Inferring Human Activities from Observation via Semantic Reasoning: A novel method for transferring skills to robots

Karinne Ramírez Amaro

Die Dissertation wurde am 11.09.2014 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 24.03.2015 angenommen.
To my family
Abstract

One fundamental issue of autonomous robots in task domains is its capability to learn new skills and to re-use their previous experiences in different situations as efficient, intuitive and reliable as possible. A promising mechanism to achieve that is via demonstrations. This thesis presents a framework that includes a new learning technique based on semantic representations that considers the re-usability and the inclusion of new skills in a robust manner. The introduced framework has been validated in a Humanoid robot under different constraints in several scenarios.

This thesis has four major contributions: a) the design of a framework that is able to observe and extract relevant information from human activities from different sources; b) an assessment of the state-of-the-art perception systems highlighting their limitations; c) a new method based on semantic representations with a reasoning engine which enhances the recognition of human activities; and d) the integration and evaluation of multiple scenarios fully implemented in a Humanoid robot. The presented framework has been assessed using different perception modalities which poses distinct and challenging levels of complexity to demonstrate that our framework does not depend on the analyzed task.

The key and most important contribution of this work is our approach to determine adequate semantic representations of the observed activities. Our method enables robots to obtain and define a higher-level understanding of demonstrator’s behavior via semantic representations. The obtained semantics extracts the essence of the observed activities where a meaningful semantic description in terms of human motions and object properties is obtained. This allows our framework to generalize the learned experiences for new situations by including new behaviors on-demand in an intuitive manner.

We quantitatively and qualitatively assess our framework by implementing it on a Humanoid robot (iCub) using different perception sources in several situations. The results show that our robot correctly recognizes human behaviors from on-line demonstrations with an accuracy of 87.44%, which is even better than a random participant recognizing the same demonstrations (about 76.68%). The obtained results indicate that the learned representations can be generalized and utilized for the difficult and challenging problem of transferring skills to robots.
Zusammenfassung


Diese Arbeit leistet vier wesentliche wissenschaftliche Beiträge: a) die Entwicklung eines in sich konsistenten Satzes von Lösungsansätzen und Konzepten, um menschliche Aktivitäten zu beobachten und relevante Informationen verschiedenen Ursprungs zu extrahieren, b) die Beurteilung und Einschätzung bereits etablierter Erkennungsverfahren, wobei deren Grenzen hervorgehoben werden, c) eine neue Methode, welche semantische Repräsentationen in Kombination mit einer Argumentations- und Entscheidungsmaschine verwendet und somit die Erkennung menschlicher Aktivitäten verbessert und d) die Implementierung und Evaluierung der Lösungsansätze und Konzepte unter Verwendung eines humanoiden Roboters in vielen verschiedenen Einsatzszenarien. Hierbei wurde darauf Wert gelegt, das System in möglichst vielen verschiedenen Aufgaben auf einer herausfordernden Komplexitätstufe zu testen, um zu demonstrieren, dass die vorgestellten Lösungsansätze und Konzepte nicht von der zu analysierenden Aufgabe abhängen.

neues Verhalten angepasst werden können und neues Verhalten bei Bedarf in einer Intuitiven Art und Weise erzeugt werden kann.

Wir beurteilen die Performance unseres Systems in qualitativer und quantitativer Weise, indem wir es auf einen humanoiden Roboter (iCub) implementieren und anschließend den Roboter mit verschiedenen Wahrnehmungsquellen und Situationen konfrontieren. Die Ergebnisse zeigen, dass unser Roboter menschliche Aktivitäten in Echtzeit mit einer Genauigkeit von etwa 87.44% korrekt erkennen kann. Das ist sogar besser als ein zufälliger menschlicher Teilnehmer der die gleichen Aktivitäten zu erkennen versucht (76.68%). Somit kann anhand der Ergebnisse darauf geschlossen werden, dass erlernte Fähigkeiten generalisiert werden können und dadurch den anspruchsvollen Transfer von Fähigkeiten auf Robotersysteme erleichtern.
Resumen

Uno de los principales problemas que los robots autónomos presentan es su capacidad de entender y aprender nuevas actividades mediante demostraciones. Esto implica que dichos robots sean capaces de re-utilizar sus experiencias previas de tal manera que éstas puedan ser aplicadas en diferentes situaciones. Adicionalmente, la técnica de aprendizaje utilizada debe de incluir la posibilidad de detectar nuevas habilidades de una manera eficiente. Por lo tanto, los robots deben de aprender nuevas tareas de una manera rápida, intuitiva y confiable. En esta tesis presentamos un sistema que incluye una nueva técnica de aprendizaje basada en Representaciones Semánticas, la cual es capaz de incluir nuevas tareas de manera eficiente y robusta. Nuestro sistema ha sido validado en un robot humanoide bajo diferentes condiciones y en distintos escenarios.

Esta tesis contiene principalmente cuatro contribuciones: a) el diseño de un sistema que es capaz de procesar observaciones y extraer la información más relevante sobre las actividades del humano usando diferentes tipos de sensores usados para capturar los movimientos del humano; b) la evaluación de las técnicas actuales utilizadas en el estado del arte para resolver el problema de percibir e identificar las actividades demostradas, donde las limitaciones de dichos sistemas son presentados; c) la presentación de un nuevo método que extrae las Representaciones Semánticas, dicho método incluye un proceso de razonamiento para mejorar la percepción y la inferencia de las actividades observadas; y finalmente d) la integración y evaluación de nuestro sistema en un robot humanoide en diferentes escenarios y bajo varias restricciones. Nuestro sistema ha sido evaluado usando diferentes y desafiantes niveles de complejidad para demostrar que nuestro sistema no depende de la tarea entrenada.

La contribución más importante de nuestro trabajo es nuestro nuevo método para extraer de manera apropiada las Representaciones Semánticas de las actividades observadas. De esta manera, nuestro sistema permite que los robots obtengan y determinen altos niveles de entendimiento a través de la extracción del significado de las tareas demostradas. Dicha abstracción permite obtener la esencia de la actividad observada; esto significa que una descripción semántica es adquirida en base a los movimientos del humano y de las propiedades de los objetos observados. Esto permite que nuestro sistema sea capaz de re-usar experiencias pasadas...
y aplicarlas en los nuevos escenarios. Adicionalmente, nuestro sistema automáticamente incremanta sus representaciones de razonamiento y se adapta de manera eficiente e intuitiva cuando nuevas actividades son observadas y aprendidas sin necesidad de una nueva fase de entrenamiento, lo anterior se debe a los distintos niveles de modularidad que nuestro sistema posee.

Finalmente, validamos cuantitativa- y cualitativamente nuestro sistema usando diferentes medios de observación implementados en un robot humanoide, el iCub. Los resultados demuestran que nuestro robot es capaz de correctamente reconocer las actividades observadas en tiempo real con una precisión de 87.44%, el cual es incluso mejor que cuando una persona observa y reconoce las mismas actividades (aproximadamente 76.68%). Demostrando así que las representaciones obtenidas no dependen de la tarea ejecutada y que dichas representaciones pueden ser generalizadas. Esto prueba que nuestro sistema puede ser utilizado para el complejo y desafiante problema de la transferencia de habilidades de humanos hacia los robots, las cuales permiten una interacción más natural entre un humano y un robot.
Acknowledgements

I would like start thanking my supervisor Prof. Gordon Cheng to whom I am very grateful for his unlimited support during my PhD research. Thank you for all your guidance, encouragement, advice and especially for your patience during all our invaluable discussions. Thanks for believe in my crazy ideas!

I am very thankful to my co-supervisor, Prof. Michael Beetz, for his constant support and encouragement throughout my research. I really appreciate the invaluable opportunities that he provided to me during his supervision. Moreover, I would like to thank Prof. Inamura for all his very interesting discussions when he visited our laboratory at ICS-TUM. Thank you for introducing me to a new way of thinking and for your kindness as a person. I would like to thank Prof. Dr. Diepold for serving on my thesis committee.

During my research I have the support of my colleagues, friends, lab-mates and students at TUM and I would like to thank them for all the useful feedback to my work and especially thanks to Ilona that kindly help me through my administrative procedures. By this means I would like to thank from the faculty of Electrical Engineering to Philipp, Andreas, Florian, Erhard, Bren, Stefan, Ewald, Wibke, Brigitte, John, Anja and Mohsen. From the Informatics and Mathematics departments to Lars, Dejan, Humera, Carmen, Lorenz, Thomas, Moritz, Sebastian, Federico. Also, I would like to thank my collaborators from the Psychology department for all the interesting exchange of ideas, thanks to Michael Zehetleitner, Michael H., Lisa, Agnieszka and Jasmin.

Thanks also to Prof. Byoung-Tak Zhang for hosting me one month in the Bioinformatics laboratory at the Seoul National University as part of the GenKo project founded by DAAD. Especially, I would like to thank Eun-Sol, Jiseob, Chung-Yeon and to all the laboratory members for you kindness and assistance during my staying in Seoul.

With all my hard I wish to express my gratitude to my companion in life and science, my husband Emmanuel Dean. He represents one of the most important motivation and inspiration not only during this period but at all times. I really do not have words to describe the deeply love that I feel for you, I love you so much ♥. Thank you for all your patience and kindness during this long and fun road.
I would like to take the opportunity to express my deeply gratitude to my parents Aniceto Ramírez and Margarita Amaro, you are the best example that a daughter can have during life. I am very thankful for all the sacrifices and efforts that you did to guide me so wisely until this day. Thank you so much, I love you. Similarly, I would like to thank my siblings Margarita, Sergio, Eduardo y Julien for all the joyful moments that we have live together and at the distance. Moreover, I would like to thank my grandmother Chinta for always guide me and advise all the time. The same manner, I would like to express my gratitude to my three guardian angels, my aunts Rosa, Socorro and Ninfa for all your love and understanding. Also, I would like to thank my mother-in-law Maria del Carmen for always being there for us. In general I wish to thank all my big family that has been always even at the distance encouraging me. Many thanks to all of you.

I gratefully acknowledge the funding sources that made my Ph.D. work possible, such as my scholarship CONACYT-DAAD, the European Community Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 609206 and the DFG cluster of excellence Cognition for Technical systems (CoTeSys).

Munich, 11th September 2012
Agradecimientos

Me gustaría empezar agradeciendo a mi asesor el Prof. Gordon Cheng, con el cual estoy enormemente agradecida por su apoyo incondicional durante mis estudios de doctorado. Muchas gracias por su orientación, aliento, consejo y en especial por su paciencia durante nuestras valiosas discusiones. Gracias por creen en mis ideas locas!

Estoy enormemente agradecida con mi segundo supervisor el Prof. Michael Beetz, por su constante apoyo y estímulo a lo largo de mi investigación. Realmente aprecio mucho la oportunidad tan valiosa que me concedió como miembro de sus grupo y durante su constante supervisión. También, me gustaría agradecer al Prof. Inamura por todas las charlas tan interesantes que tuvimos mientras nos visitaba en el laboratorio de ICS-TUM. Muchas gracias por introducirme a una nueva manera de pensar y por su amabilidad y humildad como persona. Asimismo, quisiera agradecer al Prof. Dr. Diepold por formar parte de mi comité.

A lo largo de mi investigación tuve el apoyo constante de mis colegas, amigos, miembros de laboratorio y estudiantes en el TUM y quisiera agradecerles por sus muy útiles comentarios hacia mi trabajo. Deseo expresar un agradecimiento especial a Ilona, que amablemente me ayudó con mis trámites burocráticos. A sí mismo a mis colegas de la facultad de Ingeniería Eléctrica: Philipp, Andreas, Florian, Erhard, Bren, Stefan, Ewald, Brigitte, John, Mohsen. Ahora a los de la facultad de Informática y Matemáticas: Lars, Dejan, Humera, Carmen, Lorenz, Thomas, Moritz, Sebastian, Federico. También, quisiera agradecer a mis colaboradores del departamento de Psicología por el interesante intercambio de ideas, gracias a Michael Zehetleitner, Michael H., Lisa, Agnieszka and Jasmin.

Gracias también al Prof. Byoung-Tak Zhang por amablemente aceptarme un mes en su laboratorio de Bio-informática en la Universidad Nacional de Seoul como parte del proyecto GenKo pagado por la DAAD. Especialmente, quisiera agradecer a Eun-Sol, Jiseob, Chung-Yeon y a todos los miembros del laboratorio por su amabilidad y ayuda durante mi estancia en Seoul.

Con todo mi corazón deseo expresar mi eterna gratitud a mi compañero de vida y ciencia, mi adorado esposo Emmanuel Dean. El representa el pilar más importante de mi vida, mi mayor motivación y grande inspiración en todo momento y especialmente en los momentos difíciles.
Realmente, me quedo sin palabras y con nudos muy grandes en la garganta al momento de expresar y describir todo el amor que le tengo, te amo enormemente ♥. Muchas gracias por toda tu paciencia y amabilidad durante este camino aunque largo, divertido :)

Quisiera también aprovechar la oportunidad para agradecerle enormemente a mis padres Aniceto Ramírez y Margarita Amaro, que significan el mejor ejemplo a seguir que he tenido a lo largo de estos años. Les agradezco enormemente todos sus sacrificios y esfuerzos para guiarme tan sabiamente como lo han hecho. Muchas gracias, los quiero mucho. De la misma manera quisiera agradecer a mis hermanos Margarita, Sergio, Eduardo y Julien por todas las alegrías que hemos vivido en todos estos años y por todas sus muestras de cariño. También quisiera agradecer a mi abuelita Chinta por siempre apoyarme y guiarne en todo momento. Además, quiero agradecer a mis ángeles guardianes, mis tías Rosa, Socorro y Ninfa por todo su comprensión y cariño siempre. También quisiera agradecer a mi suegra María del Carmen por estar siempre al pendiente de nosotros. En general quiero agradecer a toda mi familia que a estado en todo momento para apoyarme. Muchas gracias a todos.

Estoy muy agradecida con las instituciones que financiaron mis estudios de doctorado: CONACYT-DAAD, la comisión Europea (Seventh Framework Programme FP7/2007-2013) bajo el financiamiento no. 609206 y el DFG cluster de excelencia CoTeSys (Cognition for Technical systems)

_Munich, 11th September 2012_
# Contents

Dedicatoria ............................................................................. III

Abstract ................................................................................ III

Kurzfassung .......................................................................... V

Resumen .............................................................................. VII

Acknowledgements ................................................................. IX

Agradecimientos ................................................................... XI

Contents ............................................................................... XIII

List of Publications ............................................................... XVII

List of Figures ..................................................................... XXI

List of Tables ....................................................................... XXVII

List of Algorithms ............................................................... XXIX

1. Introduction ........................................................................ 1
   1.1. Motivation ................................................................. 3
   1.2. Problem description .................................................. 4
   1.3. Semantic approach to infer human activities ............... 7
   1.4. Contributions .......................................................... 9
   1.5. Thesis outline ........................................................ 10

2. Related work ....................................................................... 13
   2.1. Research areas investigated ...................................... 15
       2.1.1. Computer Vision and Machine Learning communities .... 17
4.6. Recognition of 3-class problem using Color-Based techniques .......... 87
4.6.1. Pipeline of the Color-Based recognition .......................... 88
4.6.2. Results of hand motions and object properties .................... 95
4.7. Summary and contributions of this chapter .......................... 99

5. **Inferring Semantic Human Activities Reasoner (ISHAR)** .......... 103
5.1. Inferring human intentions using ISHAR ............................ 105
5.2. ISHAR algorithm description ........................................... 108
  5.2.1. Basic human activities method .................................... 108
  5.2.2. Granular human activities approach ................................. 111
5.3. Results of semantic reasoning .......................................... 112
  5.3.1. Advantages of the ISHAR method .................................. 116
  5.3.2. Generalization toward different scenarios .......................... 120
  5.3.3. Testing the robustness of the obtained semantic rules ............. 122
  5.3.4. Enhancing activity recognition using ISA and ISHAR ............. 125
  5.3.5. Activity recognition using Color-Based technique and ISHAR ..... 127
5.4. Knowledge and reasoning engine ....................................... 133
  5.4.1. Knowledge framework ............................................... 134
  5.4.2. Proposed algorithm for expanding the ontology .................... 136
5.5. Results of knowledge base and reasoning engine ..................... 140
5.6. Summary and contributions of this chapter .......................... 143

6. **Assessment and integration of our framework in complex scenarios** .... 147
6.1. Transferring the inferred task to robots ............................. 149
6.2. Implementation on the iCub humanoid robot .......................... 150
  6.2.1. Design of iCubActionRecognition module ........................ 153
6.3. Experimental validation on the iCub .................................. 161
6.4. Enhancing ISHAR using egocentric views ............................ 165
  6.4.1. Approach to combine the external and gaze information in ISHAR .. 166
  6.4.2. Results of human activity recognition using gaze input .......... 170
  6.4.3. Enhancing the reasoning and knowledge engine with gaze input .. 174
  6.4.4. Transferring the obtained models into humanoid robots .......... 175
6.5. Testing ISHAR in more complex scenarios .......................... 178
  6.5.1. The SIGVerse simulator ............................................ 179
  6.5.2. Extracting information from SIGVerse .............................. 180
  6.5.3. Results of activity recognition of a virtual robot using ISHAR ..... 183
6.5.4. Transferring the acquired models into humanoid robots 184
6.6. Summary and contributions of this chapter 186

7. Conclusions 189
7.1. Summary and contributions 191
7.1.1. Key elements addressed in this work 192
7.1.2. Enhancing ISHAR with egocentric views 193
7.1.3. Assessment of ISHAR using complex and new scenarios 194
7.1.4. Contributions and overall achievements 195
7.2. Outlook 197

A. Optimization principles to transfer human-like motions to robots 199
A.1. Using optimization principle analysis 201
A.2. Experimental validation of the human-like motions on the iCub 204
A.3. Summary of this appendix 206

Bibliography 207
List of Publications

1. Own published papers

The following papers were published or have been submitted as part of my research:

Journal papers


Conference and workshop papers


Submitted Papers


2. Awards

During my PhD research studies, I was granted with the following awards:

- **Dr. Kanako Miura Travel Grant**: In recognition of Dr. Kanako Miura’s contributions to Humanoid Robotics, the IEEE RAS Technical Committee on Humanoid Robotics initiate the Miura Memorial award. The goal is to assist female PhD and MS students entering the field with travel to Humanoids 2013 in Atlanta. This year (2013), 6 people received the award.

