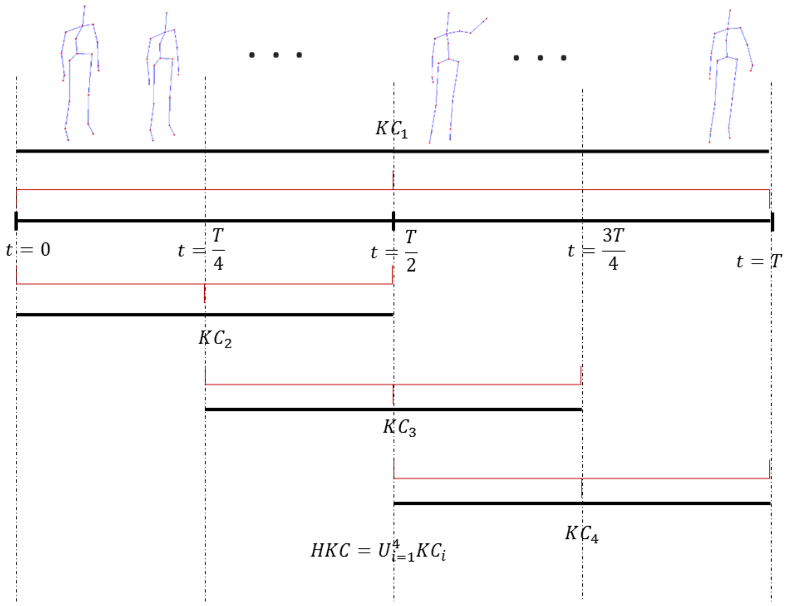


FIGURE 5.3: Computation of the Hierarchical Kinematic Covariance (HKC) descriptor as in [45]: T corresponds to the skeleton sequence length and $\mathbf{KC_1}, \mathbf{KC_2}, \mathbf{KC_3}$ and $\mathbf{KC_4}$ are the covariance matrices calculated from each corresponding range.

5.6.1 Hierarchical Kinematic Covariance descriptor

Hussein et al. [45] proposed to use a covariance matrix, containning only the joint position information, as a human action descriptor. Also, they noticed the lack of the temporal evolution. To overcome this limitation, they proposed a hierarchical covariance descriptor. This descriptor contains the concatenation of covariance descriptors calculated on 3 sub-ranges and on the whole range of the sequence. Then, to carry out the classification a linear SVM is used.

Inspired by this idea, we propose to follow it by applying it to our (KC) descriptor.

As in [45], we extract 4 Kinematic covariance matrices $(\mathbf{KC}_i)_{1 \leqslant i \leqslant 4}$ from a skeleton sequence using respectively 3 sub-ranges and the whole range of the action period. We call the set of these Kinematic Covariance matrices, the Hierarchical Kinematic Covariance (HKC) descriptor, as depicted by Equation (5.24).

$$\mathbf{HKC} = \cup_{i=1}^{4} \mathbf{KC}_i \tag{5.24}$$

Figure 5.3 illustrates how different ranges are extracted from a skeleton sequence. More precisely, the HKC descriptor represents the combination of four KC descriptors. The question that appears now, is how to classify actions using these four matrices simultaneously.

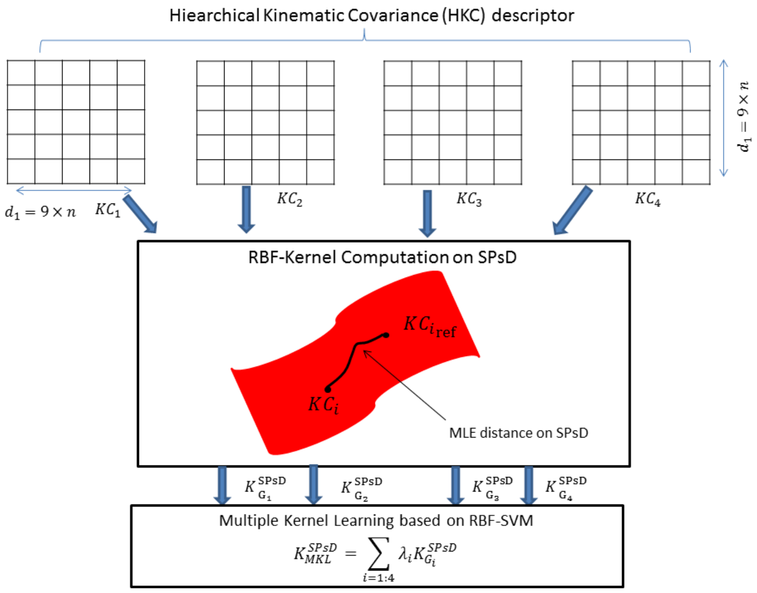

Hiearchical Kinematic Covariance (HKC) descriptor
$d_1 = 9 \times n$
$d_1 = 9 \times n$
KC_1
KC_2
KC_3
KC_4
RBF-Kernel Computation on SPsD
$KC_{i_{\mathrm{ref}}}$
KC_i
MLE distance on SPsD
$K_{G_1}^{\mathrm{SPsD}}$
$K_{G_2}^{\mathrm{SPsD}}$
$K_{G_3}^{\mathrm{SPsD}}$
$K_{G_4}^{\mathrm{SPsD}}$
Multiple Kernel Learning based on RBF-SVM
$K_{MKL}^{SPsD} = \sum_{i=1:4} \lambda_i K_{G_i}^{SPsD}$

FIGURE 5.4: The dynamical proposed approach: For every sub-range i extracted from the whole sequence, a Kinematic Covariance $\mathbf{KC}_i$ is computed. Then, based on every $\mathbf{KC}_i$, an RBF-kernel $K_G^{SPsD}(\mathbf{KC_i})$ using the MLE distance is calculated. To fuse the information of the different kernels, a Multiple Kernel Learning approach is followed using a linear combination given by the kernel denoted by K_{MKL}^{SPsD}. Finally, an SVM model is learned using K_{MKL}^{SPsD} to perform the classification.

5.6.2 Classification using a Multiple Kernel Learning (MKL) strategy

To realize the classification with the use of the HKC descriptor, we propose to make use of a Multiple Kernel Learning (MKL) strategy. Figure 5.4 illustrates the proposed approach. The idea of Multiple Kernel Learning is to use more than one kernel for learning as reflected by its name. In this work, we use a linear combination of four different kernels. For each $\mathbf{KC}_i$, an RBF-kernel based on the distance MLE is computed, that we denote by $K_G^{SPsD}(\mathbf{KC}_i)$. Thus, the final kernel used for the learning is computed as presented in Equation (5.25), where the value μ_i represents the weight attributed to the kernel $K^{SPsD}(\mathbf{KC}_i)$. The values $(\mu_i)_{1 \leq i \leq 4}$ are fixed empirically.

$$K_{MKL}^{SPsD} = \sum_{i=1}^{4} \mu_i K_G^{SPsD}(\mathbf{KC}_i) \tag{5.25}$$

Since the linear combination of positive definite matrices is a positive definite matrix, we can conclude that K_{MKL}^{SPsD} is positive definite.

Therefore, an SVM approach can be used for the learning based on the optimization of the following dual problem:

$$
\begin{gathered}
max_\alpha \sum_{i=1}^{N_t} \alpha_i - \frac{1}{2} \sum_{i=1}^{N_t} \sum_{j=1}^{N_t} \alpha_i \alpha_j y_i y_j K_{MKL}^{SPsD}(\mathbf{HKC}_i, \mathbf{HKC}_j) \\
\text{with} \sum_{i=1}^{N_t} \alpha_i y_i ; \alpha_i \geq 0;
\end{gathered}
\tag{5.26}
$$

N_t represents the number of instances used for the training, $\alpha = (\alpha_1, ..., \alpha_i, ..., \alpha_{N_t})$ are the Lagrangian multipliers and $\mathbf{HKC}_i$ and $\mathbf{HKC}_j$ respectively represent the descriptor of the instance i and the instance j, with y_i, y_j their associated labels. Algorithm 4 resumes how an action is recognized based on this dynamical approach.

Algorithm 8: Action recognition of an instance $\mathscr{I}$ based on the proposed dynamical algorithm

```
Input  : skeleton sequence $(\mathbf{P_j}(t_k))_{1\leq j\leq n, 1\leq k\leq N}$
Output: $label$
Normalize Skeleton $(\mathbf{P_j^{norm}}(t_k))_{1\leq j\leq n, 1\leq k\leq N}$ (4.6)
Compute Kinematic Features $\mathbf{KF_i}(t_k))_{1\leq i\leq M}$
Compute $(\mathbf{HKC}_{\mathscr{I})}$ (5.25)
$label$ =Action recognition using the extended version of RBF-based MKL (5.23)
```

5.7 Experiments

To validate our approach, we propose to test it (in terms of accuracy and latency) on three benchmarks of human actions, namely MSRAction3D dataset, UTKinect dataset and Multiview3D dataset.

In the rest of the paper, KC+SVM-MLE refers to the proposed static method which is the association of Kinematic Covariance descriptor with an MLE distance-based SVM, while HKC+MKL-MLE refers to the proposed dynamical method combining Hierarchical Covariance descriptor with MLE distance based MKL. Also, KSC refers to the proposed descriptor in the last chapter (KSC-NPMSE-l_2).

To overcome the orientation variability of skeletons, we apply to the data the same preprocessing of skeleton alignment presented in Section 4.4.1 to the Multiview3D dataset.