- **DFG Student Travel Support**: I was awarded by the Deutsche Forschungsgemeinschaft (DFG), with a travel support to participate at ICRA 2013. Only 20 students obtained this support out of 89 applications.

- **Google Fellowship**: Google Anita Borg Memorial Scholarship: Europe, the Middle East and Africa 2011. In Anita Borg honor, Google support women in technology with the Google Anita Borg Memorial Scholarship. Google hopes to encourage women to excel in computing and technology and become active role models and leaders in the field.

- **DAAD Fellowship**: Scholarship for Ph. D. researchaward by German Academic Exchange Service (DAAD) and The National Council on Science and Technology (CONACYT). From October 2009 until September 2013. Scholarship code: A/09/72618.
## List of Figures

1.1. Motivation behind our research .............................................. 3  
1.2. Different pre-processing outcomes from observable inputs ........... 5  
1.3. Examples of the parameters that can be extracted from the pouring activity .. 6  
1.4. Different steps of our proposed system .................................. 7  

2.1. Main research areas investigated for learning human activities from observation 15  
2.2. Examples of the evolution of cognition in different robotic platforms ........ 16  
2.3. Examples of activities typically analyzed on the benchmark data sets ........ 20  
2.4. Examples of the complex behaviors of the analyzed data sets .............. 21  
2.5. Challenges in human activity recognition either using visual observation or trajectories .......................................................... 31  

3.1. Classical diagram of a flexible imitation system ............................ 39  
3.2. Main modules that conform our proposed framework ...................... 41  
3.3. Set-up description to acquire the new proposed data sets ................. 45  
3.4. Exemplification of the location of the sensors as well as the items used for the preparation of the sandwich .................................................. 46  
3.5. Examples of different cameras used in the *pancake making* scenario .... 48  
3.6. Different points of view of the cameras used for the analysis of the *sandwich making* scenario .......................................................... 50  
3.7. Examples of one subject performing the *setting the table* task ........... 51  
3.8. General view of the labeling tool and the outcome file .................... 53  
3.9. Kinematic configuration of the iCub ....................................... 54  
3.10. Experimental communication set-up of the robot .......................... 57  
3.11. Robotic setup where the experimental validation of this work was performed .. 58  
3.12. Conceptual diagram of the modules implemented in our system for the understanding of human activities ........................................... 59  

4.1. General diagram of the perception techniques evaluated in this work ...... 63
4.2. Architecture of the ISA neural network ........................................... 66
4.3. Extension of the ISA neuronal network architecture for video data ........ 67
4.4. Principal steps involved in human motion recognition methodology ....... 68
4.5. Examples of the sandwich making scenario of two subjects ................ 70
4.6. Full joints space of the RAMSIS human model ............................... 72
4.7. Scenes from the TUM kitchen data set ........................................... 73
4.8. Example of the observed trajectories of one person during setting the table . 74
4.9. Overview of the methodology used to cluster the trajectories using DTW and k-means ................................................................. 76
4.10. Clusters of reaching trajectories for the motions of the human wrist from the TUM Kitchen Data Set ................................................. 77
4.11. Results of the obtained clusters and subclusters ................................ 78
4.12. Different places where an object, e.i. a cup, is being grasped .............. 79
4.13. The first levels of abstraction proposed to recognize human complex activities 80
4.14. Different levels of abstraction proposed in this work ........................... 81
4.15. Trajectory analysis of velocity produced by the same activity ............. 82
4.16. Trajectory analysis of distances between the hand and the detected object produced by the same activity ............................................ 83
4.17. Pancake making scenario evaluated with the ISA algorithm ................. 88
4.18. Expected outcome of the perception module using a simple Color-Based method 89
4.19. Color-Based pipeline to detect and track objects from an image ............ 90
4.20. Examples of the velocity signals produced by both hands during pancake and sandwich making ..................................................... 95
4.21. Examples of two distance thresholds to determine the object properties .... 96
4.22. Plots of the original and clustered hand trajectories for the pancake making scenario ....................................................................... 97
4.23. Plots of the original and clustered hand trajectories for the sandwich making scenario .......................................................... 98
4.24. Examples of the obtained results from the Color-Based perception module .. 99

5.1. Semantic representation using two levels of abstraction ....................... 105
5.2. Examples of three implemented submodules: 1) Action Recognition, 2) Semantic Rules, 3) Action Understanding ................................. 107
5.3. Attributes used during the training process to generate a decision tree ....... 110
5.4. Proposed pipe-line to infer human activities using semantic representations . 112
5.5. Obtained tree for basic activities ....................................................... 113
5.6. Obtained tree for basic and granular activities ........................................ 116
5.7. Illustration of the obtained decision tree using classical approaches ........ 118
5.8. Different data sets used to test the obtained semantic rules ................. 121
5.9. Different styles of preparing a sandwich .................................................. 122
5.10. Graphical results of different people (male and females) making sandwiches under two conditions: normal and fast ............................ 124
5.11. Recognition results of external people for the sandwich making scenario . 125
5.12. Confusion matrices of the left hand recognition for the cutting the bread scenario and the right hand recognition for the pouring the pancake mix scenario 128
5.13. On-line generated signals and the immediate inference of human activities for the sandwich making scenario in a fast condition .................. 129
5.14. On-line generated signals of the hand and object detected for the pancake making scenario as well as a snapshot of the current output of the system . 130
5.15. Recognition of the flipping activity ....................................................... 131
5.16. Recognition of parallel activities using both hands for the pancake making scenario ........................................................................ 131
5.17. Results of recognition of the activities made by: a) Human experts, b) Non-expert humans, c) the color-based method and c) the ISA algorithm for the data sets of pancake and sandwich making ........................................ 132
5.18. Knowledge framework example ............................................................... 135
5.19. Illustration of the two types of computables: Computable Classes and Computable Properties ................................................................. 136
5.20. Comparison between the KnowRob ontology and our improved ontology . 138
5.21. Illustrative example of the semantic description between activities and objects ................................................................. 139
5.22. Examples of the inferred activities and highlighted objects on the semantic layer 141
5.23. Input signals to our semantic reasoning engine ....................................... 144

6.1. General overview of the steps needed to transfer the observed task into the iCub robot ................................................................. 149
6.2. Illustration of all the implemented applications, modules, libraries and the communication between them ............................................. 151
6.3. Data flow between the processes through Yarp, iCub and external libraries . 155
6.4. Main applications implemented on the iCub to infer the human activities from observations ............................................................... 157
6.5. Illustration of the applications, modules and libraries used to perform the executePrimitive() function ............................................. 158
6.6. Integration of the system into the control block of the iCub ........................... 162
6.7. Examples of the activity recognition and execution made by the robot for the
    *pancake making* scenario ..................................................... 163
6.8. Overview of our approach to combine external and egocentric views ............ 166
6.9. Examples of the problems faced with the egocentric view for the *pancake
    making* and *sandwich making* scenarios .................................. 167
6.10. Sample model templates used to recognize the knife from the gaze camera .... 168
6.11. Examples of the *objectSeen* recognition for the *pancake making* scenario . 168
6.12. Tree obtained from the *sandwich making* scenario ($T_{sandwich}$) using the object
      properties: *ObjectActedOn* and *ObjectSeen* ............................. 169
6.13. Confusion Matrix of the inferred activities for the *pancake making* scenario
      obtained using the information from the external cameras ................. 171
6.14. Obtained results from the inference of human activities with and without gaze
      information ................................................................. 172
6.15. Confusion Matrix of the inferred activities for the *pancake making* scenario
      obtained using the information of the external cameras and gaze .......... 173
6.16. Examples of the activity recognition and execution made by the robot with the
      inclusion of the gaze information ........................................ 176
6.17. Experiments executed on the iCub showing examples of the failure and correct
      recognition of human activities with and without gaze input ............. 177
6.18. Overview of our approach using as input virtual environment signals .......... 178
6.19. Principal components of the SIGVerse software .................................. 181
6.20. Segmentation results from the SIGVerse signals ................................... 182
6.21. Illustration of the recognition of human activities for the *cleaning* task ..... 183
6.22. Examples of the robot recognition of a virtual robot and its execution ...... 186

7.1. Examples of the transference of *experiences* between scenarios achieved in
      this work ............................................................... 191
7.2. Examples of predefined paths that humans follow for the *pancake making*
      and *sandwich making* scenarios ......................................... 198

A.1. Framework to imitate human reaching motions using physically inspired opti-
      mization principles. ....................................................... 200
A.2. Comparison of tracked wrist and elbow positions and trajectories computed
      from the optimal combination of cost functions. .......................... 202
A.3. RMSD between observed and optimal trajectories for some basic llp cost functions and the optimal combination .................................. 203
A.4. Comparison of robot wrist trajectory and human wrist trajectory. ............ 204
A.5. Human and iCub executing a reaching motion for a placemat ................. 205
List of Tables

2.1. Summary of the analyzed related work on human activity recognition . . . . 35

3.1. D-H table for the robot torso and right arm . . . . . . . . . . . . . . . . . . 56

4.1. Confusion matrix results in % for high-level activities from the sandwich making scenario using ISA ................................. 71

4.2. General definition of low-level hand motions and object properties ........ 85

4.3. Confusion matrix results of low-level motions in % for pancake and sandwich making scenarios using ISA method ........................ 87

4.4. Confusion matrix results of low-level motions in % for pancake and sandwich making using the Color-Based technique .................... 98

5.1. Confusion matrix (expressed in %) for high-level activities using labeled data from the sandwich making scenario ............................. 117

5.2. Summary of the advantages of our approach compared to classical ones to compute trees ................................................................. 120

5.3. Confusion matrix (expressed in %) of high-level activities from the set the table and pancake making scenarios ............................. 121

5.4. Confusion matrix in % of high-level activities from the sandwich making scenario using ISA results .................................................. 126

5.5. This table summarizes the recognition results of the ISA implementation without and with our proposed system ................................. 127

6.1. Average activity life time in Frames for the pancake making and sandwich making scenarios ......................................................... 164

6.2. Average activity life time in Seconds for the pancake making and sandwich making scenarios ......................................................... 164

6.3. Analysis of the ObjectActedOn and ObjectSeen results (in frames) for the pancake making scenario ............................................. 172
6.4. Analysis of the *ObjectInHand* and *ObjectSeen* results (in frames) for the *pancake making* scenario .................................................. 174

7.1. This table summarizes the advantages of our system compared with the analyzed related work on human activity recognition .................. 197
# List of Algorithms

4.1. k-means clustering algorithm. ................................................. 75
4.2. General Color-Based algorithm. ............................................. 91
4.3. Hand tracking and segmentation algorithm. ............................... 92
4.4. Object properties recognition algorithm. .................................. 93

5.1. Inferring human activities ..................................................... 111
5.2. General representation of inputs and outputs of the knowledge and the reasoning system. ............................................. 137

6.1. Implementation of the `threadInit()` function. ............................ 153
6.2. The `run()` method within the thread class. ............................... 154
6.3. Implementation of the function `inferActivity()` .......................... 156
6.4. Definition of `getActivity()` algorithm. .................................... 156
6.5. Description of the function `executePrimitive()` .......................... 159
6.6. Skill planner of `reaching` ..................................................... 160
6.7. Definition of the function `doPour()` ....................................... 161
6.8. Switching between `ObjActOn` and `ObjSeen`. .............................. 175
6.9. Determine `ObjActOn` and `ObjInHand`. .................................... 185
Humans have amazing capabilities to learn new skills by extracting and fusing new information from the demonstrations. We can integrate and adapt the new information into our previously learned models using our cognitive capabilities, for example: perception, reasoning, prediction, learning, planning, etc. In other words, we are able to adapt to new situations since we re-use the learned models to infer unknown activities instead of just reproducing the observed behaviors. This is possible since we understand what we do and why we are doing it. Namely, we extract the semantics of the observed behavior. Then, the ideal scenario is to transfer such capabilities to robots; hence they can better learn from us. This vision represents a big challenge and the main inspiration for this work.

This chapter presents in Section 1.1 the motivations of pursuing this research. Then, Section 1.2 describes the challenges and problems addressed in this work. Afterward, Section 1.3 briefly explains the key points of our proposed system. While, Section 1.4 expresses the main contributions of this thesis. Finally, section 1.5 explains the outline of this document.
1.1. Motivation

One of the main purposes of humanoid robots is to improve the quality of life of elderly and/or disabled people by helping them in their everyday activities. In other words, robots need to interact with humans in a meaningful, flexible and adaptable manner, specially when new situations are faced. Hence, robots need to be as autonomous as possible to perform a wide variety of complex tasks in real human scenarios (see Fig. 1.1). Autonomous robots are expected to gain information about the environment, work without human supervision/intervention, learn new knowledge, etc. Additionally, the robot has to coordinate its actuators to produce an expected behavior, while considering any kind of possible obstacles and constraints.

![Multiple sources of observable information](image)

**Figure 1.1.** This figure depicts the motivation behind our research. Different sources of observable information are shown. The robot should be able to interpret and learn new tasks, while preserving its previous experiences. The learned model should not be restricted to a unique robotic platform, rather be transferable among different systems.

Another very important aspect that robots should have is the capability of transferring the learned behaviors regardless the source of information, e.g. observing a human performing a task, a robot demonstrating an activity or an animated character simulating the desired behav-
ior. This means that the learning methods need to consider the possibility of multiple sources of information as an input, for example: videos, virtual environments, kinesthetic demonstrations, wearable devices, motion sensing devices, etc. In other words, a learning mechanism that considers distinct sensory information can improve the transference of knowledge and experiences between different agents. Consequently, these mechanisms can not be restricted to a unique source of information, even though they capture the same event from multiple perspectives or in different formats, i.e. redundant information.

If robots are supposed to learn and interact with humans in a meaningful manner, the next foreseeable challenge for the robotics community is the semantic understanding of human activities, i.e. enabling robots to extract and determine higher level understanding of the perceived activities. The ability to automatically recognize human behaviors and react to them in a proper manner according to the human expectations will enrich humanoid robots substantially.

In summary the following points motivate our work:

- Robots that can learn a new behavior from any kind of observation and reproduce it as a new skill.

- Generate a meaningful interaction between humans and robots via semantic understanding of perceived activities.

### 1.2. Problem description

In general, when humans learn new tasks, the principal source of information is the observation, which could be obtained from different sources, for instance: from Internet videos, from observing another human, from observing a virtual agent, etc. That visual information contains data from the environment such as: body postures/motions, object properties, gaze analysis, exemplified written instructions with drawings, etc. (see Fig. 1.2). This means that the obtained information can not be processed in the same form due to the fact that each of the input sources contains different points of view of the observed task. Then, the implemented techniques to process the visual input strongly depend on the type of observation, either they focus on analyzing the color of objects, or the gaze, or the trajectories or the mapping. This leads to the first challenging problem addressed in this work, which is, combining these multiple information domains to correctly identify human behaviors. Robots with cognitive capabilities should be able to deal with the above problem, specially for transferring learned skills among different situations and between different robots.
**Figure 1.2.** This figure depicts the different pre-processing outcomes from observable inputs. Notice that each source gathers different information.

Then, the correct interpretation of the observed information is highly complex and challenging and not only depends on the analyzed source of information, but also on the variations of the observed task. For example, if I prepare a pancake in my kitchen, then I may follow a predefined pattern [Wolpert and Ghahramani, 2000]. On the other hand, if I prepare a pancake in my office’s kitchen under time pressure, then I will follow another pattern even though I execute the same task. Then, the execution of a similar activity could be performed in many different forms depending on the person, the place or the constraints, i.e. everybody has its own style to perform a desired activity for achieving a similar goal.

As a consequence, the observed patterns are sometimes defined by different parameters whose values are not unique, even when they represent the same activity. Fig. 1.3 depicts some examples of the parameters that can be obtained from the executions of a similar activity [Kunze et al., 2011]: the position of the pancake mix over the stove, different speeds to perform the activity, how much force needs to be exerted to open a bottle, the inclination of the bottle to pour its content, how much waiting time is needed before flipping the dough, etc. However, some of these parameters can not be observed, therefore extra sensors are needed, such as: RFID, magnetic sensors, force sensors, etc.

Using observable information to imitate human activities is a very challenging and difficult
problem since this implies that an artificial system (robot) should be able to 1) perceive the relevant aspects of the activity, 2) use that information to infer the activity of the demonstrator and 3) reproduce the demonstrator’s behavior to achieve a similar result. This means that robots should be able to extract and determine higher-level understanding of his/her demonstrator. In other words, it should interpret in an intentional level the effects between objects and motions involved in the human task.

In summary, there are a several problems that are detected and analyzed in this thesis:

- Design a system that interprets observed information from multiple sources, e.g. videos, gaze, virtual environment, etc.

- Define a method that efficiently learns and understands everyday human activities from observations.

- Design a flexible, adaptable, reliable, generalizable, fast and scalable framework capable to accommodate new real-life situations into meaningful skills and transfer these new skills from humans to robots.

- Equip robots with on-line capabilities of activities recognition based on semantic understanding of these activities.

If a robotic system shows the above capabilities, then it will be able to learn with higher flexibility than the classical approaches. Since it will be able to reproduce the observed activity by understanding the human behaviors and generating motions according to his/her expectations.
1.3. Semantic approach to infer human activities

In this work, we propose a framework that combines several observable inputs together with suitable reasoning tools to properly interpret, learn and understand human behaviors from demonstrations. Then, we propose to obtain more meaningful information from the observed data to recognize human activities in complex scenarios (see Fig. 1.4). Namely, our system will not learn the demonstrated behaviors using a specific source of information, but rather it extracts the meaning of such activities in order to re-use the learned behaviors into new scenarios and correctly execute the expected behavior.

![Diagram showing the process of robot understanding by observation](image)

**Figure 1.4.** This figure depicts the different steps of our proposed system.
In order to infer the desired goal, first we need a perception module that can determine which aspects of the activity are important or relevant to define. The challenging problem is to extract the (visual) information obtained from different sources, e.g. body movements, changes in the environment, the gaze of the demonstrator, etc., and transform that input into meaningful information for the system (see Fig. 1.4). However, complex everyday activities such as: reach, take, release, cut, pour, etc., can not be easily classified from simple observations due to the variance of the visual features and the redundancy that these activities present. Therefore, they require an additional inferring process which uses simple visual features to deal with this complexity.

Hence, it is necessary to design a method that can process the observed information to make inferences about the goal of the demonstrator. One powerful tool to accomplish this is through semantic reasoning and an appropriate abstraction of the problem. The first part of this work is devoted to provide a solution for both problems, by means of extracting the semantics of the observed behavior.

Finally, in order to transfer the observed human behaviors into robotics systems, it has been shown that it is more useful that the robot extracts the goal of the demonstrator, rather than purely copying the human’s actions [Nehaniv and Dautenhahn, 2007]. This allows the robot to evaluate and then decide the best way to achieve the goal, taking into account its own constraints, i.e. whether to use the demonstrator’s means or its own means to achieve the same outcome. In this case, the robot needs to determine which aspects of the inferred task should be performed in order to achieve a similar task as the human. Furthermore, our system has been fully implemented on a humanoid robot (iCub) to experimentally validate the performance and the robustness of our system during on-line execution of tasks integrated in the control loop of the robot.

In this thesis, we present a new methodology that accounts for the extraction of observed human behaviors with an estimation of the intended activities, using an automatic segmentation and recognition of these activities. This estimation will enhance the synthesis of robot behaviors using a reasoning engine based on the generated semantic representations (rules). It is important to mention that the obtained rules are preserved even when different kind of scenarios are observed and evaluated. Specifically, we enable robots not only to recognize human activities but also to understand what it is observing.
1.4. Contributions

It is worthwhile mentioning that our proposed framework is able to transfer the learned task into different situations and it can be also used in different robots. This is possible since the extracted representations of the observations are given in an abstract form which allows a better level of generalization of the demonstrated tasks. This property additionally represents a major advantage compared with classical approaches where the task is learned for a specific scenario or a particular robot. In this cases, the generated model only represents the trained task, therefore the generalization of the model to include different tasks is not possible.

The key contributions of this thesis are:

- A modular perception system to extract the relevant information from observed human activities obtained from different sources.

- An assessment of the current state-of-the-art techniques that use different sources of visual observations. Demonstrating that the current techniques are limited and depend on the training task.

- A multilevel framework has been designed and implemented to provide an automatic segmentation and recognition of human behaviors from observations using semantic reasoning tools. Our presented system is flexible and adaptable to new situations due to the re-usability of the learned rules, which allows the integration of new behaviors.

- A user-friendly imitation system to transfer the observed behaviors from humans to robots is presented. This system integrates and evaluates multiple scenarios using different perception modalities (with different levels of complexity) to demonstrate that our system is not task dependent.

We assessed our system quantitatively and qualitatively and the obtained results show that our system is able to deal with: multiple sources of observable information, time restrictions, different execution styles of the same task, dynamic variability of the observations, among others. This means that the rules obtained from one scenario are still valid for even new situations; thus demonstrating that the inferred representations are not depended on the task used for training. The results show that our system correctly recognize complex human behaviors above 85%, which is even better than a human recognizing another human activities (about 79%). This demonstrates that the semantic reasoning of human observations can provide more powerful tools for robots to learn from demonstrations.
Beside the research contributions we also provide two real-world data sets that are publicly available in the following link: http://web.ics.ei.tum.de/~karinne/DataSet/dataSet.html. This data sets provides synchronized information of 3rd. and 1st. view perspectives from several cameras in different scenarios (see Section 3.3).

1.5. Thesis outline

This dissertation consists of seven chapters including this one. The outline of the next chapters is the following:

Chapter 2. Related work. We present the revision of the state-of-the-art of the different topics used in this thesis, such as: action segmentation and recognition from videos; trajectory level analysis; semantic representations approaches; and skill transfer from human to robots. At the end of this chapter we present an example of the problem we analyzed in the rest of this document and the advantages and disadvantages of the state-of-the-art techniques used to solve this problem.

Chapter 3. Framework based on semantics for human activity recognition. We explain the importance of our multi-level system describing the key modules of our framework. Additionally, we explain the set-ups and tools used to record the new data sets to train and test our framework. At the end of this chapter we explain the robotic set-up used to experimentally validate our system.

Chapter 4. Perception and extraction of visual features (an evaluation). This chapter introduces different approaches typically used to extract visual features from videos. We will demonstrate that even using a state of the art technique it is very difficult to recognize the desired activities without the use of a semantic representation. Then, at the end of this chapter we present our proposed levels of abstraction. After that, we present the enhancement of the evaluated techniques for the same test-cases using our proposed abstractions.

Chapter 5. Inferring Semantic Human Activities Reasoner (ISHAR). This chapter represents the core of this dissertation, which will explain the methodology used to obtain the semantic rules. Afterward, these rules will be tested under different constraints. At the end of this chapter we will introduce the integration of our semantic rules into a knowledge base that will improve the dynamic grow of the ontology-base knowledge representation using the obtained semantic rules into the reasoning engine.
Chapter 6. Assessment and integration of our framework in complex scenarios. We show the robustness of our framework by evaluating it on the iCub Humanoid robot. This validation complements our system by proving that we can achieve on-line decision making into a robotic system. Afterward, we assessed our framework using more complex situations with new observable inputs, for instance instead of having a 2D image, now we tested our system with 3D signals coming from Virtual Environment scenarios. Then, we show the adaptation of our system and its flexibility toward these new constraints.

Chapter 7. Conclusions. Finally, we present the last remarks of this work as well as some further discussions of our presented framework and contributions. We also explain some future improvements on the introduced system.
Chapter 2

Related work

Currently, there are several methods proposed to solve the problem of human activity recognition from demonstrations. The majority of these methods strongly depend on the analyzed task. That implies that the exchange and generalization of the obtained models between different tasks would not be possible even when a similar problem is analyzed, since the models are restricted to the trained input data. In this work we considered the advances made in multiple disciplines to address the recognition problem specially for the robotics domain.

This chapter presents in Section 2.1 the main areas investigated in this work. Then, Section 2.2 presents the state-of-the-art techniques to recognize human activities from videos and the typically data sets used in the Computer Vision community. Section 2.3 introduces the trajectory level techniques widely used in the Robotics community. Afterward, Section 2.4 explains the recent methods to map the continuous events into symbolic representations to extract the meaning of human motions. Section 2.5 introduces a case study to show the advantages and disadvantages of the presented techniques. Finally, 2.6 presents the summary of this chapter.
2.1. Research areas investigated

In this research, we are dealing with problems of segmenting, recognizing, understanding and transferring human activities from observations to robots. These problems have interested the researchers from different disciplines, therefore several methods have been proposed to solve a subset of these problems. The following disciplines have made very interesting advances regarding the problem of interpreting and learning the human behaviors: Computer Vision [Poppe, 2010; Le et al., 2011; Patterson et al., 2005], Artificial Intelligence [Aggarwal and Ryoo, 2011; Beetz et al., 2010; Park and Aggarwal, 2004], Cognitive Science [Koppula et al., 2012; Wörgötter et al., 2009; Aksoy et al., 2011; Vernon et al., 2007], Robotics [Kuniyoshi and Inoue, 1993; Inamura and Shibata, 2008; Billard et al., 2008], Neurologists [Hermens and Gielen, 2004] to name a few. These disciplines use different approaches such as trajectory-level, feature-extraction, and/or semantic-representations, as shown in Fig. 2.1.

![Figure 2.1. Main research areas investigated for learning human activities from observation. We observe that they are mainly three: trajectory level, feature extraction and semantics.](image)

Learning and understanding human activities can greatly improve the transference and generalization of the acquired knowledge among different robotic platforms such as: service robots, humanoid robots, industrial robots, social robots, etc. (see Fig. 2.2). These robotic platforms have different embodiments and different cognitive capabilities, therefore the trans-
ference of trained models from one platform to another is not straightforward and typically
the obtained models work fine with the tested platforms only. In general to achieve a correct
transference a new training phase is required [Le et al., 2011; Azad et al., 2004; Billard et al.,
2004]. However, if instead of learning how the motions are executed, we learn the meaning of
such movements, then we can transfer these learned models to different situations and among
different robotic platforms as proposed in this work. This can be done by understanding what
the demonstrated behavior is, and why it has been executed.

Fig. 2.2. Examples of the different levels of cognition in different robotic platforms. First,
a) and b) present the systems with less cognitive capabilities, mainly industrial robots.
Whereas c) and d) show the robots with more cognitive capabilities, used typically in ser-
vice/care robotics.

The challenging problems that need to be addressed in order to allow a better autonomy
of robots are the following: segmenting, recognizing, understanding and transferring human
behaviors from observations. Each of those problems are being solved for different purposes
such as: surveillance systems [Poppe, 2010; Schuldt et al., 2004], monitoring of patients [Ag-
garwal and Ryoo, 2011; Kanda et al., 2009], anticipation of human behaviors for assistance
[Koppula and Saxena, 2013a,b; Lei et al., 2012], etc.

The goal of human activity recognition expressed by Aggarwal and Ryoo [2011] is to “au-
tomatically analyze ongoing activities from an unknown video”. The authors also identified
various types of human activities. These are categorized on different levels depending on their
complexity such as: gestures, actions and interactions. For example, gestures refer to atomic
and general human movements such as raising a leg or stretching an arm [Aggarwal and
Ryoo, 2011]. Actions refer to single person activities that could temporarily include multiple
gestures, for example: walking, running, waving, etc. [Schuldt et al., 2004]. Interactions are human activities and these involve a combination of two or more persons and/or objects, for instance, answering a phone, hand shaking, cutting a bread, etc. [Marszalek et al., 2009]. The last category represents the most complex activities for recognition and in this work we will focus on them.

The used approach depends on the complexity of the analyzed activities and several challenges have to be addressed, for instance: automatic segmentation of human motions [Dillmann, 2004; Taniguchi et al., 2011], identification of important features of the motion [Krüger et al., 2010], definition of the importance of the object(s) to the task [Philipose et al., 2004], as well as the definition of different levels of abstraction [Ikeuchi et al., 1993]. One of the main issues about those problem domains is that in order to translate the methods from one area to solve a similar problem in another area is not straightforward. In other words, the recognition of human activities is still far from being an off-the-shelf technology [Aggarwal and Ryoo, 2011]. Then, how can we transfer the findings and advantages of one technique to solve a similar problem on a different domain?

2.1.1. Computer Vision and Machine Learning communities

The problem of recognition from observing a video has been widely addressed by the computer vision and machine learning communities. They are focused on solving the problem of activity recognition by identifying the important/characteristic features from images, such as spatio-temporal volumes [Rodriguez et al., 2008a], space-time trajectories [Rao and Shah, 2001; Lee et al., 2011], space-time local features [Yilmaz and Shah, 2005], among others. In general, the main assumption is that the segmentation of such actions is given mostly manually, i.e. the training and testing phases are evaluated on manually labeled images. This indicates that this community is focusing on solving the so called simple case, as expressed by Aggarwal and Ryoo [2011], where the video has been manually segmented and contains only one demonstration of the human activity. In this cases, the main goal is to correctly classify the separated videos into its corresponding category, which is not an easy task specially if they analyze the motions of the human body [Aggarwal and Cai, 1999]. Therefore, most of these techniques depend on the good quality of either images or videos. More details on these approaches will be given in Section 2.2.
2.1.2. Robotics community

On the other hand, the robotics community mainly investigates techniques on the *trajectory level* to learn and transfer human motions into robots. This is a very challenging task due to the embodiment problem [Nehaniv and Dautenhahn, 2002]. Most of the techniques used in this community require the information of the joints and/or the Cartesian position of the human, as well as several trials of the same task to learn the correct model [Bentivegna et al., 2006]. This indicates that, they look for the parameters involved in the learned task/skill mainly to recognize and replicate the observed human motions using dynamic stochastic models [Kulic et al., 2008]. This community focuses on solving more general cases, where the goal is the continuous recognition of human activities by detecting starting and ending times of the observed activities from an input video [Aggarwal and Ryoo, 2011]. For example, extracting the human pose information from the observations to correctly recognize the executed motions in a continuous manner, which implies new problems such as occlusions [Lee and Nakamura, 2014] and disturbances [Billard et al., 2006]. More details about the methods used in this community are shown in Section 2.3.

2.1.3. Artificial Intelligence community

Recent studies have focused on determining the levels of abstraction to extract meaningful information from the produced task to obtain *why* certain task was recognized [Krüger et al., 2007]. For example, Ekvall and Kragic [2006] improved the trajectory level techniques using task planning level approaches to make the system flexible. These approaches are focused on explaining the observed behaviors via *reasoning* and *inference* mechanisms [Charniak and Goldman, 1993]. This will help to understand the meaning of the recognized task and will enable the system to be flexible, robust and adaptable to new situations, for instance by parsing the observations into stochastic grammars [Barton, 1985; Pynadath and Wellman, 2000] or using knowledge representation [Gray, 1990]. This represents the area of interest of this thesis and a detail analysis of the state-of-the-art for these techniques is provided in Section 2.4. The next subsections will analyze in more detail the above topics to solve the problem of understanding human behaviors from observations.

2.2. Activity recognition from videos

Cognitive and neurophysiology studies suggest that understanding the actions from other humans requires the mapping of the observed action into motor representations of the action.
SECTION 2.2  Activity recognition from videos

[Berniker et al., 2014]. It has been proved that humans instinctively are able to implement motor commands equivalent to the ones used while executing the action [Johansson and Flanagan, 2009]. Mainly, since we naturally find this mapping by developing learning mechanisms from extracting and incorporating new perceived information from the environment. This is performed by observing other humans (3rd. view) or directly by observing ourselves while interacting with the environment (1st. view experience) or a combination of both. In the neuroscience domain, it has been demonstrated that for controlled scenarios, there is a robust coupling between the eye movements and the hand movements specially for planning and control, e.g. for block stacking tasks [Flanagan and Johansson, 2003]. Then, the challenge is to translate that eye-hand coordination into robotic systems to achieve a similar behavior. Therefore, the next subsections extend the analysis of the related work using external or egocentric views.

2.2.1. External observations

Recognizing human activities is currently an active research area in Computer Vision, where the local image representation is considered as a promising direction compared to the global representations. The first one addresses visual occlusion and generalization to different scenarios by taking into account spatio-temporal correlations between patches [Poppe, 2010]. However, the action analysis is focused on recognizing the movement or/and the changing of posture of humans, such as walking, running, swinging, etc. [Le et al., 2011]. The analysis of human activities is typically done on the four well known benchmark datasets: KTH [Schuldt et al., 2004], Hollywood2 [Marszalek et al., 2009], UCF sport [Rodriguez et al., 2008b] and YouTube [Liu et al., 2009]. A subset of the activities typically analyzed on these benchmark data sets is depicted in Fig. 2.3. Those data sets are mainly used for the recognition of human activities, but not for the segmentation of the activities. This means, the recognition is done on manually segmented videos, since the main goal is the recognition and not the segmentation of the video, which are two different and challenging problems.

From Fig. 2.3 we can observe that usually the action classes that have to be recognized from the videos are: driving a car, eating, hand shaking, hand waving, running, golf swinging, etc., where the highest accuracy of recognition reported is around 75% using the Independent Subspace Analysis (ISA) algorithm [Le et al., 2011]. Nevertheless, if we take a closer look at the activities that interact with object(s) such as: basketball shooting, golf swinging, answering the phone, etc., then we can notice that the accuracy of recognition decreases significantly (in average 43.62%).