Descriptor	AS1(%)	AS2(%)	AS3(%)	Overall(%)	MET
HOG2 [63]	90.47	84.82	**98.20**	91.16	**6.44**
HON4D [65]	94.28	91.71	**98.20**	94.47	27.33
SNV [102]	**95.25**	**94.69**	96.43	**95.46**	146.57
JP [91]	82.86	68.75	83.73	78.44	0.58
RJP [91]	81.90	71.43	88.29	80.53	2.15
Q [91]	66.67	59.82	71.48	67.99	1.33
LARP [91]	83.81	84.82	92.73	87.14	17.61
KSC (ours)	83.81	87.5	97.3	89.54	0.092
KC+SVM-MLE (ours)	88.57	83.04	92.79	88.133	**0.043**
HKC+MKL-MLE (ours)	**91.42**	**92.85**	**92.79**	**92.35**	0.044

TABLE 5.1: Accuracy of recognition and Mean Execution Time per descriptor (MET)on MSRAction3D: AS1, AS2 and AS3 represent the three groups proposed in the experimentation protocol of [53]

Descriptor	Accuracy (%)	MET (s)
HOG2 [63]	74.15	5.025
SNV [102]	79.80	1365.33
HON4D [65]	90.92	25.33
Random Forest* [110]	87.90	-
LARP [91]	**97.08**	42.00
KSC (ours)	96.00	0.082
KC+SVM-MLE (ours)	90.91	**0.032**
HKC+MKL-MLE (ours)	94.95	**0.033**

TABLE 5.2: Accuracy of recognition and MET on UTKinect dataset. *The results of Random Forest have been recovered from [110] because the code is not available.

5.7.1 A trade-off between computational latency and recognition accuracy

Table 5.1, Table 5.2 and Table 5.3 respectively report the results of both proposed methods: KC+SVM-MLE and HKC+MKL-MLE by comparing them to state-of-the-art methods in terms of MET per descriptor and accuracy on MSRAction3D, UTKinect and Multiview3D datasets. According to the experimentation conducted on these three datasets, we can conclude that our descriptor (HKC+SVM-MLE) is very accurate and is also very fast to compute.

The majority of descriptors that exceeds our descriptor in terms of accuracy such as LARP on UTKinect and SNV on MSRAction3D requires a more important computational time. For example, LARP needs a mean execution time of 17.61s per descriptor on MSRAction3D dataset, 42.00 s per descriptor on UTKinect dataset and 10.51s per descriptor on

Descriptor	Same view (%)	Different view (%)	MET (s)
HOG2 [63]	87.8	74.2	9.06
HON4D [65]	89.3	76.6	17.51
SNV [102]	94.27	76.65	271.3
Actionlet* [94]	87.1	69.7	0.139
LARP [91]	96.00	88.1	10.51
KSC (ours)	90.45	90.10	0.099
KC+SVM-MLE (ours)	80.72	75.17	**0.033**
HKC+MKL-MLE (ours)	**96.17**	**93.40**	**0.035**

TABLE 5.3: Accuracy of recognition and MET using SV and DV tests on Multiview3D. *The results of Actionlet have been recovered from [41] because the code is not available.

Multiview3D dataset. This might be due to the high number of approximation and calculation required by this method.

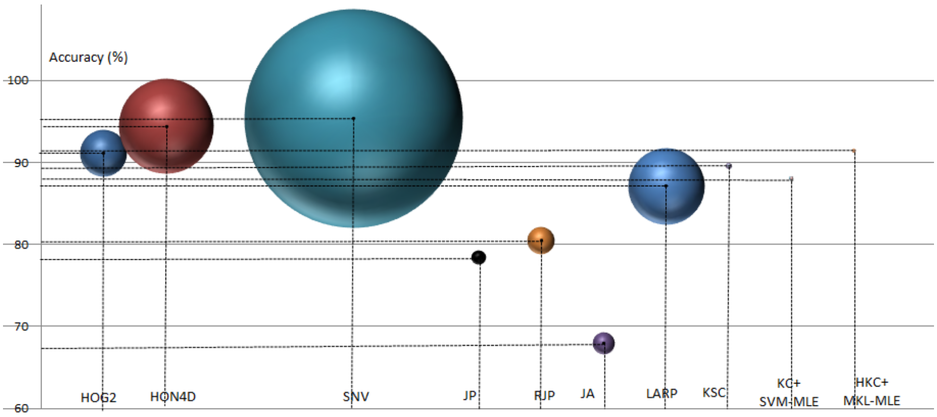
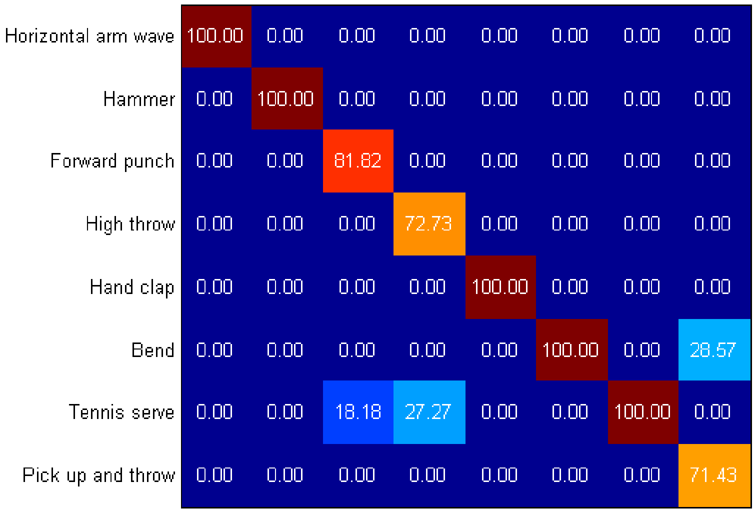


FIGURE 5.5: Illustration of the recognition accuracy and the MET of KSC, KC, HKC descriptors and state-of-the-art-descriptors: As described in previous chapters, the center of the balls represents the accuracy of every method and the surface area its MET per descriptor.

The first descriptors HOG2 [63], HON4D [65] and SNV[102] which represent depth-based descriptors are very accurate according to the recognition accuracy results obtained on MSRAction3D. However, as remarked in [37], they are greedy in terms of computational time since the dimension of depth images is more important on the three datasets. Furthermore, HOG2 [63] and SNV [102] give very low accuracy on UTKinect dataset. That could be explained by the fact that UTKinect dataset contains some videos with a very small number of frames.

It can be noted that compared to skeleton representations, the Hierarchical Kinematic Covariance (HKC) descriptor combined with an MLE-distance based MKL presents the most



Horizontal arm wave	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Hammer	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
Forward punch	0.00	0.00	81.82	0.00	0.00	0.00	0.00	0.00
High throw	0.00	0.00	0.00	72.73	0.00	0.00	0.00	0.00
Hand clap	0.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00
Bend	0.00	0.00	0.00	0.00	0.00	100.00	0.00	28.57
Tennis serve	0.00	0.00	18.18	27.27	0.00	0.00	100.00	0.00
Pick up and throw	0.00	0.00	0.00	0.00	0.00	0.00	0.00	71.43

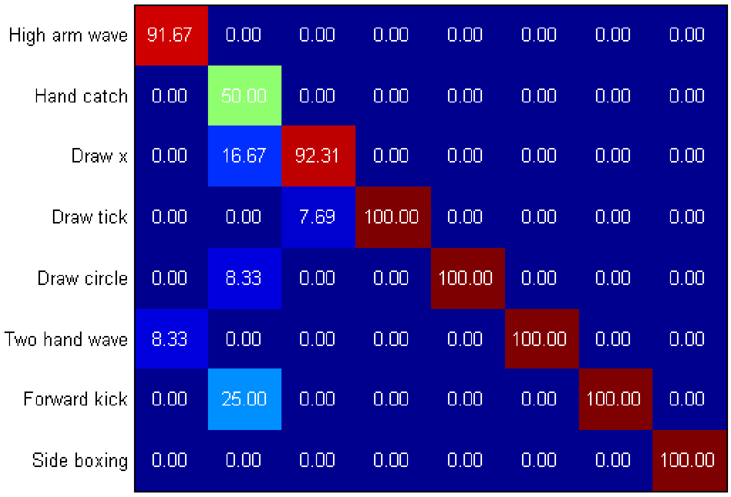
FIGURE 5.6: Confusion matrix obtained using the approach HKC+MKL-MLE on the group AS1 of the dataset MSRAction3D

accurate recognition on MSRAction3D and Multiview3D datasets. Moreover, it presents one of the lowest MET per descriptor with 0.044s per descriptor (after Kinematic Covariance descriptor with 0.043s per descriptor). Even if our method does not present the best results in terms of accuracy on UTKinect dataset with 94.95 % (against 97.08% for LARP and 95% for our descriptor KSC), it remains accurate and its low computational latency with an MET per descriptor which is equal to 0.033s represents a very motivating result, since it realizes a nice trade-off between latency and accuracy.

To illustrate simultaneously the information of accuracy and MET per descriptor, we propose the representation of the results in Figure 5.5. Every ball corresponds to a method making use of a specific descriptor. The surface area of the ball represents the MET, while the center of the ball corresponds to the recognition accuracy. It is easy to notice that our method (HKC+MKL-MLE) presents one of the best trade-off between latency and accuracy on MSRAction3D.

Although the use of the skeleton needs pre-processing, skeleton modality remains more suitable for fast recognition. Indeed, according to [68], skeleton extraction process takes around 45ms per frame. For an action of 30 frames (very reasonable length for an action), the time needed to extract a skeleton sequence is equal to nearly 1,35 s.

Figure 5.6, Figure 5.7, Figure 5.8 and Figure 5.9 respectively present the different confusion matrices provided by the groups AS1, AS2 and AS3 of MSRAction3D dataset and



High arm wave	91.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Hand catch	0.00	50.00	0.00	0.00	0.00	0.00	0.00	0.00
Draw x	0.00	16.67	92.31	0.00	0.00	0.00	0.00	0.00
Draw tick	0.00	0.00	7.69	100.00	0.00	0.00	0.00	0.00
Draw circle	0.00	8.33	0.00	0.00	100.00	0.00	0.00	0.00
Two hand wave	8.33	0.00	0.00	0.00	0.00	100.00	0.00	0.00
Forward kick	0.00	25.00	0.00	0.00	0.00	0.00	100.00	0.00
Side boxing	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00

FIGURE 5.7: Confusion matrix obtained using the approach HKC+MKL-MLE on the group AS2 of the dataset MSRAction3D

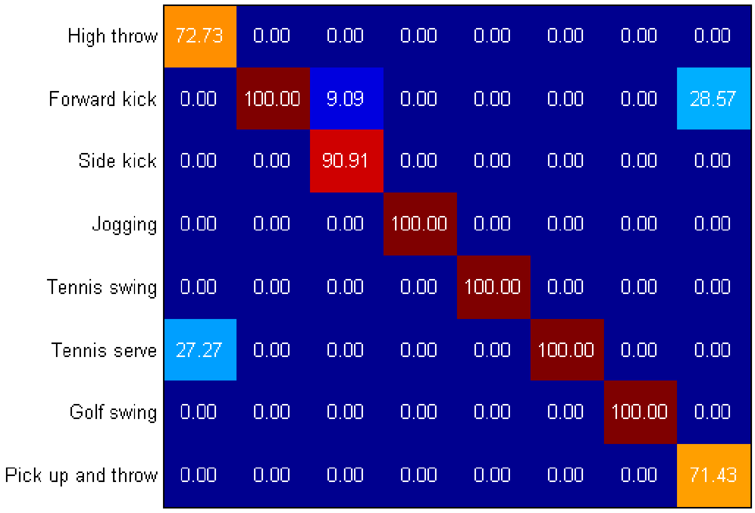
UTKinect dataset. It can be noted that the majority of actions are perfectly recognized as for *Bend* or *Hand Clap* on the group AS1 of MSRAction3D dataset (see Figure 5.6).

The problem of wrong recognition occurs mostly with very close actions. For instance, the two actions *High throw* and *Tennis serve* are very similar: they induce the use of same human joints in the same way. As shown in Figure 5.8, it results from that an important confusion. This remark can also be noted on the dataset UTKinect: in Figure 5.9, the actions *push* and *throw* are sometimes confused. The same happens for the actions *stand up* and *pick up*.

We recall that in our study, we only use an action descriptor related to human joints, since we aim just at analyzing the motion. Nevertheless, including object descriptor further to the action descriptors can be necessary in these cases.

5.7.2 Robustness to viewpoint changes

Table 5.3 which reports the obtained results on Multiview3D dataset shows the robustness of our method (HKC+MKL) to viewpoint changes compared to other methods. Although KSC looks less sensitive to viewpoint variation and is very robust to the change of orientation as demonstrated in Chapter 4, it can be noted that the proposed approach presents the best results in terms of accuracy for both Same View (SV) and Different Views (DV) tests.



High throw	72.73	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Forward kick	0.00	100.00	9.09	0.00	0.00	0.00	0.00	28.57
Side kick	0.00	0.00	90.91	0.00	0.00	0.00	0.00	0.00
Jogging	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00
Tennis swing	0.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00
Tennis serve	27.27	0.00	0.00	0.00	0.00	100.00	0.00	0.00
Golf swing	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00
Pick up and throw	0.00	0.00	0.00	0.00	0.00	0.00	0.00	71.43

FIGURE 5.8: Confusion matrix obtained using the approach HKC+MKL-MLE on the group AS3 of the dataset MSRAction3D

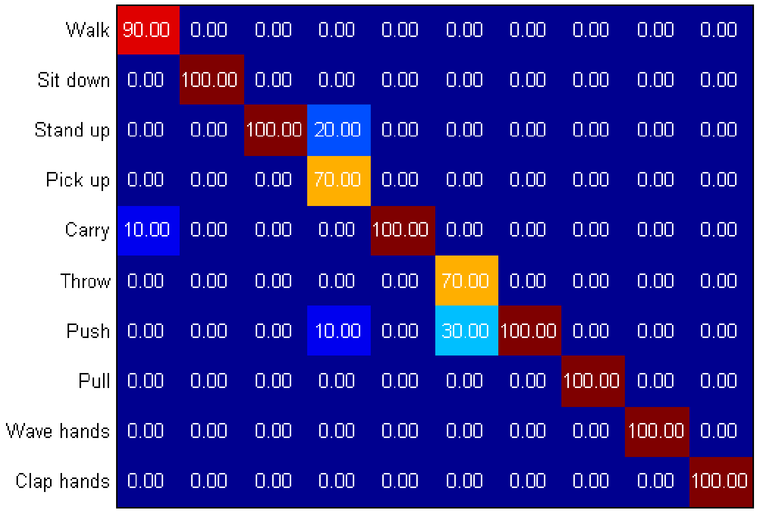
Indeed, HKC combined with MKL-MLE gives 96.17% for data with Same View (SV) and 93.40% for data with Different Views (DV). Moreover, the differences between the accuracies obtained for SV and DV tests are the lowest one after our descriptor KSC (less than 3% of differences for HKC+MKL).

Table 5.4 details the different SV and DV tests. It can be noted that even if SV tests present a global better accuracy, the accuracy registered for the different SV and DV tests belong to almost the same range of values (between 88.54% to 98.96%).

5.7.3 Role of Kinematic Features

To analyze the role of Kinematic Features (KF), we propose to perform the following experimentation. The same descriptor is built by removing every time a kinematic component. The results are reported in Table 5.5.

In the first column, the different kinematic components used to build the descriptor are specified, with the knowledge that P, V and A respectively refer to position, velocity and acceleration. All the experiments (except for the results found on UTKinect dataset) show that the use of the three kinematic component provides the best results. For UTKinect dataset, the best results are provided when only the position and the velocity are used. This might be



Walk	90.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Sit down	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Stand up	0.00	0.00	100.00	20.00	0.00	0.00	0.00	0.00	0.00	0.00
Pick up	0.00	0.00	0.00	70.00	0.00	0.00	0.00	0.00	0.00	0.00
Carry	10.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00
Throw	0.00	0.00	0.00	0.00	0.00	70.00	0.00	0.00	0.00	0.00
Push	0.00	0.00	0.00	10.00	0.00	30.00	100.00	0.00	0.00	0.00
Pull	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00	0.00
Wave hands	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00
Clap hands	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00

FIGURE 5.9: Confusion matrix obtained using the approach HKC+MKL-MLE on the dataset UTKinect

caused by the fact that the error is increasing with derivative calculation and that UTKinect dataset is more noisy than other datasets. For this reason, on this dataset, only these two kinematic values are integrated in KF.

orientation	0°	30°	−30°
0°	98.96	90.63	90.63
30°	94.79	89.58	88.54
-30°	95.83	88.54	92.71

orientation	0°	30°	−30°
0	98.96	96.88	94.79
30	96.88	97.92	96.88
-30°	95.83	90.63	98.96

TABLE 5.4: Accuracy of every test on Multiview3D dataset using HKC+MKL-MLE approach: We detail here the accuracy obtained for every test. The table on the left represents the results obtained when the training data are performed by subjects 1,2,3 and 4. The table on the right represents the results obtained when the training data are performed by subjects 5,6,7 and 8. The orientation specified in the columns represents the orientation of the data used for training, while the orientation specified in the lines represents the orientation of the data used for testing

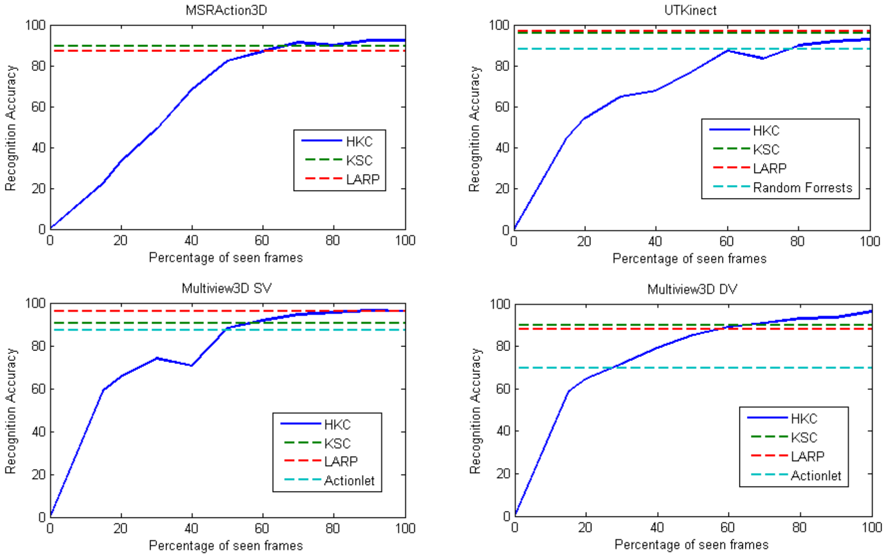


FIGURE 5.10: Accuracy of recognition on different datasets according to the percentage of seen frames

Kinematics	MSRAction3D (%)	UTKinect (%)	Multiview3D SV (%)	Multiview3D DV (%)
P+V+A	**92.35**	92.93	**96.18**	**93.40**
P+V	91.76	**94.95**	95.31	94.27
P	77.53	92.93	95.48	93.66
V	78.08	89.90	91.15	88.45
A	85.13	82.83	85.59	80.03

TABLE 5.5: Effect of each kinematic component on the accuracy of recognition using HKC+MKL-MLE

Deleted process(%)	MSRAction3D	UTKinect (%)	Multiview3D SV (%)	Multiview3D DV (%)
Nothing	**92.35**	**94.95**	**96.18**	**93.40**
SN	84.32	92.93	90.62	82.37

TABLE 5.6: Effect of each kinematic component on the accuracy of recognition using HKC+MKL-MLE

5.7.4 Skeleton Normalization Interest

Table 5.6 reports the benefits of the used Skeleton Normalization which plays the role of reducing the negative effect due to anthropometric variability on the recognition accuracy (as in Chapter 4). We can see that without the use of Skeleton Normalization, the accuracy decreases by 8% on MSRAction3D, by 2% on UTKinect, by 6% on Multiview3D for SV tests and by 11% on Multiview3D for DV tests.