That shows how challenging is the automatic recognition from videos of those kind of activ-
Figure 2.3. This figure depicts some examples of activities typically analyzed on the benchmark data sets. We can observe general activities as well as goal-directed such as answering the phone or kicking/shooting a ball. This figure has been adapted from [Schuldt et al., 2004; Marszalek et al., 2009; Rodriguez et al., 2008b; Liu et al., 2009].

For these activities is difficult to determine which features from the video are important and relevant. Mostly, this is due to the fact that the features for detecting the same activity may change in different environments, for example observe from Fig. 2.3 the activity riding a horse, from the data set UCF, has a different environment than the same activity presented in the data set YouTube. Additionally, in some cases the information of the human posture is the most important feature for the recognition, for example in the activities shown in the data set KTH, where the posture clearly defines a given activity. Furthermore, there are some activities that depend on recognizing the manipulated object in order to identify the correct activity, for example answering the phone from the Hollywood2 data set. Then, if we want to recognize even more granular activities such as the ones proposed in this work, e.g. reaching, taking,
cutting, flipping, etc. (see Fig. 2.4) it will be very hard using a single feature. The identification of such kind of activities is a very complicated problem and we can not expect accurate results even when we use the state-of-the-art methods for action recognition from videos.

There are several methods developed in the machine learning community to solve the learning activities problem using classification learners, such as: Decision Tree learners [Quinlan, 1993] or Support Vector Machines [Vapnik, 1998], function approximators, such as: Artificial Neural Networks [Haykin, 1999], Sequence Learning algorithms, and Reinforcement Learners [Mitchell, 1997]. These techniques determine optimal selection strategies for uncertain situations. The learning algorithms are complemented by more general approaches such as data mining and integrated learning systems [Beetz et al., 2008]. Nevertheless, these techniques are not as accurate, fast and robust enough for the particular case of automatically detecting and segmenting human activities from observations.

Recent machine learning researches in the Deep Learning domain showed that a learning-based feature extraction algorithm is more effective than a hand-designed visual feature algorithms such as SIFT, HOG and SURF [Lee et al., 2009]. The key advantage of the feature extraction methods is that they are able to discover unexpected features. In contrast to the hand-designed features, which is totally based on the researcher’s own heuristics. One restriction of the vision-based techniques is the definition of the viewpoints of the camera which in most of the cases is fixed and known. This severely limits the functionality of such systems specially when we want to translate these models into robots and use them in real world applications.

2.2.2. Egocentric (first-person) vision

In order to recognize and understand human activities it is important to analyze not only the human movements from one camera [Ramirez-Amaro et al., 2014a] or several external
cameras, e.g. [Ramirez-Amaro et al., 2013b] but also it is necessary to extend this analysis using the gaze information, e.g. [Noor et al., 2011; Pirsiavash and Ramanan, 2012; Ramirez-Amaro et al., 2013a; Noor and Minhas, 2014] to enhance the recognition of daily life activities for robotic systems. Then, it is important to combine the external observation with the self-observations to improve the recognition of human activities.

Neurophysiological evidence suggest that the gaze information is a very strong indicator to predict the next contact point/place in action observation. For example, Johansson and Flanagan [2009] demonstrated that the gaze of humans arrived at a given contact zone before the hand, i.e. the input of the visual gaze led the hand motion. Then, Kurby and Zacks [2008] measured the eye movement during the segmentation of movies into events and they proved that the saccades were more frequent around event boundaries, which reflect the re-orientation of the viewers at the beginning of a new event. Similarly, Salvucci and Goldberg [2000] proposed a taxonomy of a fixation identification algorithm of the eye-movement to classify spatial and temporal information in eye-tracking protocols.

The correct identification of the fixation of the eye significantly reduces the size and complexity of the eye-movement protocol, removing raw saccade data points and collapsing raw fixation points into a single representative tuple [Salvucci and Goldberg, 2000]. This reduction is useful for at least two reasons. First, little or no visual processing can be achieved during a saccade, and thus the actual paths traveled during saccades are typically irrelevant for many research applications. Second, smaller eye movements that occur during fixations, such as tremors, drifts, and flicks, often have less importance in higher-level analysis. Thus, fixation identification is a convenient method of minimizing the complexity of eye-tracking data while retaining its most essential characteristics for the purposes of understanding cognitive and visual processing behaviors.

On the other hand, the Computer Vision community has focused on the visual input of the human gaze. For instance, Pirsiavash and Ramanan [2012] presented a first-person camera view data set to detect human everyday activities. The authors develop composite object models to deal with the fact that objects look different when they are manipulated, specially since their appearance may change. However, the visual data come from the chest instead of the human’s gaze, which does not provide information of gaze attention of the people. Another first-person analysis through a wearable camera was presented by Spriggs et al. [2009], where additionally they include Inertial Measurement Units (IMUs) to temporally segment human motions into actions in the context of cooking scenarios. However, they deal with a high-

---

1The authors used a GoPro wearable camera typically used for athletes during sport events mounted on the chest of the participants.
dimensionality problem which can not be solved with Hidden Markov Models (HMM) and Gaussian Mixture Models (GMM). Recently, Fathi et al. [2012] modeled the spatio-temporal relationships between the gaze point, the observed objects and the labeled action using probabilistic models from egocentric cameras. The authors were able to predict the best sequence of gaze locations which improve the action recognition assuming that the object classifiers were pre-learned. However, they also conclude that using the egocentric view is not enough to robustly detect everyday activities due to the low accuracy of recognition (average 47%).

The Carnegie Mellon University data set (CMU-MMAC) [De la Torre et al., 2009] contains information of 5 external cameras and 1 wearable camera which captures the 3rd. and 1st. views respectively. The wearable camera does not really capture the information of the subject gaze, since the camera is mounted on a helmet. This makes the camera’s point of view above the current view of the human, which makes the video difficult for further processing of human gaze. Interestingly, there has been very limited research done in this data set, either using the manually labeled data from the one external camera [Tenorth et al., 2013b], or using only the egocentric data [Spriggs et al., 2009]. However, there has not been any analysis of this data set that combines the information from the external and gaze cameras.

To the best of our knowledge, the combination of both sources has not been done before. Therefore, in this work (see Section 3.3), we introduce a data set that contains the Outside-in view (obtained through 3 external cameras) with the Inside-out vision (obtained via Gaze-based camera). We aim to provide camera information that can align the first-person perspective with that of third person to enhance the understanding of human behavior and this data set is freely available.

### 2.3. Trajectory level

The robotics community indicates that understanding and transferring skills to humanoid robots from observing human activities is well considered to be one of the most effective ways to increase the capabilities of such systems [Schaal, 1999; Nehaniv and Dautenhahn, 2007]. With the recent advancements of sensory technologies (such as Kinect), perceiving reliably human behaviors have become tenable [Xia et al., 2011].

As mention previously, the trajectory level methods are mostly used by the robotics community and they develop systems based on the observed human pose, i.e. the Cartesian and Joint spaces, to segment and learn the correct model of the human motions [Takano and Nakamura, 2012]. Then, they learn and extract the parameters involved in that task/skill from the demonstrations typically using several trials of the same task and mostly for one specific scenario.
Related work

[Atkeson and Schaal, 1997b; Calinon and Billard, 2007; Bentivegna et al., 2007]. For example using Programming-by-Demonstration (PbD) techniques [Billard et al., 2008], which are powerful and well-established mechanisms used widely in the robotics community to teach robots new activities from observations.

One significant challenge of the teaching technique PbD is to correctly identify and answer the question: what to imitate? [Nehaniv and Dautenhahn, 2007]: a) similar motion or b) the meaning of the motion. Regarding the first point, an interesting approach presented by Billard et al. [2008], identifies a general policy for learning relevant features of the task. They identified what to imitate by detecting the time-invariants of the demonstrator [Billard et al., 2004]. This method is mostly used to teach robots new activities due to its robustness, however does not answer the second question. The mapping between the robot and the demonstrator motion is a very difficult problem known as the Correspondence Problem, since in general the robot and the human have different embodiments [Nehaniv and Dautenhahn, 2002]. Therefore, in the cases such as the ones analyzed in this work the method PbD is not suitable due to the fact that it requires the Cartesian and Joint information, which is not possible to obtain from videos, specially in the case where the parameters of the camera are unknown.

Recently, Ude et al. [2010] presented the idea of using a library of Dynamic Motion Primitives (DMPs), which enables the generalization of DMPs to new situations. The advantage of this approach is that it takes into account perturbations and includes feedback [Ijspeert et al., 2002]. This means that the obtained relevant parameters identify and reproduce similar motions to those of the demonstrator, also similar motions are adjusted based on the parametrization of the given goal. In such cases, the goals are generally given manually, i.e. the system (robot) will not be able to extract the meaning of the action due to the fact that they do not possess reasoning capabilities. Therefore, the transference of the obtained models to new situations will not be straightforward and generalizable. Additionally, such techniques require sophisticated visual-processing methods to extract human trajectories [Azad et al., 2004]. However, these kind of techniques are usually used by roboticists in the context of imitation learning, even though the proposed techniques need considerable time to finally learn the specific task since they need multiple demonstrations [Bentivegna et al., 2006].

The advantages of learning from demonstration are shown in [Atkeson and Schaal, 1997a], where pole-balancing was accomplished after one trial. Two types of learning strategies in the context of robot control have to be distinguished: Motion Imitation, where the observed human state is directly mapped onto the imitating robot, as in [Atkeson and Schaal, 1997b; Do et al., 2008; Dube and Tapson, 2009; Gribovskaya et al., 2011; Kim et al., 2006], and Imitation Learning, where characteristics found in the demonstrated motions are used to control the
robot. Two central techniques in *Imitation Learning* are *Motion Primitives*, e.g. [Mataric, 2002; Nakaoka et al., 2005], and *Markov Decision Problems*, e.g. [Calinon and Billard, 2007; Lee et al., 2008; Ratliff et al., 2006]. The basic assumption of such *Markov Decision Problems* is that the set of states is finite and that state transition probabilities are known. Under these assumptions the techniques of *Inverse Reinforcement Learning* can solve the inverse problem of finding the cost function applied in the demonstrations [Abbeel and Ng, 2004; Ziebart et al., 2008].

There have been some techniques to classify human motions based only on the shape of the trajectory, without taking into account the object information, e.g. using similarity measurements like Dynamic Time Warping [Albrecht et al., 2011] or using a multi-step hierarchical clustering algorithm motivated by CART (Classification and Regression Trees) [Nyga et al., 2011], or using Machine Learning techniques such as Expectation Maximization [Ramirez-Amaro and Beetz, 2010]. These later techniques rely on the generation of trajectories depending on the location of the objects, this implies that if a different environment is being analyzed then the trajectories will be altered completely, thus, new models have to be acquired for the classification. Another drawback of this approach is that the segmentation of the trajectories into meaningful classes is mostly given by the researchers, i.e. it is done manually. Another approach is presented by Beetz et al. [2010] where a Hierarchical action model is constructed from observing human tracking data based on the linear-chain Conditional Random Fields (CRF). The CRF uses as training data the pose-related features such as extension, retraction or expansion of the hands beyond a certain threshold, information obtained from the environment model, for example it uses RFID and magnetic sensors to detect when the human carries an object or if a hand is near the handle of a cupboard, or if a drawer is being opened. These features are combined, and CRFs are learned on a labeled training set. Even though this latter approach considers the information of the objects, several parameters need to be adjusted in advance, which means that recognizing activities from the same class in a different environment is highly complex.

The field of describing human characteristics has various facets and is growing fast. Disciplines dealing with the analysis and description of these movements range from psychology and biology over computer and engineering sciences to mathematics. Consequently, the relations and differences between all these approaches cannot be discussed within the limits of this document; e.g. refer to reviews [Engelbrecht, 2001; Ostry and Feldman, 2003; Schaal et al., 2003].

The trajectory level techniques are very useful not only to extract the relevant information of the activities but also to transfer these models into artificial systems such as robots.
Transferring the obtained models acquired from human demonstrations to robots represents yet another challenging task while building adaptive and autonomous robots. This is mainly since it requires to generate task-specific robot motions which should fit naturally in a human environment. This indicates that the robot should generate motions based on the current task, the object type and the current state, while additionally considering information given by the current context and environment. This requires to analyze the performed human stereotypical and pre-planned motion patterns in order to achieve the desired task [Wolpert and Ghahramani, 2000]. Therefore, the major advantage of such approaches is their ability to analyze the details of human movements [Aggarwal and Ryoo, 2011]. However, such analysis can only be done with a very sophisticated perception systems to identify the pose of the human joints, e.g. motion capture system or state-of-the-art tracking systems [Bandouch et al., 2008a; Azad et al., 2004]. Another drawback of these methods is the inability to generalize the learned models toward new situations due to the fact that they depend on the correct identification of the human pose, since is difficult to extract from 2D videos, which is the case analyzed in this work.

2.4. Semantic representation methods

Recent studies focus on determining the levels of abstraction to extract meaningful information from the produced task to obtain what and why certain task was recognized. For example, hierarchical approaches are capable to recognize high-level activities with more complex temporal structures [Aggarwal and Ryoo, 2011]. Such approaches are suitable for a semantic-level analysis between humans and/or objects. Additionally, they are able to cope with less training data by incorporating prior knowledge into their representations. This prior knowledge is mostly included manually by an expert, who gives a semantic meaning to the sub-activities that compose the high-level activity. Such mechanisms help to understand the meaning of the recognized task and will enable the system to be flexible and adaptable to new situations. This represents the area of interest of our work and a more extensive analysis on such techniques are provided.

Extracting symbolic descriptions from observations have been proposed as a bottom-up approach to obtain the motion sequences [Ikeuchi and Suchiro, 1992]. However, this later method is limited since it does not consider the continuity of the human sequences. One pioneer work of high-level representations was introduced by Kuniyoshi et al. [1994], where the authors suggested to map the continuous real world events into symbolic concepts by using an active attention control system. Later, Ogawara et al. [2001] presented a framework that
integrates multiple observations based on attention points. They proposed a two-step system which observes and extracts the attention points to examining the sequence of the human motions. However, this system is constrained with a very specific input information such as sensory glove data, depth information, etc. A similar work by Jäkel et al. [2010], shows a (partially) symbolic representation of manipulation strategies to generate robot plans based on pre- and post-conditions. Nevertheless, those frameworks are not able to either reason about the intentions of the users or extract the meaning of the actions. Another work that addresses this problem is presented by Fern et al. [2002], where a logic sub-language is introduced to learn specific-to-general event definitions using manual correspondence information.

Regarding the addition of the term *semantics* into the recognition of human behaviors, Park and Aggarwal [2004] define the semantic descriptions using the linguistic “verb argument structure” in terms of $(agent − motion − target)$ triplets. This means that a meaningful semantic description for the recognition of human behavior is obtained in terms of subject-verb-object. In order to obtain these triples the authors need to associate visual features with natural language verbs and symbols from a defined vocabulary to build the semantic descriptions of video events. One advantage of the natural-language description is its rich structure of syntax and semantics that represents domain-free rules and contexts. However, the number of triples has to be defined in advance and depending on the complexity of the action, multiple triples may be needed for an specific action. Therefore, the re-usability of those triples for new situations is complicated.

From the robotics point of view, Takano and Nakamura [2006] proposed an approach to encode observed trajectories based on a Hidden Markov Models (HMMs) mimesis model in order to segment and generate humanoid robot motions through imitation. They transformed the motion patterns into location proto-symbols in a topological space, called the proto-symbol space. However, one of the weak points of this mimesis model is to find a physical meaning of each dimension of the proto-symbol space. This issue was addressed by Inamura and Shibata [2008] where a physical meaning was given by means of natural language. Even though this system enables the generation of novel motion patterns, it is limited to only use the joint angles to create the proto-symbol space. Therefore, such systems will not work with the data sets proposed in our work, since the joint angles are not available.

Another interesting definition of semantic representations is given by Turaga et al. [2008]. The authors suggested that the semantics of human activities requires higher level representations and reasoning methods. They discussed the following approaches: *Graphical Models* (Belief Networks [Gupta and Davis, 2007], Petri Nets [Ghanem et al., 2004], etc), *Syntactic Approaches* (Grammars [Kitani et al., 2008], Stochastic Grammars [Ivanov and Bobick, 2008], etc).
2000], etc), Knowledge and Logic Approaches (Logic-based approaches [Siskind, 2001], Ontologies [François et al., 2005], etc.). Therefore, the semantic definition of the activities will depend on the used approach. For example, Graphical Models such as the one presented by Sridhar et al. [2008], where a graphical model is used to learn functional object-categories. The obtained graphs encode and represent interaction between objects using spatio-temporal patterns. The taxonomy of the learned graph represents the semantic of the studied object categories mapped to a set of spatial primitives relationships, e.g. two objects are Disconnected, connected through the Surroundings (S) or Touching (T). However, in order to obtain the activity graph all the episodes need to be observed. Regarding Syntactic Approaches, for instance Context-Free Grammars (CFGs) and Stochastic Context-Free Grammars (SCFGs) have been used by previous researchers to recognize high-level activities [Aggarwal and Ryoo, 2011]. These grammars are typically used as a formal syntax for the representation of human activities. This means that these grammars directly describe the semantics of the activities.