5.7.5 Observational latency

The observational latency is also an important criterion for online action recognition. For this reason, we propose to carry out the following experiments.

The idea is to report the accuracy using only a specific percentage of seen frames as proposed in [105]. Figure 5.10 illustrates the recognition accuracy according to the percentage of observed frames on MSRAction3D, on UTKinect, on Multiview3D dataset for SV tests and Multiview3D dataset for DV tests. The dotted lines indicate that the corresponding methods have only been tested by using the whole sequence. For MSRAction3D dataset, it can be noted that starting from 50% of seen frames, the accuracy exceeds 80% and that starting from 70% of seen frames, our descriptor registers a better score than KSC and LARP. Also, on UTKinect, after 50% of observed frames, the score is superior to 80% and after 60% of seen frames, the accuracy is more or less stable. In the same way, the accuracy is more or less stable starting from 50% of seen frames on Multiview3D dataset for both SV and DV tests.

5.7.6 Parameter Analysis

The parameter ϵ. As mentioned previously, the parameter ϵ should be meticulously chosen. According to our previous analysis in Section 5.5.1, it is preferable that ϵ be smaller than covariance eigenvalues. In practical, the problem is that if we choose ϵ too small, it is considered null. To study the effect of varying ϵ, we propose the illustration of Figure 5.11,

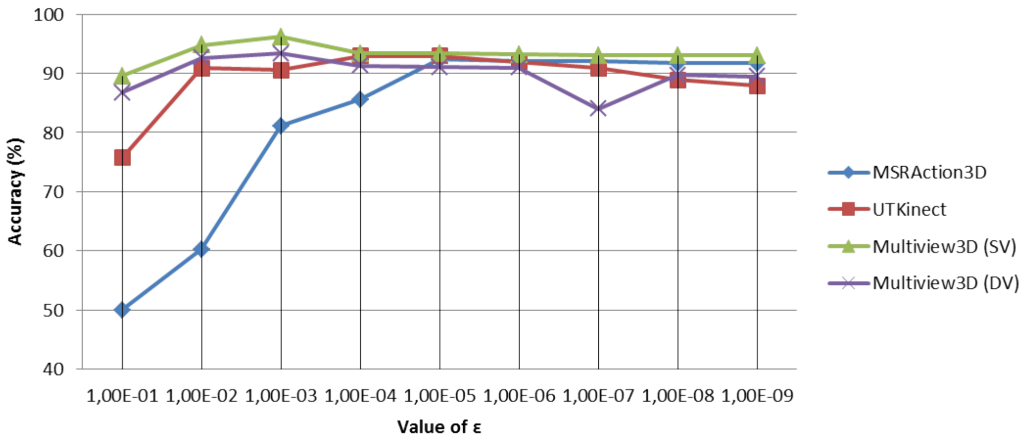


FIGURE 5.11: Threshold influence on the accuracy of three benchmarks

which reports the accuracy according to the chosen value for ϵ. The graph shows that by choosing $\epsilon \leq 10^{-5}$, the accuracy does not vary importantly. However, the highest accuracy has been respectively registered for $\epsilon_{MSR} = 10^{-5}$, $\epsilon_{UT} = 10^{-4}$ and $\epsilon_{UT} = 10^{-3}$ on MSRAction3D, UTKinect and Multiview3D datasets.

The parameters μ_i. We recall that these parameters weight the contribution of each kernel used in the MKL approach and weight therefore the contribution of each covariance matrix. In all experiments, we found empirically the best combination $\mu = (\mu_1, \mu_2, \mu_3, \mu_4)$. Nevertheless, the experiments have shown that even with varying μ randomly the accuracy does not decrease significantly, with a maximum of 5%.

5.8 Conclusion

In this chapter, the extension of RBF-kernel methods has been applied to the case of action recognition. To do that, a static approach has been first proposed by introducing a novel descriptor called Kinematic Covariance. However, this static approach is limited by the lack of temporal ordering information. For this reason, a dynamical approach has been also proposed consisting on the combination of a Hierarchical Covariance descriptor (HKC) and an MKL strategy. The experimentation outlines the very good properties of this method which is accurate, fast to compute and presents a low observational latency, etc. The most interesting point in this method is that it can be extended to the online case. As shown in [51], there is an incremental relation between a covariance descriptor at an instant t and at an instant $t + 1$. With the use of a sliding window, it is very convenient to extend a covariance based action recognition method to the online case. Moreover, the low computational latency and observational latency required for the method running makes possible the conception of a true real-time action recognition system. However, some improvements can be done, since the parameters ϵ and μ_i have been fixed in an empirical way.

After presenting the works realized during this thesis, the synthesis of this dissertation as well as perspectives are presented in Chapter 6.

Chapter 6

Conclusion

This chapter is dedicated to summarizing this PhD thesis and to presenting interesting future directions. Firstly, Section 6.1 presents the scientific findings by outlining the goals which have been achieved in this thesis. Secondly, Section 6.2 discusses possible future research works that we have already started to explore.

6.1 Summary

The goal of this thesis was to explore the issue of low-latency human action recognition using RGB-D cameras, by reducing the latency and conserving a good accuracy. As the latency is defined as the sum of computational latency and observational latency, we have respectively divided this issue in two parts: fast action recognition and online action recognition.

Thus, the beginning of this thesis is devoted to the task of fast human action recognition. In order to compare different existing descriptors, we proposed a comparative study by reporting the accuracy as well as the Mean Execution Time (MET) per descriptor provided by different methods. From this comparison, two main conclusions have been drawn. First, these experiments highlighted the differences existing between skeleton-based descriptors and depth-based descriptors. While depth-based are sometimes more accurate, skeleton-based descriptors present generally lower computational latency, are more robust to viewpoint variations and realize a better trade-off between accuracy and computational latency. Second, it has been noted from this comparative study that even accurate skeleton-based descriptors present sometimes a considerable execution time due mainly to approximation and complex mathematical models. The first conclusion encouraged us to choose the skeleton sequence modality that is considered more suitable for low-latency action recognition, while the second conclusion prompted us to design a novel descriptor realizing a trade-off between accuracy and computational latency by using non complex models.

Hence, we proposed a fast, accurate and invariant human action recognition descriptor called Kinematic Spline Curves (KSC). To ensure low computational latency, the different algorithms made to build this descriptor have been carefully chosen. This feature vector is calculated based on the spline interpolation of Kinematic Features. Also, we introduced a fast temporal normalization called TVR since the execution rate variability represents one of the most important challenge in action recognition. Simple skeleton normalization and alignment have been proposed in order to overcome the anthropometric and the viewpoint variability. We have shown the effectiveness of this descriptor combined with a linear SVM classifier in terms of accuracy and computational latency compared to state-of-the-art methods. In this way, this proposed method can be used in off-line applications requiring a fast answer such as rehabilitation, coaching and medical assistance. However, this method remains hardly adaptable to online scenarios, as the temporal normalization TVR requires the a priori knowledge of the whole sequence (to calculate the normalized energy NAE or NPMSE), as depicted in Chapter 4.

As we are also interested by online scenarios, we proposed a second descriptor, called Hierarchical Kinematic Covariance (HKC) which presents a low observational latency, further to low computational latency and good accuracy. Since covariance matrices have been proven to be efficient in online applications thanks to its fast calculation and its possible incremental calculation, we think that this descriptor, which integrates Kinematic Features in a covariance matrix, can be easily applied to online scenarios. As we show that Kinematic Covariance matrices are elements of the space of SPsD matrices, we proposed to extend the RBF-kernel-based classification methods to this space using a new MLE distance. Thus, the classification is done thanks to an SVM-based MKL using the extended approach. The experimental study has shown the low-latency (in terms of observational anf computational latency) and the accuracy of HKC descriptor combined with the cited classification method.

6.2 Future work

Thanks to the work done during this thesis, we could propose many interesting perspectives for future works such as continuous action recognition, fast activity recognition, action

Descriptor	Accuracy (%)	MET (s)
AOG [93]	53.8	-
HON4D[65]	80.00	-
SNV [102]	86.40	-
Holistic HOPC[71]	88.80	-
KSC (ours)	53.13	0.18
HKC (ours)	50.63	0.08

TABLE 6.1: Accuracy of recognition and MET on MSRDailyActivity3D dataset

recognition from a multi-camera system and incremental learning of human actions. We detail these prospects below.

6.2.1 Continuous action recognition

Since the proposed method in Chapter 5 using the Hierarchical Kinematic Covariance (HKC) descriptor has shown its low latency in terms of observational latency and computational latency and since covariance matrices can be incrementally calculated, this work should be suitable for continuous action recognition. The task of continuous action recognition consists in recognizing actions from unsegmented videos (the beginning and the end of actions are not known) in real-time. The first track that we are planning to explore is the extension of our method with the use of a sliding window or multiple sliding windows.

6.2.2 Fast activity recognition

Both descriptors that have been proposed in this work (KSC and HKC) are adapted to fast action recognition. However, in this thesis we did not focus on the activity recognition which implies the interaction of humans with objects. To test the performance of the proposed methods for activity recognition, we propose to use the dataset MSRDailyActivity3D proposed in [94].

MSRDailyActivity3D is a dataset composed of 16 activities: *drink, eat, read book, call cellphone,write on a paper, use laptop, use vacuum cleaner, cheer up, sit still, toss paper, play game, lay down on sofa, walk, play guitar, stand up, sit down*. They are performed two times by 10 subjects, once sitting and once standing. We follow the same experimental settings proposed by Rahmani et al. [71], where the activities performed by half of the subject are used for the training and the rest are used for the testing. The obtained accuracy and MET on this dataset for KSC and HKC are reported in Table 6.1.

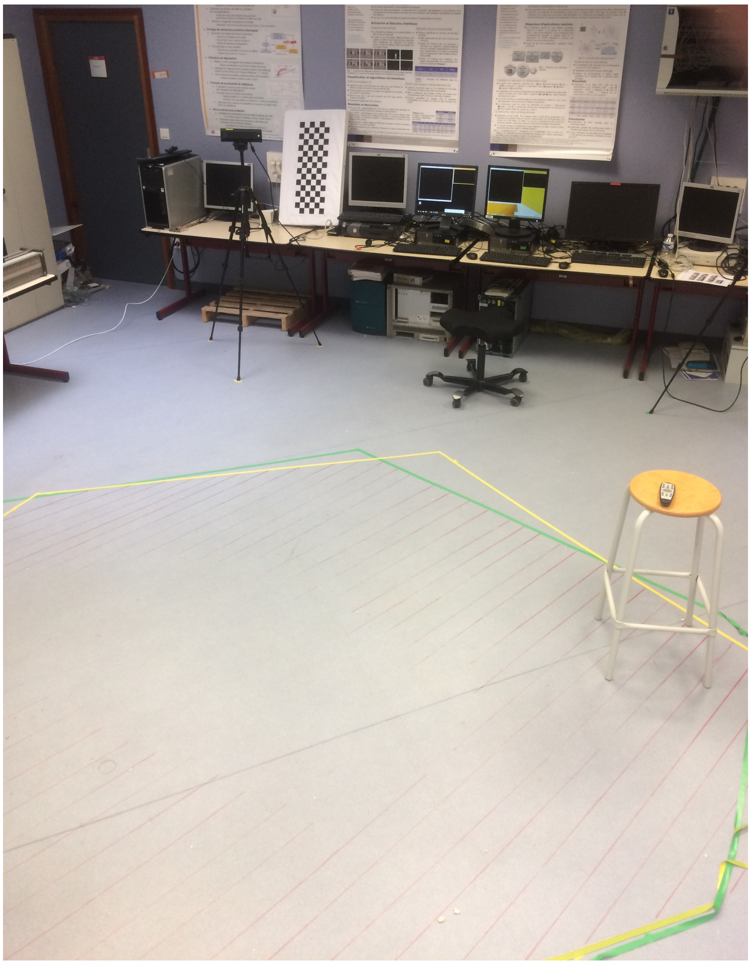


FIGURE 6.1: The experimental conditions of the dataset acquisition: the green surface represents the limits of the scene observed by the first camera, while the yellow surface represents the limits of the scene observed by the second camera.

The results obtained prove that KSC and HKC are not efficient for activity recognition. This could be explained by the fact that only the human skeleton is used to build these descriptors. To extend them to activity recognition, we propose in a future work to associate them to an object descriptor extracted from depth images, with the respect of the time constraints.

6.2.3 Action recognition from a multi-camera system

The majority of state-of-the-art methods has proposed approaches using only one RGB-D camera. In future works, we propose to study the issue of action recognition using multiple RGB-D cameras.

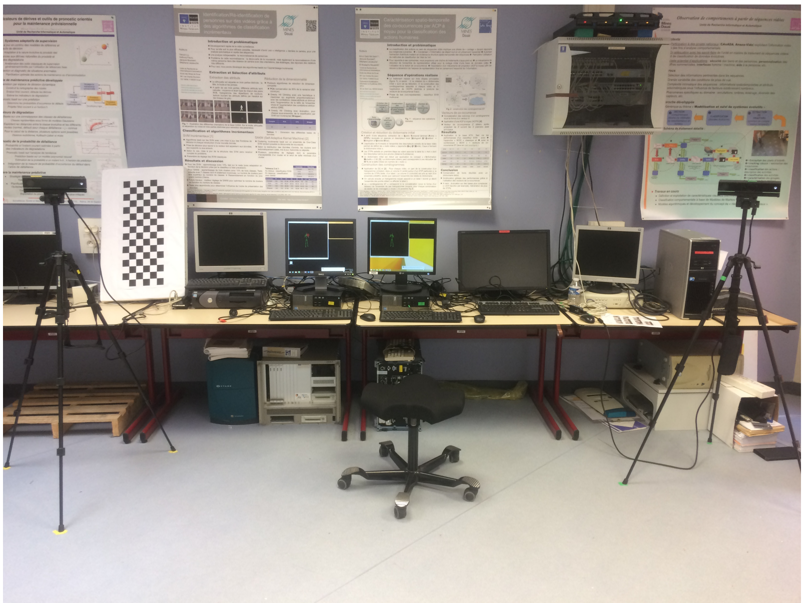


FIGURE 6.2: The RGB-D cameras positioning during the dataset acquisition: it can be noted the presnece of two RGB-D cameras (Kinect 2) with different viewpoints

For this reason, a dataset has been recently introduced in the IA department of IMT Lille Douai, making use of two synchronized RGB-D cameras with two different viewpoints. This dataset has been called Multiview 3D Human Object Interaction dataset. It includes 8 activities: *press button, pick phone, use remote and take it back, fetch water from dispenser, walk around holding cane, walk around holding umbrella, remove chair, walk with plate and put it back on the table*. 10 different subjects have repeated 2 times these activities once by behaving normally, once by simulating an injured leg and once by simulating an injured arm. Figure 6.1 and Figure 6.2 illustrate the experimental conditions.

By using this dataset, we are excepting by proposing an algorithm suitable for Assisted Ambient Living (AAL) applications.

6.2.4 Incremental learning of human actions

One of the most challenging issue in pattern recognition and machine learning is the acquisition of data in order to estimate an accurate model. To progressively adapt a classification model by including new data, many scientists propose to use incremental learning. Contrary to batch classification which needs to remake the whole classification model estimation in the presence of new data, incremental learning updates the classification model by adjusting its parameters. The main advantage of incremental learning is its rapidity and the possibility of rejecting biased data. In future works, we propose to study the extension of the RBF-based SVM for SPsD matrices from the batch to an incremental version.

Appendix A

Appendix: Classification via Linear Support Vector Machine

We propose to recall in this appendix the main principle of SVM. This classification method belongs to distance based approaches. It has been introduced in 1997 by Vapnik [90]. Because its efficiency (even with a restricted number of training data) and its relative simplicity, this method is still attracting the interest of the machine learning community. First, the basic idea of linear binary SVM is presented. Then, the extension of SVM to multi-label classification is recalled. More details about non linear SVM can be found in Chapter 6.

A.0.1 Binary classification via Linear Support Vector Machines

Linearly separable data

As presented in Chapter 2, feature description allows placing data in a same feature space denoted by S, characterized by a vector space structure. Data are said linearly separable when it is possible to completely separate data belonging to different classes using only hyperplanes of S. In Figure A.1, where data of two classes (Blue and Red) are illustrated, it is possible to observe an example of linearly separable data. In the following, we suppose that we are working in the case of a binary classification.

The classification using SVM is based on the estimation of a hyperplane which separates the training data according their label and maximizes the margin (the distance between the hyperplane and the closest samples). The training data are represented by N couples $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$ with $\mathbf{x}_i \in \mathbb{R}^p$ represents the feature vector and $y_i \in \{-1, 1\}$

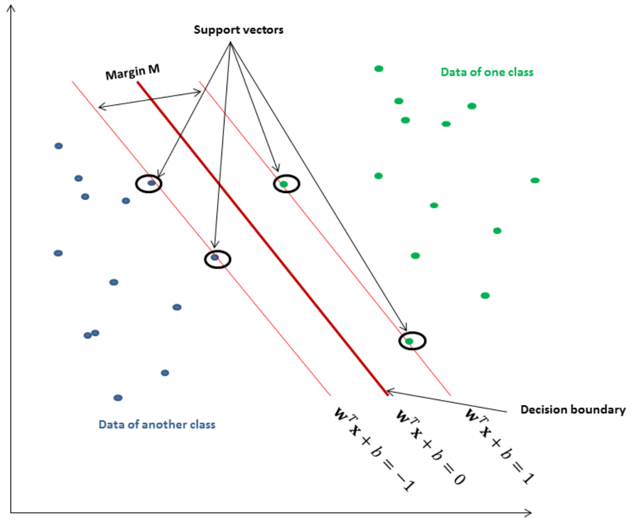


FIGURE A.1: Classification via SVM of linearly separable data: The line in red bold is the estimated hyperplane which separates the data into two distinct classes (red and blue). In this case, a linear classification is possible thanks to the particular distribution of data in the feature space.

represents the label. Figure A.1 illustrates this method of classification in a 2D vector space.