Another direction for action recognition has been proposed through the recognition of the object(s), human motions and the effects on the objects. For example, the concept of Object-Action Complexes (OACs) introduced by Wörgötter et al. [2009] investigates the transformation of objects by actions, i.e. how object A (cup-full) changes to object B (cup-empty) through the execution of Action C (drinking)\(^2\). This approach has been recently used to segment and recognize an action from a library of OACs using the preconditions and effects of each sub-action which will enable a robot to reproduce the demonstrated activity [Wächter et al., 2013]. However, this system requires a robust perception system to correctly identify the attributes (full-empty) of the objects, which is obtained and executed off-line from motion capture systems.

Analogous to OACs and based on the Affordance Principle, Aksoy et al. [2011] presented the approach called Semantic Event Chain (SEC), which determines the interactions between the hand and the objects, expressed in a rule-character form. These interactions are based on the changes in the visual environment represented in a dynamic graph where the nodes are the center of the image segments and the edges define whether or not two segments touch each other. Then, the spatial relationships between the graphs are stored in a transition matrix which represents the Semantic Event Chain. One drawback of this technique is that it highly depends on the time and sequence of the events, and on the perception system to define the object interactions. In other words, if the computer vision system fails, then this approach will be greatly affected. A different approach based on the affordances of objects has been introduced by Kjellström et al. [2011], where the authors categorize the manipulated objects

\(^2\)This action is defined by the current attribute of object A.
and human actions in context of each other, where only one hand action is considered. They have a semantic level to represent action-object dependencies (drink-cup, drink-glass, etc.) modeled using CRF method. However, this approach requires that the training data-set has been fully and correctly manually labeled, which indicates that new unlearned behaviors can not be identified.

Recently, Koppula and Saxena [2013a] modeled the activity and object affordances to anticipate/predict future activities. They introduced an Anticipatory Temporal Conditional Random Field (ATCRF) that models the spatio-temporal relations through object affordances based on the concepts and methods introduced in [Koppula et al., 2012]. This technique has better performance than the state-of-the-art techniques analyzed in that paper, nevertheless the type of interaction will define the affordances that are being considered from a predefined library. The same authors Koppula and Saxena [2013b] proposed to sample the spatio-temporal structure in addition to the future nodes to enhance the temporal segmentation and the anticipation problem by considering multiple graph structures. However, the structure of the graphs needs to be fully known and depends on a correct temporal segmentation. We believe that the latter approach can be enhanced using our methodology to segment and infer the activities.

Another work dealing with the problem of human intentions is presented by Gehrig et al. [2011] where their framework combines the motion, activity and intention recognition of humans using visual information of a monocular camera in combination with the knowledge domain. This system is restricted to manual annotations of the domain (time of the day and the presence of the object). Also, the relationship between the activity and the motion is neglected.

The work presented by Yang et al. [2013], introduces a system that can understand actions based on their consequences, e.g. split or merge. Nevertheless, the core of this technique relies in a robust active tracking and segmentation method that it is able to detect changes of the manipulated object, its appearance and its topological structure, i.e. the consequences of the action. Later, this system was improved by including a library of plans composed of primitive action descriptions presented by Guha et al. [2013]. However, this system is not implemented in a robot and it will fail if the plan is not known a priori. Another work based on plan recognition presented by Kautz et al. [1991] states that human behavior follows stereotypical patterns that could be expressed as preconditions and effects. However, these constraints must be specified in advance, which represents a problem when trying to use them in different domains.

An Ontology-based activity recognition is presented by Patterson et al. [2005], where a model to generalize object instances through their classes using abstract reasoning is proposed.
A problem with this model is that in some cases the activities are misclassified due to the fact that those activities belong to the same class when using the same object. For example, the activities *doing laundry* and *getting dressed* are misclassified since they have the same class of object, i.e. clothes.

Concerning the recognition using knowledge representations, the work introduced by Tenorth and Beetz [2013] presents a practical approach to define robot knowledge that combines description logic knowledge bases with data mining and (self-) observation modules. The robot collects experiences while executing actions and uses them to learn models and aspects of action-related concepts grounded in its perception and action system. Nevertheless, this knowledge representation needs to be specified by the object-action relations manually.

### 2.5. Case study

Let's consider the scenario of setting the table from the TUM Kitchen Data set\(^3\), which consists of transporting kitchen items from its storing place to the table (see Section 3.4.3). An example of the setting the table task is depicted in Fig. 2.5. This scenario exemplifies the complexity and challenging problem of activity recognition for continuous and real environments. From this data set we can retrieve the visual information from the captured video, and we can also have the Cartesian and joint positions of the human skeleton. Along with information from external sources such as RFID tag readers, magnetic sensors, etc. Using the data obtained from this environment different techniques, presented in the above sections, can be analyzed in order to solve the same problem of activity recognition and its possible transference toward new scenarios. As a results, we can create a set of advantages and disadvantages of the current state-of-the-art algorithms of the different research communities. This will be presented in the next subsection.

#### 2.5.1. Advantages and disadvantages of the related work techniques

First, we will present the analysis of the techniques of feature extraction (see Section 2.2) from the video images shown in the top part (a) of Fig. 2.5. to recognize activities such as: *reach, take, release*, etc. Then, we can notice that this represents a very complicated problem since different human poses can be observed for a similar activity. For example, for the activity *reach*, the video sequences show different postures or stereotypical patterns of the human body to achieve the desired activity, such as: *stretching the arm, lowering the

---

\(^3\)The TUM Kitchen Data Set is publicly available here [http://ias.cs.tum.edu/software/kitchen-activity-data](http://ias.cs.tum.edu/software/kitchen-activity-data)
This figure shows the challenges in the human activity recognition, either from visual observation from videos (top) or by analyzing the trajectories (bottom). As we can observe from the bottom part of the figure, the motions of the right and left arm are segmented. In this case, they only represent the reaching activity.

Then, the desired visual features that need to be extracted out of the images have to focus on the targeted object which in some cases is occluded, e.g. reach for a plate from a cupboard, where the object plate is not visible when the cupboard is closed. Moreover, the features of the objects can lead to misclassified activities, since in most of the cases the same object is involved in more than one activity, e.g. reaching for a placemat and taking a placemat involved the same targeted object. Some quantitative results of this analysis are presented in more detail in Section 4.2 where the state-of-the-art algorithm on feature extraction is used to classify human activities.
On the other hand, if we use the information of the human trajectories (see Section 2.3) for the recognition of the activities (e.g. reach, take, release, etc.) during the task of setting the table, then we can observe from the bottom left part (b) of Fig. 2.5 that the obtained trajectories have different patterns as expected and these will depend on the current position of the object of interest as presented by Albrecht et al. [2011]. However, some activities such as reach and release may follow the same trajectory patterns. Further analysis on the trajectory level could lead to motions similar to the bottom right part (b) of Fig. 2.5, where all the depicted trajectories represent the patterns of both hands (end effector) of the human during the reaching activity. This indicates that the motion pattern will depend on the hand that executes the motion. Furthermore, when we take a closer look at the trajectories obtained just from the right hand, we can observe that these trajectories follow different patterns which depend on the location of the objects and their grasping positions, even though they represent the same activity, i.e. reaching as presented by Ramirez-Amaro and Beetz [2010]. Then, the classification of the human activities using only the trajectory information represents a very complicated problem, specially since the obtained trajectories contain information of the environment that can not be observed and need to be inferred such as the intention of the human. Another drawback of this approach is that the segmentation of the trajectories into meaningful classes is mostly given by the researchers, i.e. it is done manually [Tenorth et al., 2013b].

Finally, we will analyze the semantic or symbolic representation (see Section 2.4) to recognize the human activities. In other words we could use hierarchical approaches [Aggarwal and Ryoo, 2011] to combine the information obtained from the image sequences with the trajectories, which could give more accurate systems when trying to recognize human activities in real scenarios. These techniques allow to abstract the recognition problem by mapping the continuous motions into symbolic events which permit not just to recognize the human activities but also to segment them over time. Such transformations strongly depend on time [Koppula and Saxena, 2013b; Aksoy et al., 2011] and as a results they need to define the preconditions and postconditions or effects produced by the activities [Wörgötter et al., 2009; Kautz et al., 1991]. In the example depicted in Fig. 2.5 can be observed that the effects produced by the reaching activity are not always the same, for example reaching for a placemat will produce that the top of the stove gets empty and reaching for a fork will outcome that the drawer is lighter. Then, those events will be classified as two different activities even when they represent the same activity (reaching). Another disadvantage of these techniques is that they also depend on a strong and very accurate perception system that can detect the effects of the performed activity, e.g [Kuniyoshi et al., 1994; Yang et al., 2013].
2.6. Summary of this chapter

Overall, we can conclude from the studied and analyzed related work, that the proposed and existing methods work well when trying to solve a particular aspect of the problem of activity recognition. Then, a new paradigm has to be defined that can use the advantages of the current systems and overcome the disadvantages of those for solving a similar task. This is precisely one of the contributions of this thesis.

Table 2.1 presents a summary of the advantages and disadvantages of the main techniques typically used to recognize and understand complex human everyday activities. From that table we can observe that using the abstract representations is the best option if we want the robot to learn skills and re-use these new skills in different scenarios with different robots. This is difficult to achieve with other approaches since they learned the characteristics/parameters for a specific task and robot. The underlying problem is when we would like to transfer the learned task to a different robot in a more flexible and reliable manner, which sometimes is not possible due to the different embodiments of the human and the robots.

Another important issue, during the recognition of human activities in real scenarios, is the generalization of the obtained models with respect to the learned task. For example, could the trajectory level techniques such as DMPs or HMMs obtain a model that recognizes motions such as reaching, similar to the presented in the bottom right part of Fig. 2.5? Could we use that learned model to recognize human activities in a different scenario, e.g. in a sandwich making scenario? To the best of our knowledge, there is not a positive answer to these questions yet.

For this reason, in this work we focus on the semantic representations methods to solve the above issues, since the recognition of the activity is achieved using its understanding via the extraction of the semantic representations (meaning) of such kinds of activities. Therefore, the generalization of such models is possible for different environments. Nevertheless, for this kind of techniques there are issues that need to be solved first, such as: what kind of information do we need to observe from the activity in order to be able to correctly recognize it? Could we re-use the learned meaning of the motions to generalize them toward different scenarios?

Action recognition and segmentation are very difficult and challenging problems. As we observed from the above sections, the community of computer vision and machine learning are mainly focused on solving the problem of recognition disregarding the problem of segmentation, which is generally done manually. On the other hand, the robotics community needs to solve both problems in an effective, reliable and fast way to allow robots to make the best
decision. In this work we are addressing the non-trivial problem of segmentation, recognition and transference of the human activities into a robotic system. That means that our system has the advantage of the above studied techniques and successfully overcome their problems to have a good performance of around 85% accuracy on recognizing and understanding human activities in real and new scenarios. These findings are presented in the next chapters.
### Table 2.1. This table summarizes the analyzed related work on human activity recognition.

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
</table>
| Video level         | - The image representation technique addresses the issue of occlusion and generalization to different scenarios using spatio-temporal information.  
                         - This approach is very useful when recognizing the change of human posture. | • Difficult to extract the visual features into something meaningful.  
                         • These techniques are mostly used for the recognition but not for the segmentation of the activities, which is given manually.  
                         • The recognition accuracy decreases when more granular activities would need to be recognized. |
| Trajectory level    | - Learn features of the task in order to control the robot by identifying a general policy.  
                         - DMPs can generalize to new situations at some extent.  
                         - HMMs are generally used to segment and generate observed human motions.  
                         - Obtain informative representation of the observations to acquire skill parameters (*what to learn*). | • DMPs and HMMs do not model the trajectories by their meaning.  
                         • Usually these techniques can not re-use the learned models on new scenarios.  
                         • The obtained models are usually for a specific robot and the transference to a different one is difficult.  
                         • In order to learn the models several teaching trials are mostly required. |
| Semantic level      | - Learn models to capture the meaning of the motions.  
                         - Obtained models with the possibility to be re-used on different scenarios.  
                         - Extract abstract representations of the learned task allows to transfer them to different robots.  
                         - Recognition of the demonstrated activity via its understanding. | • It is hard to identify the *relevant* information of the motions with simple perception systems.  
                         • Highly dependent on the time to define the pre- and post-conditions in a symbolic manner.  
                         • Sometimes these techniques could be very abstract to reproduce the observed motions in a physical, e.g a robot. |
The majority of the recognition systems are designed to fit perfectly the studied task, however most of these systems can not easily scale toward new tasks or to allow different inputs. This chapter presents the overall design and main components of our proposed framework. The main advantage of our system is its multiple levels that enhance its scalability and adaptability to new situations. For instance, our perception module permits the use of different input sources, which bootstraps the learning process.

First, Section 3.1 explains the design of our proposed framework. Then, Section 3.2 presents the conceptual structure, connections and functionality of the main modules of our system. Section 3.3, mentions the technical specifications of the new data sets. Then, Section 3.4 presents the data used to test our framework. Afterward, Section 3.5 explains the experimental robotic scenario. Finally, Section 3.6 summarizes the contributions of this chapter.
3.1. Framework design

During the learning process, several issues have to be addressed, where the most important for this research are: the kind of information that needs to be observed as well as the definition of appropriate source(s) of information, e.g. video demonstrations, virtual reality, haptic demonstrators, gaze information, etc. Therefore, we need to design a framework with components and processes that allows flexibility in the perception input data for inferring and reproducing the observed human behaviors.

The structure of our framework is inspired by Carpenter and Call [2007] and contains three main modules to infer and reproduce the goal of the demonstrator’s activity (see Fig. 3.1):

1. Perceive the relevant aspects of the demonstrations.
2. Infer the goal of the demonstrator.
3. Execute the inferred behavior. If needed, evaluate different mechanisms to achieve the desired goal.

![Classical diagram of a flexible imitation system](image)

**Figure 3.1.** Classical diagram of a flexible imitation system. This figure has been adapted from [Carpenter and Call, 2007].
Then, from Fig. 3.1 we can observe that one of the challenges of inferring the observed task greatly depend on observed input data. Since, the target of our system is to use different input sources, then first we need to detect and eliminate the non-relevant data and focus the attention on the important information. That is a very hard problem due to the fact that this relevant information will change depending on the desired task and on the context. For this reason, first we need to define which information should be observed and which one should be inferred. This means that certain level of abstraction has to be employed in a way that it allows to recognize the desired activities.

Noticeably, not all aspects of a task are observable but may need to be inferred, such as the goal of the demonstrator, which in many systems is given by the researchers (as noted by Nehaniv and Dautenhahn [2007]). We then need to define what kind of motions can be observed and which ones should be better inferred. Examples of observable motions are: move, not move or tool use (see Fig. 3.2.1). From these observations we can also retrieve the position of the object(s) of interest from which we can obtain its properties either ObjectActedOn or ObjectInHand. Therefore, taking into account the above criteria, our system contains principally three modules: 1) low-level activity observation; 2) interpretation of high-level human intentions to infer the human activities; and 3) the execution of the inferred activity by the robot. This is further elaborated in the next section.

3.2. System modules

The main focus of this work is the analysis and understanding of human everyday activities from observations out of real scenarios. However, the definition of such activities is ambiguous and it is not very well defined. For example, the reaching activity, which can be defined as an action that consists in moving one hand from a comfortable position in the workspace to the desired position, in order to grasp or touch something [Nori, 2005]. That definition is ambiguous since there exist different positions of the arm that correspond to the same position of the hand. If a subject is asked to repeat the same reaching movement several times, there will be relatively high variability of the individual joint trajectory and a lower variability of the end point trajectories. A similar analysis can be done to other human activities, which indicates that complex human activities should be inferred instead of observed.

Considering the components and processes of the classical systems shown in Fig. 3.1, we design our framework, which contains three main modules:

1. Extracting the relevant aspects of the task from observation, i.e. observation of human motions and object properties.
2. Interpreting the perceived information to infer the behaviors of the demonstrator using our proposed hierarchical approach.

3. Transferring the inferred activity to the robot, in order to execute the best action to achieve the desired goal.

![Diagram](image)

**Figure 3.2.** The framework is composed mainly by three modules: 1) perception of relevant information, 2) inference of the observed goal and 3) the execution of the goal by the robot. The core of our framework is defined in the second module, where the semantic rules and the enhancement of the system by the knowledge base is defined.

The first module from Fig. 3.2 enables robots to perceive different types of visual information from different sources, such as video recordings, virtual reality environments, etc. This means, that the visual features obtained from the environment (raw video) are analyzed and preprocessed. Three primitive motions are segmented from the videos, i.e. *move, not move* and *tool use*. We will refer to these motions as the low-level human motions. Additionally, the information from the environment is also extracted, for example the objects in the scene and their properties, in our case we considered the following two properties: *ObjectActedOn* and *ObjectInHand*. More details about this module can be found in Chapter 4.
The second module, from Fig. 3.2 represents the core of our system. Which interprets the visual data obtained from the first module and processes that information to automatically infer and understand the human behaviors. It is responsible for identifying and extracting the meaning of human motions by generating semantic rules that define and explain these human motions, i.e. it infers what we consider in this work as high-level human activities, e.g. reach, take, pour, cut, etc.

At this point the system evaluates if the provided input information is enough to infer the observed human activity. Notice that this is the step where most systems fail [Park and Aggarwal, 2004]. For example, if during the training stage we only have objects such as: bread, knife and cucumber, then a classical system would be restricted to only accept those values in order to correctly infer human activities. Nevertheless, with the inclusion of the knowledge base we have enhanced our system. This means during the testing stage if we encounter new objects such as: pancake or spatula, then our system looks for their corresponding class and infers the human activity associated to this class. This is the main reason for the inclusion of the knowledge base into our system. In other words, if the system fails, then it uses its knowledge base to obtain an alternative and equivalent state of the system. This new state of the system contains instances of the desired class (see Chapter 5). This knowledge-base is a very important advantage of our system, since we do not need to recompute the obtained model every time we face a new situation, i.e. the generated rules are preserved even in different scenarios (see Chapter 5).

The above modules are used in order to infer the human activities. This outcome represents the input of the third module from Fig. 3.2. This module uses the inferred activity and enables an artificial system, i.e. a robot to imitate/reproduce an action in order to achieve a similar goal as the one inferred/observed. Specifically, the robot executes the motion primitive that could achieve a similar activity as the one observed. This implies that given an activity, the robot needs to execute a skills plan and command the primitives from a library. These motion primitives are executed within the inner joint control loop of the robot, until the desired activity has been successfully achieved. More details about the implementation of this module are explained in Chapter 6.

In the following sections we explain in more detail the technical aspects that we used in order to construct our obtained framework. We start with the description of the new acquired data sets.
3.3. Recording set-ups: pancake and sandwich

Humans develop learning mechanisms from extracting and incorporating new perceived information from the environment either by observing other humans (3rd. view) or directly by interacting with the environment (1st. view experience). Therefore, it is important to analyze not only the human movements from several external cameras while he/she is doing certain tasks but also it is necessary to extend this analysis using the gaze information that could be considered as one human sense “sight”. Will the gaze input be important to learn and recognize new activities? Which observations improve our process of making informative decisions? when I observe other people or when I observe myself? Would the different views change the way of perceiving the same activity or would this enhance the recognition?

In this work we want to explore more realistic scenarios, such as making pancakes or sandwiches for the reason that in these scenarios several goal-directed movements can be observed, which could be typically perceived in such natural environments but can not be observed in a control environment as it is usually done. Furthermore, we want to explore different input sources, such as the gaze of the person during the execution of the task. This is important since we would like to know whether the information obtained from observing the motion of someone else is enough or will our own observations will also improve the learning of the unknown activity. We believe that such real life constraints will help on testing and validating the robustness of our models.

For testing our system, we recorded two new data sets that contain real human everyday activities as we will explain in the current section. As we mentioned in Section 2.2, the current data sets explore very general motions such as running, walking, swinging, etc. The data obtained from videos, for example in Fig. 2.3, are usually manually segmented into the desired activities and they are not analyzed as a whole. This represents another motivation of our need for new recordings, i.e. to capture the continuity between activities from different sources. We have been looking for data sets that include not only external cameras but a camera that records what a human sees while performing the activities and to the best of our knowledge such data sets do not exist.

In Section 2.2.2, we mentioned the CMU data set [De la Torre et al., 2009] that contains information of 5 external cameras and 1 wearable camera which captures the 3rd. and 1st. view respectively. However, the input of the wearable camera is not very helpful for further analysis, specially for the combination of 3rd. and 1st. view perspective while performing everyday activities. For this reason we decided to record a new data set on everyday human activities. Here we focus on combining the Inside-out vision (obtained via Gaze-based camera) with the
Outside-in view (obtained through external cameras). Using these different views, we aim to provide a perception system that can align the first-person perspective with the third person in context of the overall goal and help in understanding the human behavior.

Such data sets are useful not just for the Robotics and Computer Vision community, but also for the Psychology and Neuroscience community. Hence, jointly with our psychologist partners from Ludwig Maximilian Universität München (LMU -Ludwig Maximilian University of Munich) we designed a new experiment for the data sets. In order to validate our findings on the data sets we control the number of subjects, repetitions, conditions and other details that will later be clarified.

With these new recordings, it is possible to analyze the hand-eye coordination during the execution of the desired task. Furthermore, we could find informative evidence to answer the following questions:

- Do the eyes first look at the object of interest before manipulating it? If yes, for how long?
- How long do the eyes fixate their attention on the goal object during its manipulation?
- Where do the eyes look at next? Do the eyes move to the next to-be-manipulated object or somewhere else?
- Does the gaze give us an indication on the next executed activity?

In order to answer these questions we need to investigate the fixation of the eyes. The process of fixation is an essential part of the eye-movement data analysis and can have a dramatic impact on higher-level analysis. Therefore, the analysis of fixations and saccades require the fixation identification (or simply identification). That is the translation from raw eye-movement data points to fixation locations (and implicitly the saccades between them) on the visual display. Such kind of analysis are possible to explore with our new data sets.

3.3.1. Technical specification of recordings

We recorded two real and challenging tasks: pancake and sandwich making. The experimental setup was similar for both tasks as shown in Fig. 3.3. It consists of three cameras located in different positions and a gaze camera with attached markers.

The recordings have the same configuration. They all contain the following modalities:

- Three video static cameras. The cameras have a frequency of 60 frames per second with Bitrate of 24012 Kbps. The video was captured with Sanyo digital cameras HD2000. The videos are of high resolution with dimensions of 1920 x 1080 pixels.
Recording set-ups: pancake and sandwich

**FIGURE 3.3.** Description of the acquisition of data sets. a) shows the location of the cameras during the pancake making scenario. b) depicts the recording setup for the sandwich making scenario and c) shows the sandwich making activity in a different scenario.

- One mounted camera that was capturing images on top of the head of the subjects with the EyeSeeCam [Schneider et al., 2009]. In other words, this camera has the first view perspective. This camera has a frequency of 25 frames per rate. The dimensions of the video are 720 x 406 pixels.

- One gaze camera that is rotating following the user’s eye motion. This is a very unique video due to it gives the focus of attention of the subject during the execution of the task. This camera has a frequency of 25 HZ. The video’s dimensions are 720 x 406 pixels similar to the mounted camera, but zooming the focus of the user. This camera has more noise since it has a rotation motion additionally to the motion of the head of the user.

The location of the sensors into the kitchen as well as the objects used to prepare the sandwich are depicted in Fig. 3.4. The number of subjects, repetitions and conditions are: a) for the pancake scenario is 1 subject, 9 repetitions and 1 (normal) condition; b) regarding the sandwich scenario we have 8 subjects, 8 repetitions and 2 (normal and fast speed) conditions.

Then two real-world data sets are provided and they are publicly available under the following link [http://web.ics.ei.tum.de/~karinne/DataSet/dataSet.html](http://web.ics.ei.tum.de/~karinne/DataSet/dataSet.html), which synchronizes the information of the external and gaze view.
3.4. Information used from the data sets

The new data sets, i.e., pancake and sandwich making present different levels of complexity, since they involve several combinations of the basic activities under different objects. Furthermore, some tasks are performed in parallel using both hands while executing different activities. In addition to the new recorded data sets, we use a publicly available data set for
the activity of setting the table, which was taken from the TUM Kitchen Data Set.

In the following subsections, we show snapshots of the video data used in this work. We will highlight the information that we used for the different kind of analysis, such as: the number of participants, repetitions and conditions executed for each of the recordings. Please notice that we use three different kitchen scenarios to evaluate the robustness of our system, i.e., we demonstrate that the obtained models are not restricted to one scenario, but they could be used on different kinds of human activities among different scenarios. The data sets presented here are mainly for kitchen scenarios but our models could be extended toward other scenarios.

3.4.1. Pancake making

First, we choose a scenario that contains few objects such as the pancake making. The objects of interest on the scene are: 1) dark blue plate, 2) white spatula, 3) beige pancake mix, 4) pancake\(^1\), 5) black electric stove.

Additionally, from that scenario we can analyze the transition between mainly three tasks: 1) pouring, 2) flipping and 3) sliding-out. Each of those tasks is achieved by combining several basic activities, such as: reach, take, release, etc. In other words, the combination of those activities defines a task.

These recordings contain one subject performing the task nine times. The human motions are captured by three cameras located in different positions, see Fig. 3.3a. These new recordings contain information of three external cameras, one mounted camera and one gaze camera. It is possible to observe that the pancake task involves objects that could be used as tools such as the spatula and other objects that mainly help in the preparation of the pancake, such as the pancake mix. These objects are important to define the tool use motion. Fig. 3.5 shows the output of all the cameras during the preparation of the pancake. Besides the external cameras the head mounted and gaze cameras are also shown.

\(^1\)Important to notice that item 4 first appears as dough and after cooking became pancake.
### Chapter 3

Framework based on semantics for human activity recognition

#### Figure 3.5

Depicts the 5 views of the *pancake making* scenario. Rows 1-5 show examples of cameras output and columns a-d depict the human activities performed by right hand: a) *reach*, b) *pour*, c) *flip* and d) *slide_out*.

#### 3.4.2. Sandwich making

As a second scenario, we recorded a more complex activity which is making a sandwich. These recordings also contain information of three external cameras, one head mounted and one gaze camera. The same set-up as the previous data set. This task contains more objects in the environment which are: 1) bread, 2) bread slice, 3) cucumber, 4) cucumber slice, 5) yellow knife, 6) purple knife, 7) blue knife, 8) pepper, 9) mayonnaise, 10) mayonnaise container, 11)
wrapped cheese, 12) cheese container, 13) trash container, 14) sandwich, 15) placemat/plate for the final sandwich.

The conditions for these recordings differ from the previous one, due to we have more subjects, more repetitions, more conditions and different backgrounds (two kitchens). Those conditions can be summarized as follows:

- 2 background scenarios: 1) different objects in the back, such as drawers, robots, etc. 2) a white wall in the back.

- 8 subjects performed the activity. 2 with the noise background and the rest with a white background.

- 2 constrained conditions were tested in all recordings such as 1) normal speed for the preparation of the sandwiches and 2) fast speed for making the sandwiches. The order of those conditions was switched for every participant randomly.

- 8 repetitions per condition by each participant.

- The subjects were not instructed in any way on how to prepare the sandwiches.

Fig. 3.6 shows the different views of the cameras used in our analysis. It also depicts a subset of the main activities performed by the first subject during the preparation of a sandwich. This task contains several objects as well as different activities to successfully complete the task. It is important to notice that some activities are performed simultaneously using both hands, for example when the left hand is holding the bread while the right hand is cutting it with a knife.

As expected the preparation of the sandwiches were never performed the same way, since each participant has his/her own style. An example of the different styles of preparing a sandwich by the subjects can be observed in Fig. 5.9.
3.4.3. Setting the table

The final experimental set up uses videos from the TUM Kitchen Data Set [Tenorth and Beetz, 2013], which contains observations of four subjects setting a table in different ways (each recorded sequence is between 1-2 min). All subjects perform more or less the same activity, including the same objects and similar locations but using different orders (see Fig. 3.7). The manipulated objects within this task are: 1) cup, 2) plate, 3) napkin, 4) placemat, 5) fork, 6)
spoon and 7) knife.

![Images of setting the table activity]

1) Cam A

2) Cam B

3) Cam C

4) Cam D

**Figure 3.7.** The subject is performing the setting the table activity. This task is captured by 4 cameras mounted on the ceiling. Columns a-c) present examples of activities performed by the right hand: a) take, b) put something somewhere, c) open a drawer. And d) presents an activity performed by the left hand which is open a cupboard, while the right hand is reaching something.

Variations include different ordered actions, a different distribution of tasks between the left and the right hand, and different poses that result from different body sizes. Subjects are performing the actions in a natural way, this implies that no further instructions were provided about how to perform the action — apart from the sometimes unnatural transport of only one object at a time — there are fluent transitions between sub-actions. These actions are performed in parallel using both the left and the right hand.

Some subjects perform the activity like a robot would do, transporting the items one-by-one. Where as other subjects behave more natural and grasp as many objects as they can handle at once. There are also a few sequences where the subject repetitively performed actions.
like picking up and putting down a cup (~5 min each). We could notice certain variations
during the executed tasks, which include actions executed in different order and a different
distribution of tasks between the left and the right hand.

This data set is publicly available for download at http://ias.cs.tum.edu/software/kitchen-
activity-data.

3.4.4. Labeling tool

In order to have meaningful analysis of our new recordings, we need to create ground truth to
compare our automatic findings. For this purpose, we have developed a labeling tool within
this project. This labeling tool is used to manually segment the acquired videos into meaning-
ful classes. This label information contains the following data for each frame:

- Manual annotations of the hand motions such as: move, not move and tool use, which
we called low-level movements. This annotation is done separately for each hand.

- We could label the object information from external cameras that were involved during
the activities. These objects are of two kinds: ObjectActedOn and ObjectInHand, their
definition and usage is explained in Chapter 5.2.

- Information of the ObjectSeen obtained from the gaze camera, which could be more
than one object at the same frame.

- The information of the PlaceSeen which is captured by the gaze camera.

- We can label the information of the high-level human activities, such as: reach, take,
cut, sprinkle, etc.

All this information can be merged into a single database for further analysis as shown in
Fig. 3.8. As mentioned before, this manual annotation of the videos will provide a ground
truth for activity recognition using supervise training algorithms. This labeling tool can label
n number of cameras. This means that this tool has a configuration file where the number
of cameras is specified and the output of the labeling tool will automatically adapt to these
requirements. This tool is not restricted to only identify hand motions, due to we can specify
different labels with the configuration file. This makes the tool very flexible to use in different
scenarios. For example, in industrial applications where the activity could be demonstrated by
another robotic system.
### 3.5. Robotic set-up: execution on the iCub

Another main aspect of this work is not only to contribute on the theoretical part but also on the experimental aspect which most of the time is not straight forward. One problem transferring the acquired models into robotic systems or any artificial system is the fact that there are factors that have to be included into the model. For example, the system has to be fast and reliable enough to make the robot to execute the desired motion. Another factor is the synchronization of the control process to have the desired response. Additionally, there is the constant problem of noise perceived from the environment such as the light or a bad calibration of the robot cameras or actuators. This means that applying a model into a robotic system is not an easy task.

For this reason, the transferring of our acquired models into robots is considered as another
important contribution of this thesis. In order to do this, we use a robotic setup, where we validate the proposed framework successfully. The Robot Control System comprises of the humanoid robot iCub (see Fig. 3.9), which is a 53 degrees of freedom humanoid robot [Metta et al., 2008].

![iCub Kinematics](image)

**Figure 3.9.** Kinematic configuration of the iCub. In this work we focus on the upper part of the robot’s Kinematics. This figure has been adapted from [http://wiki.icub.org/wiki/iCubForwardKinematics](http://wiki.icub.org/wiki/iCubForwardKinematics).

We conducted our experiments on the upper part of the iCub, which consist of 25 DOF organized as follows:

- We focus on the right arm which contains 7 DOF from the shoulder until the wrist.
- The right hand has 9 position controllable DOF.
- The torso contains 3 DOF, for the pitch, roll and yaw motions.
- Finally, we use the head, which has 6 DOF. The 3 first DOF correspond to the motions of the neck and the last 3 DOF controls the eye movements.
The iCub was controlled with a Forward and Inverse Kinematics modules as we will later explain in Chapter 6. The Forward kinematics of the iCub are described with respect to the root reference frame, which is located at the waist level as shown in Fig. 3.9. In order to compute the Forward Kinematics of the right arm, torso and head, the Denavit Hartenberg (D-H) representation was followed [Spong and Vidyasagar, 1989]. The effectiveness of this methodology lies in its simplicity and intuitiveness. The main idea behind the D-H convention is to set the coordinate frames for each link in a specific form. This form will limit the relative motion between consecutive frames to a pair of axes. In other words, motions (rotation and translation) on a specific axis are constrained. In particular, D-H constrains the motion on the y-axis. Therefore, in order to specify the position and orientation between two consecutive frames, only four parameters are needed.

To obtain the roto-translation matrix from the root coordinate frame \( R_o \) to the right arm \( R_a \) of the iCub \( (T_{Ra}) \), two steps are required, see eq. \((3.1)\)

\[
T_{Ra}^R = T_{Ro}^R * T_{0}^n
\]  

\((3.1)\)

Note that this homogenous transformation also includes the torso into the kinematic chain. First we need to obtain the matrix \( T_{Ro}^0 \), which describes the rigid roto-translation from the root reference frame to the beginning of the chain 0 reference frame, which in this case is the first joint of the torso. Then, we compute the Matrix \( T_{0}^n \), which is the composition of \( n \) matrices defined by the D-H convention and it is computed as follows:

\[
T_{0}^n = T_{1}^1 T_{2}^2 ... T_{n}^n
\]  

\((3.2)\)

To obtain the matrix \( T_{Ro}^R \), we use the matrix \( T_{Ro}^0 \), defined in eq. \((3.3)\).

\[
T_{Ro}^R = 
\begin{bmatrix}
0 & -1 & 0 & 0 \\
0 & 0 & -1 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  

\((3.3)\)

Table 3.1 shows the D-H parameters for computing the Forward Kinematics of the robot torso including the right arm. In this case the Kinematic chain starts in joint \( q_1 \) located on the first DOF of the torso. Ending in the last joint of the hand \( q_{10} \), as shown in the right side
of Fig. 3.9. The hand represents the reference end-effector frame located in the palm of the robot. The head motions were controlled using the Inverse Kinematics, therefore we do not show the D-H parameters for those joints.