The equation of the hyperplane H is defined by Equation (A.1),

$$H : f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \mathbf{b} = 0 \tag{A.1}$$

where $\mathbf{w}^T = (w_1, w_2, ..., w_p)^T$ represents the normal vector of the hyperplane H, and $\mathbf{b}$ represents the bias. Therefore, the rule of classification is defined thanks to the sign of the hyperplane function. The label $G(\mathbf{x})$ of a testing data $\mathbf{x}$ is deduced in accordance with Equation (A.2).

$$G(\mathbf{x}) = sign(f(\mathbf{x})) \tag{A.2}$$

Hence, it leads to maximize the margin M defined by Equation (A.3)

$$\begin{aligned} M &= \min_{\mathbf{x}\in cl_1} d(H,\mathbf{x}) + \min_{\mathbf{x}\in cl_{-1}} d(H,\mathbf{x}) \\ &= \min_{x\in cl_1} \frac{|\mathbf{w}^T\mathbf{x}+\mathbf{b}|}{\|\mathbf{w}\|} + \min_{x\ \in cl-1} \frac{|\mathbf{w}^T\mathbf{x}+\mathbf{b}|}{\|\mathbf{w}\|} \\ &= \frac{1}{\|\mathbf{w}\|} + \frac{1}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \end{aligned} \tag{A.3}$$

We denote respectively by cl_1 and cl_{-1} the positive and the negative classes.

To find a unique solution $(\mathbf{w},\mathbf{b})$, only the canonical hyperplanes are taken into account and are thus constrained by Equation (A.4).

$$\min_i |\mathbf{w}^T\mathbf{x}_i + \mathbf{b}| = 1 \tag{A.4}$$

This algorithm can be seen as an optimization problem with the goal of minimizing the norm $\|\mathbf{w}\|$. To obtain a quadratic formulation, the optimization problem is written as follows,

$$\min_{\mathbf{w},\mathbf{b}} \frac{1}{2}\|\mathbf{w}\|^2 \tag{A.5a}$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) \geq 1,\ \ i = 1,...,N \tag{A.5b}$$

To solve this constrained convex optimization problem the Lagrangian is used as depicted by Equation (A.6),

$$L(\mathbf{w},\mathbf{b},\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N}(\alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) - 1)) \tag{A.6}$$

with $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_N]$ Lagrange multipliers. At the minimum, the partial derivatives of the Lagrangian with respect to the primary variables are null, as depicted below.

$$\frac{\delta L}{\delta \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \tag{A.7a}$$

$$\frac{\delta L}{\delta \mathbf{b}} = 0 \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{A.7b}$$

The conditions of Karush-Khun-Tucker show that only the points belonging to the border hyperplans F_1 and F_{-1} (that constitute the margin and which respect the equation $|\mathbf{w}^T\mathbf{x} + \mathbf{b}| = 1$) play a role in the estimation of the main hyperlane (See Figure A.1). Thus, only the Lagrange multipliers of these points are not null. In other words, if $\mathbf{x}_i \in F_1 \Rightarrow \alpha_i \neq 0$, otherwise $\alpha_i = 0$. The points with $\alpha_i \neq 0$ are called support vectors.

Using only support vectors, the problem can be expressed by Equation (A.8).

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{A.8a}$$

$$\text{subject to } \sum_{i=1}^{N} \alpha_i y_i \alpha_i \geq 0 \text{and} \alpha_i \geq 0 \tag{A.8b}$$

$\langle ., . \rangle$ represents the scalar product of the vector space S. The optimal Lagrange $\alpha^{opt} = (\alpha_1^{opt}, \alpha_2^{opt}, ..., \alpha_N^{opt})$ is obtained by the resolution of the system equation. The calculation of the optimal couple $(\mathbf{w}, \mathbf{b})$ is therefore done by the introduction of α^{opt} in Equation (A.9).

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i^{opt} y_i \mathbf{x}_i \tag{A.9a}$$

$$\mathbf{b} = -\frac{1}{2} \mathbf{w}^T (\mathbf{x}_{i_1^*} + \mathbf{x}_{i_{-1}^*}) \tag{A.9b}$$

$\mathbf{x}_{i_1^*}$ and $\mathbf{x}_{i_{-1}^*}$ respectively represent the support vectors of the regions equipped with the label 1 and -1.

Soft Margin

When the data are not perfectly linearly separable by a hyperplane, a simple method relaxes the constraints by adding simple errors ϵ_i (controlling the overruns) [18]. If the data are not linearly separable, the optimal hyperplane is the one who separates the data with a minimum of errors. If the error $\epsilon_i > 1$, the answer is considered false. This idea aims to maximize the margin when the data are well classified and to minimize it when they are misclassified. Thus, the problem could be written as in Equation (A.10).

$$\begin{gathered}\max_{\mathbf{w},\mathbf{b}} \frac{1}{2}\|\mathbf{w}\|^2 \sum_{j=1}^{N} + C \sum_{i=1}^{N} \epsilon_i \\ \text{with } y_i(\mathbf{w}^T\mathbf{x}_i + \mathbf{b}) > 1 - \epsilon_i,\ i = 1...N\end{gathered} \tag{A.10}$$

C represents a parameter which weights the errors and is called regularization term. In Section 5.3.2, non linear SVM will be presented, since they represent an important part in our work.

A.0.2 Multi-class SVM strategies

As demonstrated in Section 3.4.1, SVM has been initially designed for binary classification. Principally, two strategies have been proposed to extend SVM to multi-label classification. In what follows, we present an overview of these different strategies. Let us suppose that the number of classes is equal to k.

One-against-All

This approach represents one of the earliest and most intuitive extension. It consists in learning k binary SVM models (one model for each class). Similarly to the name of this strategy, for every model i, the training data belonging to the class i are associated to positive labels, while the rest of the data are associated to negative labels. That leads to approximating k hyperplanes. During the testing phase, all the estimated classifiers are used and the affected label corresponds to the index of the higher decision function.

One-against-One

The idea of this approach is to estimate an SVM classifier between each couple of classes. It results from that the learning of $\frac{k(k-1)}{2}$ binary classifiers. All the classifiers are used during the testing phase. The label is attributed by following a voting approach.

Bibliography

[1] Mohamed Abdur Rahman, Ahmad M Qamar, Mohamed A Ahmed, M Ataur Rahman, and Saleh Basalamah. “Multimedia interactive therapy environment for children having physical disabilities”. In: *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval*. ACM. 2013, pp. 313–314.

[2] Jake K Aggarwal and Michael S Ryoo. “Human activity analysis: A review”. In: *ACM Computing Surveys (CSUR)* 43.3 (2011), p. 16.

[3] Jake K Aggarwal and Lu Xia. “Human activity recognition from 3d data: A review”. In: *Pattern Recognition Letters* 48 (2014), pp. 70–80.

[4] Vincent Arsigny, Pierre Fillard, Xavier Pennec, and Nicholas Ayache. “Geometric means in a novel vector space structure on symmetric positive-definite matrices”. In: *SIAM journal on matrix analysis and applications* 29.1 (2007), pp. 328–347.

[5] Vincent Arsigny, Pierre Fillard, Xavier Pennec, and Nicholas Ayache. “Log-Euclidean metrics for fast and simple calculus on diffusion tensors”. In: *Magnetic resonance in medicine* 56.2 (2006), pp. 411–421.

[6] Serge Belongie, Kristin Branson, Piotr Dollar, and Vincent Rabaud. “Monitoring animal behavior in the smart vivarium”. In: *Measuring Behavior*. Wageningen The Netherlands. 2005, pp. 70–72.

[7] Vinay Bettadapura, Grant Schindler, Thomas Plötz, and Irfan Essa. “Augmenting bag-of-words: Data-driven discovery of temporal and structural information for activity recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 2619–2626.

[8] Victoria Bloom, Dimitrios Makris, and Vasileios Argyriou. “G3d: A gaming action dataset and real time action recognition evaluation framework”. In: *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE. 2012, pp. 7–12.

[9] Enrico Bondi, Lorenzo Seidenari, Andrew D Bagdanov, and Alberto Del Bimbo. “Real-time people counting from depth imagery of crowded environments”. In: *Advanced Video and Signal Based Surveillance (AVSS), 2014 11th IEEE International Conference on*. IEEE. 2014, pp. 337–342.

[10] Khaled Boukharouba, Laurent Bako, and Stéphane Lecoeuche. “Temporal video segmentation using a switched affine models identification technique”. In: *Image Processing Theory Tools and Applications (IPTA), 2010 2nd International Conference on*. 2010, pp. 157–160.

[11] Nicolas Bourbaki. *Lie groups and Lie algebras: chapters 7-9*. Vol. 7. Springer Science & Business Media, 2008.

[12] Luc Brun, Gennaro Percannella, Alessia Saggese, and Mario Vento. “Action recognition by using kernels on aclets sequences”. In: *Computer Vision and Image Understanding* 144 (2016), pp. 3–13.

[13] Xingyang Cai, Wengang Zhou, Lei Wu, Jiebo Luo, and Houqiang Li. “Effective active skeleton representation for low latency human action recognition”. In: *IEEE Transactions on Multimedia* 18.2 (2016), pp. 141–154.

[14] Chih-Chung Chang and Chih-Jen Lin. “LIBSVM: A library for Support Vector Machines”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 2.3 (2011), p. 27.

[15] Yao-Jen Chang, Shu-Fang Chen, and Jun-Da Huang. “A Kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities”. In: *Research in developmental disabilities* 32.6 (2011), pp. 2566–2570.

[16] Chen Chen, Kui Liu, and Nasser Kehtarnavaz. “Real-time human action recognition based on depth motion maps”. In: *Journal of real-time image processing* 12.1 (2016), pp. 155–163.

[17] Srikanth Cherla, Kaustubh Kulkarni, Amit Kale, and Viswanathan Ramasubramanian. “Towards fast, view-invariant human action recognition”. In: *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW’08. IEEE Computer Society Conference on*. IEEE. 2008, pp. 1–8.