### TABLE 3.1. D-H table for the robot torso and right arm.

<table>
<thead>
<tr>
<th>Links (i)</th>
<th>$\Theta_i$ (degrees)</th>
<th>$d_i$ (mm)</th>
<th>$a_i$ (mm)</th>
<th>$\alpha_i$ (degrees)</th>
</tr>
</thead>
<tbody>
<tr>
<td>i = 0</td>
<td>0+(-22 -&gt; 84)</td>
<td>0</td>
<td>32</td>
<td>90</td>
</tr>
<tr>
<td>i = 1</td>
<td>-90 + (-39 -&gt; 39)</td>
<td>-5.5</td>
<td>0</td>
<td>90</td>
</tr>
<tr>
<td>i = 2</td>
<td>-105 + (-59 -&gt; 59)</td>
<td>-143.3</td>
<td>-23.3647</td>
<td>90</td>
</tr>
<tr>
<td>i = 3</td>
<td>-90 + (5 -&gt; -95)</td>
<td>-107.74</td>
<td>0</td>
<td>90</td>
</tr>
<tr>
<td>i = 4</td>
<td>-90 + (0 -&gt; 160.8)</td>
<td>0</td>
<td>0</td>
<td>-90</td>
</tr>
<tr>
<td>i = 5</td>
<td>-105 + (-37 -&gt; 100)</td>
<td>-152.28</td>
<td>-15.0</td>
<td>-90</td>
</tr>
<tr>
<td>i = 6</td>
<td>0+(5.5 -&gt; 106)</td>
<td>0</td>
<td>15.0</td>
<td>90</td>
</tr>
<tr>
<td>i = 7</td>
<td>-90 + (-50 -&gt; 50)</td>
<td>-137.3</td>
<td>0</td>
<td>90</td>
</tr>
<tr>
<td>i = 8</td>
<td>90 + (10 -&gt; -65)</td>
<td>0</td>
<td>0</td>
<td>90</td>
</tr>
<tr>
<td>i = 9</td>
<td>(-25 -&gt; 25) + 180</td>
<td>16</td>
<td>62.5</td>
<td>0</td>
</tr>
</tbody>
</table>

More details on the modules and libraries used to control the robot are given in Chapter 6. We have the following experimental set-up, where we performed our experiments:

- The pc104 located inside the iCub head runs a full Debian distribution. This version of Linux has been customized for the specific application and to boot from an USB key.

- One laptop running on GNU/Linux OS, where the main server process has to be initiated. For this reason, the laptop hosts two directories and it is responsible to export them using nfs to the pc104. One of these directories is mounted by the pc104 (/exports/code-pc104), the other is mounted by the other machine on the network and by the laptop itself (/exports/code).

- The data communication between the pc104, the server laptop and any external PC, connected to the same Network, is done using a local network based on the TCP/IP communication protocol.

- A robot control unit that consists on a Workstation running on GNU/Linux OS, where the main modules developed in this work are executed.
The above experimental structure is depicted in Fig. 3.10. More details about the implementation of our system into the iCub can be seen in Chapter 6.

**Figure 3.10.** Experimental communication set-up of the robot.

The principal components implemented and used during our demonstrations can been observed in Fig. 3.11 and they are:

- The reasoning and inference modules, which are working *on-line*, this means that they are implemented within the iCub processes. In other words, the robot is processing the video of the human performing the desired task during execution time.

- The iCub has a table in front of it with the object(s) that is going to manipulate, for example the pancake mix and the electric stove.

- The right camera of the iCub is streaming in *real-time* the video from the iCub’s environment. There is a module that will take this video as input to perform the object recognition using Visual Tracking techniques. This tracking is performed by detecting an AR markers attached to the object.

- After the activity has been inferred, it will trigger a *proper* motion that the iCub needs to execute in order to perform a similar activity as the one inferred. For this a library
of motion primitives has been implemented. See Chapter 6 about the implementation of this library.

Figure 3.11. This figure shows the robotic setup, where the experimental validation of this work was performed.

3.6. Summary and contributions of this chapter

Most of the manipulation skills require the correct recognition of complex human activities. The recognition of these human activities is very important specially under real world conditions, which add more (dynamical) constraints to the problem. This chapter presents in a general manner the main modules of our final framework, which are: 1) visual extraction from observations; 2) understanding human behaviors using hierarchical approaches; and 3) transferring of human behaviors to robots and to different scenarios. Using our defined architecture we can extract meaningful perception, which is a very important feature that our system exhibits. The conceptual description of our framework is depicted in Fig. 3.12.

In this chapter we briefly introduced our proposed levels of abstraction: the low-level, which describes generalized actions such as: move, not move or tool use, obtained from the perception module and the high-level abstraction, which represents the basic human activities, such as: reach, take, release, etc., obtained from the reasoning module (see red triangle of Fig.
Figure 3.12. Conceptual diagram of the modules implemented into our system for the understanding of human activities. First, we perceive the environment information from different sources. Then, we extract the important features to infer the observed activity. Finally, the robot will execute a motion primitive to achieve a similar goal as the one inferred.

3.12). Our technique uses the information from the low-level abstraction, to infer the high-level activities. In other words, this module will determine a higher-level understanding of a demonstrator’s behavior via semantic reasoning.

The design of our system permits a better integration of several sources of input information compared to classical systems as shown in Sec. 4.2. This is due to the fact that our system is based on a hierarchical approach, where different levels of abstraction are defined. This abstraction of the observation captures the essence of the activity, which means that our system is able to indicate which aspect of the demonstrator’s activity should be executed in order to accomplish the inferred activity.

The integration of the perception and inference into a robot represents a very challenging task and it is not trivial, since it requires the implementation of high-level control (decision making modules) and the low-level control (motion control) to generate a functional system.

Beside the research contribution, we also conducted new recordings to capture more complex human behaviors than the ones observed in the currently available benchmark data-sets. Our recordings were carried out for two real and complex tasks, making pancakes and sand-
wiches. Our new data sets have the distinct characteristic that they have the 3rd view perspective synchronize with the 1st view perspective. Additionally, when analyzing human motions usually the methods are tested under very controlled environments and not on realistic scenarios. This provides a plus to our data sets since our scenarios were not controlled in any way and these can be useful for different disciplines, e.g. Psychology, where the gaze analysis is a hot topic.
CHAPTER 4

Perception and extraction of visual features (an evaluation)

Robots should be able to process the environment as fast and accurate as possible and de-
duce the best possible decision. Therefore, in this work we made an evaluation of different
state-of-the-art techniques to automatically segment and/or recognize human activities from
different sources such as: videos, trajectories, or a combination of videos and trajectories.

In Section 4.1, we present different techniques used to extract relevant information from
observations. Section 4.2 explains the state-of-the-art technique to extract spatio-temporal
features from videos. Later, Section 4.3 presents the analysis and results of classifying human
trajectories. In Section 4.4 we propose different levels of abstraction to enhance the studied
techniques. Section 4.5 presents quantitatively the improvement of recognition using the sug-
gested levels of abstraction. Then, Section 4.6 mentions a simple algorithm to recognize and
track objects from videos. Finally, Section 4.7 presents the summary of this chapter.
4.1. Analysis of visual features extraction techniques

When talking about transferring the learned task from humans to robots, we expect that such robotic system is able to automatically extract and interpret the incoming source of information, i.e. videos. This implies that, we need a perception module capable to segment the continuous video stream of human behaviors and environment information into meaningful classes. Therefore our proposed system requires a perception module and we explore different perception techniques to define about how such module can be realized. In other words, the goal of our first module is to perceive and extract the relevant information from observations in human everyday activities. In this work, we evaluate three techniques typically used to process the visual information as shown in Fig. 4.1. The analyzed methods and sources of input are 1) visual feature extraction from 2D videos; 2) whole human body tracker; and 3) the simple and fast color-based tracking technique.

![Evaluated Techniques](image)

**Figure 4.1.** General diagram of the perception techniques evaluated in this work, mainly three: a) Feature extraction using ISA, b) model-based tracker from Bandouch et al. [2008b] and c) color-based extraction used to track the hand and objects in the scene.
The outcome of this first module should be the current state of the system, i.e. the identified features from videos, the segmentation of the trajectories, the possible relations between the objects in the environment, the posture of the humans, etc. We notice that the information retrieved by this module can be unlimited. This derives into the following challenges:

- What is the relevant information observed from the environment that the system (robot) should segment and recognize from human activities in any scenario?

- Can we transfer the learned methodology from one scenario to multiple different scenarios?

- Is there a common information that perception techniques can retrieve with high accuracy?

Additionally, we want to apply the best perception methods to robots. However, robots interact in a dynamic environment, where objects change their shapes or transform into new objects. For example, one piece of bread can become several bread slices, which have different shapes and their perception is different. This introduces another problem which is the difficult task of perceiving the relevant information from the current scene. The relevance of the information explicitly depends on the task. For example, if we are interested in recognizing human motions, we are most likely extracting the position and velocity of the human limbs, instead of recognizing the information of the walls or floor. The information of the walls and floor will be part of the environment input information (e.g. inside a video), but it will not be relevant for the recognition of human motions, therefore we ignore that input and consider it as background noise. On the other hand, if the task is to walk from room A to room B, then the environment input about the walls and floor will be the most important. The community of Computer Vision is trying to solve this difficult problem of extracting the relevant information which greatly depends on the analyzed context [Poppe, 2010].

In order to answer the above challenges, we explore several perception techniques and different levels of abstraction. First, we analyze a state-of-the-art method, called Independent Subspace Analysis (ISA), which recognizes human activities. This technique has the best accuracy tested in the benchmark data sets shown in Chapter 2. Therefore, we use this technique to test our kitchen data sets. Later, we use the information of the human trajectories that was obtained from a model-based tracker developed by Bandouch et al. [2008b]. The assessments of the state-of-the-art techniques to correctly recognize human complex activities led us to the definition of different levels of abstraction to enhance the recognition. Then, with our proposed abstractions, we evaluate the ISA algorithms and we implement a simple
Motion recognition from videos using ISA

The stacked Independent Subspace Analysis (ISA) is an unsupervised learning technique that extracts invariant spatio-temporal features directly from unlabeled video data [Le et al., 2011]. The stacked ISA algorithm is used to extract low-level features from videos, which will be later classified using a Support Vector Machine (SVM) in order to recognize the human activities.

In this section we first briefly explain the basic concepts of the ISA algorithm and later we explain the extension of ISA which uses convolution and stacking to learn hierarchical representations. Finally, subsection 4.2.3 presents the methodology and results obtained to classify motion recognition from videos\(^1\). It is important to highlight that this kind of technique can only be used for the classification of videos but not for the segmentation of the videos into the corresponding classes. This means that the videos are previously divided into classes and that division is mostly done manually.

4.2.1. Stacked Independent Subspace Analysis (ISA)

The stacked ISA algorithm is a deep architecture consisting of several layers of ISA. It is often used to learn features from unlabeled image patches. The best way to describe this technique is as a two-layered network [Hyvärinen et al., 2009], where the first layer contains simple units with square non-linearities and the second layer is composed of pooling units with square-root non-linearities (see Fig. 4.2).

The weights \(W\) in the first layer are learned, and the weights \(V\) of the second layer are fixed to represent the subspace structure of the neurons in the first layer. Each node of the second layer pools over a small neighborhood of adjacent first layer units. This means that given an

\(^1\)The evaluation of this technique was done jointly with the Biointelligence laboratory from the School of Computer Science and Engineering, Seoul National University.
Figure 4.2. This figure shows the architecture of the ISA neural network. We observe that the input data to the first layer is coming from the video and the output of each layer represents the learned features from the video. This figure has been adapted from [Le et al., 2011].

Input pattern \( x' \), the activation of each second layer unit is

\[
p_i(x'; W, V) = \sqrt{\sum_{k=1}^{m} V_{ik} \left( \sum_{j=1}^{n} W_{kj} x'_j \right)^2}
\]  

(4.1)

where \( W \) is learned by finding sparse feature representations over the second layer with

\[
\text{minimize} \sum_{t=1}^{T} \sum_{i=1}^{m} p_i(x'^t; W, V)
\]

subject to \( WW^T = I \)  

(4.2)

where \( \{x'^t\}^T_{t=1} \) are linearly transformed input examples and \( W \in \mathbb{R}^{k \times n} \) represents the weights connecting the input data to the simple units. \( V \in \mathbb{R}^{m \times k} \) defines the connection weights between the simple units and the pooling units. \( n, k \) and \( m \) are the input dimension, the number of simple units and the pooling units respectively. The orthonormal constraint is to ensure that the features are sparse enough.

This algorithm needs to be adapted for the video domain. The inputs to the network are 3D video blocks instead of image patches, i.e. we flatten the sequence of patches into a vector.
This vector becomes the input features to a single ISA. Therefore, to learn high-level concepts it is necessary to stack several ISA networks. Then, a new convolutional neural network architecture is designed which progressively makes use of Principal Components Analysis (PCA) and ISA as sub-units for unsupervised learning (see Fig. 4.3.b).

**Figure 4.3.** Extension of the ISA neuronal network architecture for video data. a) shows the ISA neural network architecture and b) shows the stacked convolutional ISA for video data (Figure adapted from [Le et al., 2011]).

The advantages of using the stacked ISA algorithm are the following:

- It is able to extract low-level features from videos using the stacked Independent Subspace Analysis.
- ISA is an unsupervised learning method.
- It extracts invariant spatio-temporal features directly from unlabeled video data.
- ISA is able to discover unexpected features.
- It is computationally more efficient and robust than other unsupervised methods.
4.2.2. Steps to recognize human motions with ISA

We used a state-of-the-art processing pipeline similar to [Le et al., 2011]. First, we learn spatio-temporal features using information from 3D video blocks as input. Those learned features are then convolved with a larger region of the input data. The outputs of this convolution are inputs to the next layer, which is also implemented by another ISA algorithm with PCA as a prepossessing step to whiten the data and reduce its dimensionality.

Then, norm-thresholding is used to eliminate the features at locations where the activation norm is below the defined threshold (\(\delta\)), i.e. this threshold will filter out features from the non-informative background. In this work, we choose \(\delta = 30\%\). Finally, in our experiments we combine the extracted interesting features from both layers and use them as local features for classification using a \(\chi^2\)-kernel Support Vector Machine (SVM). Fig. 4.4 shows an overview of the methodology described above.

![Diagram showing the steps of ISA: 1) Learn spatial characteristics, 2) Extract interesting features (red boxes), 3) Recognition and classification.](image)

**Figure 4.4.** Principal steps involved in human motion recognition. 1) the spatio-temporal patterns are learned, then 2) the obtained features are filtered and finally 3) the recognition and classification is performed using SVM. The final output of this methodology is one of the three possible human motions, i.e. *move, not move or tool use*.

We implement the above methodology to solve the 9 class problem. This means, we intent to test the direct recognition of the *high-level* motions, e.g. *reach, take, put, release, etc.*

4.2.2.1. Experimental ISA set-up

In order to learn the spatio-temporal features, we use images from random video blocks of size 16 × 16 (spatial²) and 10 (temporal³). Additionally, we set the input dimension and the

---

²Spatial refers to the pixel dimensions of the image patches.
³Temporal represents the frames per second used to define a video.
number of simple units as $k = m = 300$. This implies that the input of our first ISA layer learns 300 features. Then, the inputs to the second layer are defined of size $20 \times 20$ (spatial) and 14 (temporal). The simple units of the stacked ISA are set to $k = 200$ and the pooling units are set to $m = 100$, i.e. the second layer ISA network learns 200 features.

### 4.2.3. Results of 9-class recognition problem using ISA

This section will numerically report the performance of the stacked convolutional ISA for unsupervised learning to extract local spatio-temporal features, followed by the classification of those features in order to improve the human motion recognition.

First, we analyze the 9 class classification problem for the high-level human activities (e.g. reach, take, release, etc.) from the data set of making a sandwich (see Fig. 4.5). This video contains several objects in the scene and 9 activities are performed during the preparation of a sandwich. Additionally, in this particular set-up, we constrained the speed of the preparation of the sandwich. This means that participants perform the first sandwich in normal speed and the second in high speed (simulating they were in a hurry). Therefore, the obtained variance between the trials is high, even when the task is performed by the same subject.

Therefore, we use for training the input of three-view videos (cam A, cam B and cam C) from the first participant during the normal condition and, for the testing stage, we use the same participant but with the high speed condition. The target is to correctly classify 9-classes:

1. reach
2. cut
3. unwrap
4. take
5. idle motion
6. put something somewhere
7. release
8. spread
9. sprinkle
Figure 4.5. This figure depicts the sandwich making scenario made by two subjects. A subset of activities executed with the right hand are shown: 1) and 2) present the activities performed by Subject 1: reach and cut. 3) and 4) show similar activities performed by subject 2.

For testing we use the sandwich making video and we split it into 156 subvideos. It is possible to see from the confusion matrix shown in Table 4.1 that only 40 subvideos were classified correctly. This means that the classification performance is 25.64%, which represents a very low accuracy to classify the desired human activities. This methodology will be later referred as the single-stage approach.