[18] Corinna Cortes and Vladimir Vapnik. “Support-vector networks”. In: *Machine learning* 20.3 (1995), pp. 273–297.

[19] Harold Scott Macdonald Coxeter. *Regular polytopes*. Courier Corporation, 1973.

[20] Koby Crammer and Yoram Singer. “On the algorithmic implementation of multiclass kernel-based vector machines”. In: *Journal of machine learning research* 2.Dec (2001), pp. 265–292.

[21] Walter J Culver. “On the existence and uniqueness of the real logarithm of a matrix”. In: *Proceedings of the American Mathematical Society* 17.5 (1966), pp. 1146–1151.

[22] Naresh P Cuntoor and Rama Chellappa. “Key frame-based activity representation using antieigenvalues”. In: *Asian Conference on Computer Vision*. Springer. 2006, pp. 499–508.

[23] Navneet Dalal and Bill Triggs. “Histograms of oriented gradients for human detection”. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE. 2005, pp. 886–893.

[24] Srijan Das, Michal Koperski, Francois Bremond, and Gianpier Francesca. “Action Recognition based on a mixture of RGB and Depth based skeleton”. In: *Advanced Video and Signal-Based Surveillance, September 2017. AVSS 2017. Proceedings of the 14th IEEE International Conference on*. IEEE. 2017.

[25] Carl De Boor. *Spline toolbox for use with MATLAB: user’s guide, version 3*. MathWorks, 2005.

[26] Maxime Devanne, Hazem Wannous, Stefano Berretti, Pietro Pala, Mohamed Daoudi, and Alberto Del Bimbo. “3-D human action recognition by shape analysis of motion trajectories on riemannian manifold”. In: *IEEE transactions on cybernetics* 45.7 (2015), pp. 1340–1352.

[27] Piotr Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. “Behavior recognition via sparse spatio-temporal features”. In: *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*. IEEE. 2005, pp. 65–72.

[28] Rachit Dubey, Bingbing Ni, and Pierre Moulin. “A depth camera based fall recognition system for the elderly”. In: *International Conference Image Analysis and Recognition*. Springer. 2012, pp. 106–113.

[29] Vinayak Elangovan, Vinod K Bandaru, and Amir Shirkhodaie. “Team activity analysis and recognition based on Kinect depth map and optical imagery techniques”. In: *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics. 2012, 83920W–83920W.

[30] Georgios Evangelidis, Gurkirt Singh, and Radu Horaud. “Skeletal quads: Human action recognition using joint quadruples”. In: *Pattern Recognition (ICPR), 2014 22nd International Conference on*. IEEE. 2014, pp. 4513–4518.

[31] Sean Ryan Fanello, Ilaria Gori, Giorgio Metta, and Francesca Odone. “Keep it simple and sparse: Real-time action recognition”. In: *The Journal of Machine Learning Research* 14.1 (2013), pp. 2617–2640.

[32] Pasquale Foggia, Gennaro Percannella, Alessia Saggese, and Mario Vento. “Recognizing human actions by a bag of visual words”. In: *2013 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE. 2013, pp. 2910–2915.

[33] Wolfgang Förstner and Boudewijn Moonen. “A metric for covariance matrices”. In: *Geodesy-The Challenge of the 3rd Millennium*. Springer, 2003, pp. 299–309.

[34] Baptiste Fosty, Carlos Fernando Crispim-Junior, Julien Badie, François Bremond, and Monique Thonnat. “Event recognition system for older people monitoring using an rgb-d camera”. In: *ASROB-workshop on assistance and service robotics in a human environment*. 2013.

[35] Simon Fothergill, Helena Mentis, Pushmeet Kohli, and Sebastian Nowozin. “Instructing people for training gestural interactive systems”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2012, pp. 1737–1746.

[36] Dirk Gehrig and Tanja Schultz. “Selecting relevant features for human motion recognition”. In: *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*. IEEE. 2008, pp. 1–4.

[37] Enjie Ghorbel, Rèmi Boutteau, Jacques Boonaert, Xavier Savatier, and Stèphane Lecoeuche. “3D real-time human action recognition using a spline interpolation approach”. In: *Image Processing Theory Tools and Applications (IPTA), 2015 5th International Conference on*. IEEE. 2015.

[38] Andrew Gilbert, John Illingworth, and Richard Bowden. “Fast realistic multi-action recognition using mined dense spatio-temporal features”. In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE. 2009, pp. 925–931.

[39] Alvina Goh and René Vidal. “Clustering and dimensionality reduction on Riemannian manifolds”. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE. 2008, pp. 1–7.

[40] Simon Hadfield and Richard Bowden. “Hollywood 3D: Recognizing actions in 3D natural scenes”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 3398–3405.

[41] Mounir Hammouche, Enjie Ghorbel, Anthony Fleury, and Sébastien Ambellouis. “Toward a real time view-invariant 3D action recognition”. In: *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*. 2016.

[42] Lyes Hamoudi. “Application de techniques d’apprentissage pour la détection et la reconnaissance d’individus”. PhD thesis. Lille 1, 2011.

[43] Michael Harville and Dalong Li. “Fast, integrated person tracking and activity recognition with plan-view templates from a single stereo camera”. In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. Vol. 2. IEEE. 2004, pp. II–II.

[44] Yonghong Hou, Zhaoyang Li, Pichao Wang, and Wanqing Li. “Skeleton optical spectra based action recognition using convolutional neural networks”. In: *IEEE Transactions on Circuits and Systems for Video Technology* (2016).

[45] Mohamed E Hussein, Marwan Torki, Mohammad Abdelaziz Gowayyed, and Motaz El-Saban. “Human Action Recognition Using a Temporal Hierarchy of Covariance Descriptors on 3D Joint Locations.” In: *IJCAI*. Vol. 13. 2013, pp. 2466–2472.

[46] Sadeep Jayasumana, Richard Hartley, Mathieu Salzmann, Hongdong Li, and Mehrtash Harandi. “Kernel methods on the riemannian manifold of symmetric positive definite matrices”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 73–80.

[47] Gunnar Johansson. “Visual perception of biological motion and a model for its analysis”. In: *Attention, Perception, & Psychophysics* 14.2 (1973), pp. 201–211.

[48] Hanguen Kim, Sangwon Lee, Youngjae Kim, Serin Lee, Dongsung Lee, Jinsun Ju, and Hyun Myung. “Weighted joint-based human behavior recognition algorithm using only depth information for low-cost intelligent video-surveillance system”. In: *Expert Systems with Applications* 45 (2016), pp. 131–141.

[49] Alexander Klaser, Marcin Marszałek, and Cordelia Schmid. “A spatio-temporal descriptor based on 3D-gradients”. In: *BMVC 2008-19th British Machine Vision Conference*. British Machine Vision Association. 2008, pp. 275–1.

[50] Christian Peter Klingenberg. “Cranial integration and modularity: insights into evolution and development from morphometric data”. In: *Hystrix, the Italian Journal of Mammalogy* 24.1 (2013), pp. 43–58.

[51] Igor Kviatkovsky, Ehud Rivlin, and Ilan Shimshoni. “Online action recognition using covariance of shape and motion”. In: *Computer Vision and Image Understanding* 129 (2014), pp. 15–26.

[52] Colin Lea, Gregory D Hager, and René Vidal. “An improved model for segmentation and recognition of fine-grained activities with application to surgical training tasks”. In: *2015 IEEE Winter Conference on Applications of Computer Vision*. IEEE. 2015, pp. 1123–1129.

[53] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. “Action recognition based on a bag of 3D points”. In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*. IEEE. 2010, pp. 9–14.

[54] Li Liu and Ling Shao. “Learning Discriminative Representations from RGB-D Video Data.” In: *IJCAI*. Vol. 1. 2013, p. 3.

[55] Zhi Liu, Chenyang Zhang, and Yingli Tian. “3d-based deep convolutional neural network for action recognition with depth sequences”. In: *Image and Vision Computing* 55 (2016), pp. 93–100.

[56] Yanyun Lu, Khaled Boukharouba, Jacques Boonært, Anthony Fleury, and Stéphane Lecoeuche. “Application of an incremental SVM algorithm for on-line human recognition from video surveillance using texture and color features”. In: *Neurocomputing* 126 (2014), pp. 132–140.

[57] Fengjun Lv and Ramakant Nevatia. “Recognition and segmentation of 3-d human action using hmm and multi-class adaboost”. In: *European conference on computer vision*. Springer. 2006, pp. 359–372.

[58] Syed Zain Masood, Chris Ellis, Marshall F Tappen, Joseph J Laviola Jr, and Rahul Sukthankar. “Exploring the trade-off between accuracy and observational latency in action recognition”. In: *International Journal of Computer Vision* 101.3 (2013), pp. 420–436.

[59] Georgios Mastorakis and Dimitrios Makris. “Fall detection system using Kinect’s infrared sensor”. In: *Journal of Real-Time Image Processing* 9.4 (2014), pp. 635–646.

[60] Leandro Miranda, Thales Vieira, Dimas Martinez, Thomas Lewiner, Antonio W Vieira, and Mario FM Campos. “Real-time gesture recognition from depth data through key poses learning and decision forests”. In: *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images*. IEEE. 2012, pp. 268–275.

[61] Natalia Neverova, Christian Wolf, Graham W Taylor, and Florian Nebout. “Multiscale deep learning for gesture detection and localization”. In: *Workshop at the European conference on computer vision*. Springer. 2014, pp. 474–490.

[62] Ferda Ofli, Rizwan Chaudhry, Gregorij Kurillo, René Vidal, and Ruzena Bajcsy. “Sequence of the most informative joints (smij): A new representation for human skeletal action recognition”. In: *Journal of Visual Communication and Image Representation* 25.1 (2014), pp. 24–38.

[63] Eshed Ohn-Bar and Mohan Manubhai Trivedi. “Joint angles similarities and HOG2 for action recognition”. In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*. IEEE. 2013, pp. 465–470.

[64] Kensuke Onuma, Christos Faloutsos, and Jessica K Hodgins. “FMDistance: A fast and effective distance function for motion capture data”. In: *Short Papers Proceedings of EUROGRAPHICS* 2 (2008).

[65] Omar Oreifej and Zicheng Liu. “Hon4d: Histogram of oriented 4D normals for activity recognition from depth sequences”. In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE. 2013, pp. 716–723.

[66] José Ramón Padilla-López, Alexandros André Chaaraoui, and Francisco Flórez-Revuelta. “A discussion on the validation tests employed to compare human action recognition methods using the MSR action3d dataset”. In: *arXiv preprint arXiv:1407.7390* (2014).

[67] Yanwei Pang, Yuan Yuan, and Xuelong Li. “Gabor-based region covariance matrices for face recognition”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 18.7 (2008), pp. 989–993.

[68] Georgios Th Papadopoulos, Apostolos Axenopoulos, and Petros Daras. “Real-time skeleton-tracking-based human action recognition using kinect data”. In: *MultiMedia Modeling*. Springer. 2014, pp. 473–483.

[69] Ronald Poppe. “A survey on vision-based human action recognition”. In: *Image and vision computing* 28.6 (2010), pp. 976–990.

[70] Ruizhi Qiao, Lingqiao Liu, Chunhua Shen, and Anton van den Hengel. “Learning discriminative trajectorylet detector sets for accurate skeleton-based action recognition”. In: *Pattern Recognition* (2017).

[71] Hossein Rahmani, Arif Mahmood, Du Huynh, and Ajmal Mian. “Histogram of oriented principal components for cross-view action recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 38.12 (2016), pp. 2430–2443.

[72] Michalis Raptis, Darko Kirovski, and Hugues Hoppe. “Real-time classification of dance gestures from skeleton animation”. In: *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation*. ACM. 2011, pp. 147–156.

[73] Michael Rosenberg, Ashleigh L Thornton, Brendan S Lay, Brodie Ward, David Nathan, Daniel Hunt, and Rebecca Braham. “Development of a Kinect Software Tool to Classify Movements during Active Video Gaming”. In: *PloS one* 11.7 (2016), e0159356.

[74] AQ Md Sabri, Jacques Boonaert, Stéphane Lecoeuche, and E Mouaddib. “Human action classification using surf based spatio-temporal correlated descriptors”. In: *2012 19th IEEE International Conference on Image Processing*. IEEE. 2012, pp. 1401–1404.

[75] Hamed Sarbolandi, Damien Lefloch, and Andreas Kolb. “Kinect range sensing: Structured-light versus time-of-flight kinect”. In: *Computer vision and image understanding* 139 (2015), pp. 1–20.

[76] Isaac J Schoenberg. “Metric spaces and positive definite functions”. In: *Transactions of the American Mathematical Society* 44.3 (1938), pp. 522–536.

[77] Junjie Shan and Srinivas Akella. “3D human action segmentation and recognition using pose kinetic energy”. In: *Advanced Robotics and its Social Impacts (ARSO), 2014 IEEE Workshop on*. IEEE. 2014, pp. 69–75.

[78] Amr Sharaf, Marwan Torki, Mohamed E Hussein, and Motaz El-Saban. “Real-time multi-scale action detection from 3d skeleton data”. In: *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*. IEEE. 2015, pp. 998–1005.

[79] Yuping Shen and Hassan Foroosh. “View-invariant action recognition using fundamental ratios”. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE. 2008, pp. 1–6.

[80] Yubo Shi and Yongxiong Wang. “A Local Feature Descriptor Based on Energy Information for Human Activity Recognition”. In: *Advanced Intelligent Computing Theories and Applications*. Springer, 2015, pp. 311–317.

[81] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. “Real-time human pose recognition in parts from single depth images”. In: *Communications of the ACM* 56.1 (2013), pp. 116–124.

[82] Parul Shukla, Kanad Kishore Biswas, and Prem K Kalra. “Action Recognition using Temporal Bag-of-Words from Depth Maps.” In: *MVA*. 2013, pp. 41–44.

[83] Rim Slama, Hazem Wannous, and Mohamed Daoudi. “Grassmannian representation of motion depth for 3D human gesture and action recognition”. In: *Pattern Recognition (ICPR), 2014 22nd International Conference on*. IEEE. 2014, pp. 3499–3504.

[84] Suvrit Sra. *Positive definite matrices and the symmetric Stein divergence*. Tech. rep. 2011.

[85] Chang Tang, Wanqing Li, Chunping Hou, Pichao Wang, Yonghong Hou, Jing Zhang, and Philip O Ogunbona. “Online Action Recognition based on Incremental Learning of Weighted Covariance Descriptors”. In: *arXiv preprint arXiv:1511.03028* (2015).

[86] Beau Tippetts, Dah Jye Lee, Kirt Lillywhite, and James Archibald. “Review of stereo vision algorithms and their suitability for resource-limited systems”. In: *Journal of Real-Time Image Processing* 11.1 (2016), pp. 5–25.

[87] Pavan Turaga, Rama Chellappa, Venkatramana S Subrahmanian, and Octavian Udrea. “Machine recognition of human activities: A survey”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 18.11 (2008), pp. 1473–1488.

[88] Oncel Tuzel, Fatih Porikli, and Peter Meer. “Region covariance: A fast descriptor for detection and classification”. In: *European conference on computer vision*. Springer. 2006, pp. 589–600.

[89] Raquel Urtasun and Pascal Fua. “3D tracking for gait characterization and recognition”. In: *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*. IEEE. 2004, pp. 17–22.

[90] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*. Vol. 1. Wiley New York, 1998.

[91] Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa. “Human action recognition by representing 3D skeletons as points in a Lie group”. In: *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE. 2014, pp. 588–595.

[92] René Vidal, Stefano Soatto, and Alessandro Chiuso. “Applications of hybrid system identification in computer vision”. In: *Control Conference (ECC), 2007 European*. IEEE. 2007, pp. 4853–4860.

[93] Jiang Wang, Xiaohan Nie, Yin Xia, Ying Wu, and Song-Chun Zhu. “Cross-view action modeling, learning and recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 2649–2656.

[94] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. “Mining actionlet ensemble for action recognition with depth cameras”. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 1290–1297.

[95] Pichao Wang, Wanqing Li, Zhimin Gao, Jing Zhang, Chang Tang, and Philip O Ogunbona. “Action recognition from depth maps using deep convolutional neural networks”. In: *IEEE Transactions on Human-Machine Systems* 46.4 (2016), pp. 498–509.

[96] Ruiping Wang, Huimin Guo, Larry S Davis, and Qionghai Dai. “Covariance discriminative learning: A natural and efficient approach to image set classification”. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 2496–2503.

[97] Daniel Weinland, Remi Ronfard, and Edmond Boyer. “A survey of vision-based methods for action representation, segmentation and recognition”. In: *Computer vision and image understanding* 115.2 (2011), pp. 224–241.

[98] Di Wu and Ling Shao. “Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 724–731.

[99] Lu Xia and JK Aggarwal. “Spatio-temporal depth cuboid similarity feature for activity recognition using depth camera”. In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE. 2013, pp. 2834–2841.

[100] Lu Xia, Chia-Chih Chen, and JK Aggarwal. “View invariant human action recognition using histograms of 3D joints”. In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*. IEEE. 2012, pp. 20–27.

[101] Xiaodong Yang and YingLi Tian. “Eigenjoints-based action recognition using naive-bayes-nearest-neighbor”. In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*. IEEE. 2012, pp. 14–19.

[102] Xiaodong Yang and YingLi Tian. “Super normal vector for activity recognition using depth sequences”. In: *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE. 2014, pp. 804–811.

[103] Xiaodong Yang, Chenyang Zhang, and YingLi Tian. “Recognizing actions using depth motion maps-based histograms of oriented gradients”. In: *Proceedings of the 20th ACM international conference on Multimedia*. ACM. 2012, pp. 1057–1060.

[104] Tsz-Ho Yu, Tae-Kyun Kim, and Roberto Cipolla. “Real-time Action Recognition by Spatiotemporal Semantic and Structural Forests.” In: *BMVC*. Vol. 2. 5. 2010, p. 6.

[105] Mihai Zanfir, Marius Leordeanu, and Cristian Sminchisescu. “The moving pose: An efficient 3D kinematics descriptor for low-latency action recognition and detection”. In: *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE. 2013, pp. 2752–2759.

[106] Xin Zhao, Xue Li, Chaoyi Pang, Quan Z Sheng, Sen Wang, and Mao Ye. “Structured Streaming Skeleton–A New Feature for Online Human Gesture Recognition”. In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 11.1s (2014), p. 22.

[107] Yang Zhao, Zicheng Liu, Lu Yang, and Hong Cheng. “Combing rgb and depth map features for human activity recognition”. In: *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*. IEEE. 2012, pp. 1–4.

[108] Fan Zhu, Ling Shao, Jin Xie, and Yi Fang. “From handcrafted to learned representations for human action recognition: a survey”. In: *Image and Vision Computing* 55 (2016), pp. 42–52.

[109] Wentao Zhu, Cuiling Lan, Junliang Xing, Wenjun Zeng, Yanghao Li, Li Shen, and Xiaohui Xie. “Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks”. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press. 2016, pp. 3697–3703.

[110] Yu Zhu, Wenbin Chen, and Guodong Guo. “Fusing spatiotemporal features and joints for 3D action recognition”. In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*. IEEE. 2013, pp. 486–491.