4.3. Recognition of human activities using model-based tracker

Beside video information, we also tested the trajectory data of the human postures. In order to do that, we analyzed the trajectory data obtained from a high-accuracy and versatile human motion tracker system developed by Bandouch et al. [2008b]. This system gives detailed and reliable models of human behaviors in everyday scenarios, such as setting the table.
TABLE 4.1. Confusion matrix results in % for high-level activities from the sandwich making scenario using ISA.

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Cut</th>
<th>Unwrap</th>
<th>Take</th>
<th>Idle</th>
<th>PutSmtSmw</th>
<th>Reach</th>
<th>Release</th>
<th>Spread</th>
<th>Sprinkle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut</td>
<td><strong>83.33</strong></td>
<td>0</td>
<td>16.66</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Unwrap</td>
<td>66.66</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td><strong>33.33</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Take</td>
<td>22.22</td>
<td>44.44</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td><strong>33.33</strong></td>
<td>0</td>
</tr>
<tr>
<td>Idle</td>
<td>4.16</td>
<td>0</td>
<td>8.33</td>
<td>0</td>
<td><strong>20.83</strong></td>
<td>4.16</td>
<td>50</td>
<td>12.5</td>
<td>0</td>
</tr>
<tr>
<td>PutSmtSmw</td>
<td>0</td>
<td>42.22</td>
<td>4.44</td>
<td>0</td>
<td>8.88</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td><strong>4.44</strong></td>
</tr>
<tr>
<td>Reach</td>
<td>0</td>
<td>66.67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td><strong>10</strong></td>
<td>23.34</td>
<td>0</td>
</tr>
<tr>
<td>Release</td>
<td>0</td>
<td>9.09</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td><strong>90.90</strong></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Spread</td>
<td>66.67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td><strong>33.34</strong></td>
<td>0</td>
</tr>
<tr>
<td>Sprinkle</td>
<td>0</td>
<td>33.34</td>
<td>66.67</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

This markerless high-accurate and model-based tracker system uses the industry-leading RAMSIS model. It provides 63 degrees of freedom in the joint space (see Fig. 4.6) and has a highly parameterizable skin model which can be adapted to the specific characteristics of the subject performing the kitchen activities [Bandouch et al., 2008a].

To be able to accurately track in the high-dimensional joint state space of the human model, the tracker uses a mixture of annealed and partitioned particle filtering [Bandouch et al., 2008b]. The human motion data used in this work is part of the TUM Kitchen Data Set [Tenorth et al., 2009]. The database contains data for several subjects which execute everyday kitchen tasks like setting the table without specific instructions (see Fig. 4.7).

Due to the complexity of the high-dimensionality of the human postures, we focus our analysis to the end effector of the humans, i.e. the hands. Additionally, we focus on the analysis of the reaching activities as a starting point. The reaching trajectories available in the TUM Kitchen Data Set show a high variability due to different object types and relative object positions. The observations of the hand trajectory data recorded during the table setting episode can be observed in Fig. 4.8. As one can see, these observations contain the data of the right hand and the left hand. Then, the first step is to differentiate between the observations done by the right hand from the ones done by the left hand.

In order to correctly identify the human motions, first we cluster all available reaching motions into subtypes based on the similarity of their shapes using a clustering technique.

71
Figure 4.6. Full joint space of the RAMSIS human model. Figure adapted from [Bandouch et al., 2008a].

Based on Dynamic Time Warping (DTW) and k-means (see Fig. 4.9).

The DTW [Keogh and Ratanamahatana, 2005] algorithm has earned its popularity by being extremely efficient as a similarity measure of time series which minimizes the effects of shifting and distortion in time by allowing elastic transformation of time series in order to detect similar shapes with different phases.

To align two time series $X = (x_i)_{1 \leq i \leq N}$ and $Y = (y_j)_{1 \leq j \leq M}$, $x_i, y_j \in \mathbb{R}^3$, $N, M \in \mathbb{N}$, the algorithm starts by building the $n \times m$ distance matrix

$$C = (c_{i,j}) \in \mathbb{R}^{N \times M},$$

where every $(i, j)$ element of the matrix $C$ is

$$c_{i,j} := d(x_i, y_j) = \|x_i - y_j\|$$

$c_{i,j}$ is the cumulative Euclidean distance.

Once the local cost matrix $C$ is built, DTW finds the alignment path that runs through the low-cost areas of the cost matrix. This path can be found using dynamic programming by
evaluating the following recurrence for $\gamma(i, j)$:

$$\gamma(i, j) = c_{i,j} + \min \{ \gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1) \}$$  \hspace{1cm} (4.5)

which is the summation between the squared distance of $x_i$ and $y_j$, and the minimum cumulative distance of the three elements surrounding the $(i, j)$ element. Then, to find an optimal path, we have to choose the path that gives minimum cumulative distance at $(n,m)$. The distance is defined as:

$$D_{DTW}(X,Y) = \min_{\forall w \in P} \left\{ \sqrt{\sum_{k=1}^{K} (c_{i,j})_k} \right\}$$  \hspace{1cm} (4.6)

where $w = c_{i,j}$ represents the warping paths and $P$ is a set of all possible $w$. $(c_{i,j})_k$ is the $k^{th}$ element of a warping path and $K$ defines the length of the warping path.

$k$-means is a classical clustering technique [Witten and Frank, 2005]. The clustering procedure follows a simple way to classify a given data set ($N$) through a certain number of clusters (assume $k$ clusters) which are fixed a-priori. The main idea is to define $k$ centroids- one for each cluster. For this, $N$ data samples are divided into random $k$ partitions ($k \leq n$) or clusters, where each cluster has one sample (mean) as its cluster center ($c$). This represents one sample in the cluster center of all data objects within that cluster. The better choice is to place the cluster centers as far away from each other as possible. Afterward, the rest of the objects are assigned to the closest cluster centroid according to the ordinary Euclidean distance metric.
The goal is to minimize the objective function, in this case a sum squared error function:

$$J = \arg \min_S \sum_{j=1}^k \sum_{i=1}^n \|x^{(j)}_i - c_j\|^2,$$  \hspace{1cm} (4.7)$$

where $S$ is the set of observations within one cluster. $\|x^{(j)}_i - c_j\|^2$ is a chosen distance measurement, in this case the Euclidean distance between a data point $x^{(j)}_i$ and the cluster center $c_j$. This is an indicator of the distance of the $n$ data points from the respective cluster centers. Then, from the instances within a cluster, a new centroid is computed which represents a new centroid,

$$c_j = \frac{1}{|S_i|} \sum_{x^{(j)}_i \in S_i} x^{(j)}_i$$  \hspace{1cm} (4.8)$$

The above procedure is repeated until all cluster centers are stable, i.e. the centroids do not move any more. In general, after each iteration, the quality of the clusters and the means themselves improve. The main steps of the $k$-means procedure are shown in Algorithm 4.1.

*Figure 4.8.* Example of the observed trajectories of one person during *setting the table*. Left: we observe the trajectories of the human end-effectors (both hands). Right: the trajectories are manually split in two classes: right and left hand execution.
Algorithm 4.1 k-means clustering algorithm.

**Require:** \( N \) : number of data objects.
\( k \) : number of clusters

1. Choose the first \( k \) points as the initial cluster centers \( c \)
2. **while** \( J > \) threshold **do**
3. \{minimize the objective function (see, eq. (4.7))\}
4. **for** \( i := 1 \) to \( N - 1 \) **step** 1 **do**
5. **for** \( j := 1 \) to \( k - 1 \) **step** 1 **do**
6. Compute eq. (4.7) \{Assign each point to the group with the closest centroid.\}
7. **for** \( j := 1 \) to \( k - 1 \) **step** 1 **do**
8. Recalculate the position of the \( k \) centroids using eq. (4.8)
9. **return** Clustered points into \( k \) classes

### 4.3.1. Clustering human activities

To determine the main clusters from the human activities, we focused on the human motions detected as *reaching*. With this analysis, we prove that even when the same activity is executed, the produced trajectories may have different patterns, which means that they will be recognized as different activities. Therefore, we analyzed only those trajectories that were manually labeled as *reaching* from the TUM Kitchen Data Set focusing on the right hand, i.e. we use the Cartesian information of joint HAR (see Fig. 4.6). The methodology to cluster the human trajectories is as follows (see Fig. 4.9):

1. Obtain the DTW distance \( \gamma(i, j) \) between a template trajectory against all the remaining trajectories. The template trajectory is determined as the first *reaching* trajectory from the data set.

2. Cluster the distances obtained from 1 using the *k-means* algorithm (see Algorithm. 4.1). In this case \( k=7 \) was chosen to be the optimal number of clusters. Each cluster contains the trajectories that have a similar shape.

3. Finally, the corresponding 3D trajectories are mapped according to the clusters they belong to. There is an additional step to compute sub-clusters from the obtained clusters by repeating steps 1 and 2 and changing the number of clusters \( k \).

In other words, using a distance metric on the trajectory space obtained through DTW, the k-means algorithm can be used to cluster the reaching trajectories. By varying the cluster count, different types of reaching motions can be identified.
Define the template signal

Get DTW distance

Cluster distances with k-means

Fig. 4.9. Overview of the methodology used to cluster the trajectories using DTW and k-means. First, we define one template signal. Then, we compute the DTW distance between the template and the rest of the signals. After that, we cluster the obtained distances using the k-means algorithms. Finally, we map the distances to the original trajectories and plot the clustered signals.

4.3.2. Results of human trajectories

The results obtained from the above procedure are shown in Fig. 4.10. The trajectories were obtained by taking the Cartesian positions of the hand of the right arm (see HAR in Fig. 4.6). We found several main clusters which are differentiated by object type and relative to the object positions. It can be observed that these sequences show stereotypical and pre-planned motion patterns [Wolpert and Ghahramani, 2000].

76
Each of the clusters can be divided into sub-clusters and these sub-clusters will give us an indication of different grasping types while taking an object. For example, Fig. 4.11 shows the obtained subclusters from the trajectories of cluster 2, which correspond to the trajectories performed while reaching either a placemat or a napkin. From the new obtained sub-clusters we can observe that there are three mainly grasping types chosen by the human.

It is important to notice that each end point of the trajectories from the clusters indicates the different types of grasping even when the subject is manipulating the same object. For example, cluster 1 in Fig. 4.10 represents the obtained trajectories for getting a cup out of the cupboard. From these trajectories, three different ways of grasping a cup are obtained when subclustering them further. The obtained subclusters are shown in Fig. 4.12, where we can observe the three different types of grasping: one is from the middle of the cup, the second is grasping through the handle and finally to grasp the cup on the top. We notice that there are no trajectories for the left part of the cup, maybe this is the results of an obstacle (a plate)
located in that location. Maybe this explains why the second trajectory is the most frequently used to grasp the cup. Furthermore, it is interesting to mention that the taller persons from the analyzed group decided to grasp the cup on the top. At the right part of Fig. 4.12 we can observe that there are mainly two different spots to grasp the placemat on the bottom right or on the middle right. The third spot represents an exception and this is important to mention specially if we are working on real scenarios where such trajectories should be expected [Ramirez-Amaro and Beetz, 2010].

Another technique that we implemented to cluster the motion trajectories is Expectation Maximitation (EM). The EM clustering algorithm computes the probabilities of the cluster membership based on one or more probability distributions. The goal of the clustering algorithm is to maximize the overall probability or likelihood of the data, given the (final) clusters. The EM algorithm is an efficient iterative procedure that searches the Maximum Likelihood (ML) hypothesis in the presence of missing or hidden data. In order to cluster the trajectories shown in Fig. 4.8 we use the information of the trunk which indicates implicitly the place where the human is standing still while he/she is doing the action of Reaching. The goal of this classification is to cluster the observed trajectories in subtrajectories, such as: Reaching something from the cupboard or Reaching something from the table and so on. The new subtrajectories combine the action and the place where the person is standing.
Figure 4.12. Different places where an object, e.i., a cup, is being grasped. Left: Trajectories of the right hand when reaching a cup, we also observe the main spots were the object was grasped and the bottom of figure shows the trajectories patterns for each spot. Right: Shows all trajectories for reaching the placemat, notice that there are two main spots to grasp this object, nevertheless there are some exceptions like the trajectory from the third spot.

4.4. Proposed levels of abstraction

The Psychology community suggests that humans seem to perceive activities as discrete events that have some orderly relations [Zacks and Tversky, 2001]. When someone is capable to segment in time the beginning and ending of an observation, then that is known as event. Then, the relation between different events is known as event structure perception. Therefore, two main issues arise: What are the basic units of action? What are events? Classical studies suggest that the basic-level is the most privilege in perception. It has been proved that such levels are the earliest concepts learned by children [Rosch, 1978]. There is also evidence that behaviors in natural situations are in fact described in a hierarchical manner [Barker, 1963]. Recently, Zacks et al. [2001] collected evidence suggesting that humans perceive actively ongoing activities in terms of partonomic hierarchies, specially when using goal-directed activities.

Theorists suggest that goal-directed activities are mainly exploited by the cognitive system specially for the understanding of others ([Abelson, 1981; Bower, 1982]) and of ourselves ([Vallacher and Wegner, 1987]). Goal-directed activities exhibits recurrent inter-correlated
patterns of activities that can be captured by frames [Minsky, 1974] or scripts [Schank and Abelson, 1977], or other sources.

Inspired by the above findings we propose our first levels of abstraction in terms of goal-directed hand motions and objects of interest, as shown in Fig. 4.13. However, during the labeling process of the low-level motions, i.e. non-goal directed, goal-directed and tool use, we observe that distinguishing between these primitives is not an easy task. Therefore a lower-level is needed, as we explain next.

**Figure 4.13.** The first levels of abstraction proposed to recognize human complex activities.

In the previous sections we have seen that the recognition of human activities is a very complicated problem using as input either video’s features or trajectories. On one hand, if we only use the visual information to recognize the human activities such as reach, take, put, etc., then the recognition is very low. On the other hand, if we only consider the information from the human postures, i.e. the 3D Cartesian positions, then the same activity, e.g. reaching, has different patterns which mostly depend on the relative position of the objects. As a result, these patterns are identified as different ones, even though they define the same activity. Hence, these analyzed state-of-the-art techniques will not suffice for the correct segmentation and recognition of human everyday activities. Therefore, we propose a hierarchical approach that uses different levels of abstraction of the studied problem.

Human motion segmentation represents a very challenging problem, especially if the acquired information is obtained from videos due to it is hard to determine which parts of the video are significant to the problem we want to solve [Poppe, 2010]. To define the different levels of abstraction, first we need to identify the difference between the motions and activities that a human can do [Turaga et al., 2008]. This distinction will help to represent, recognize and learn human activities from videos, and will especially help to determine the granularity level of those activities/motions. Therefore, in this work, we propose to split the complexity of the
Proposed levels of abstraction

The recognition problem in two parts as shown in Fig. 4.14. The first one will gather (perceive) information from the environment using the techniques presented in this chapter. Whereas, the second part handles the difficult problem of interpreting the perceived information into meaningful classes using our proposed reasoning engine which we further explain in Chapter 5.

**Figure 4.14.** This figure shows the levels of abstraction proposed in this work. The bottom part shows the lower-level of abstraction which is obtained from the observed features. Whereas, the top part depicts the higher-level, which correspond to the inferred complex human activities.

The proposed levels of abstraction permit to distinguish between different motion patterns that humans follow to achieve the same activities under the same or different scenarios. This does not mean that the motion patterns have to be considered as two different activities. We now focus on obtaining the lower-level of abstraction that comes directly from the observations. In this case, first we need to analyze the information from the input sensors to identify which factors make one activity different and unique respect to the others, i.e. extract the relevant information. In other words, we want to answer the following question, what are the factors that allows us to achieve such kind of generalization?

For instance, one typically analyzed signal is the velocity profile to recognize human motions between move and not move [Takano and Nakamura, 2006]. However, when dealing with complex tasks such as: reach, take, put, pour, etc., that information would no longer be
sufficient since, as shown in Fig. 4.15, the velocity signals, produced to achieve the activity *Put something somewhere* during the preparation of a pancake, can have different velocities profiles. This typically happens due to that the obtained signals strongly depend on the current position of the object(s) [Ramirez-Amaro et al., 2013b]. Consequently, the obtained velocity trajectories have different length, amplitude, shape, etc. over time, even though they represent the same activity. For example, moving from $A$ to $B$ implies motions to the left, right, up, down or/and a combination of these motions, which originates different motion patterns similar to the ones shown in the top part of Fig. 4.15.

![Figure 4.15](image-url)

**Figure 4.15.** This figure shows the trajectory analysis of the velocity produced by the activity *Put something somewhere*. The top part of the figure depicts a zoom analysis of the trajectories that represent the same activity.

In a similar manner we could include into the analysis the signals obtained by the distances between the hand and the object(s), the orientation of the objects, the acceleration of the hand, etc. In the case of the distances between the hand and object as depicted in Fig. 4.16, the produced patterns are different for the same human activity as expected, since, similarly to the velocity trajectories, the distance signals also depend on the current position of the objects, which will vary over time. However, in this case two out of four of the obtained signals presented a similar pattern to describe the activity *Put something somewhere*, as depicted in the top part of Fig. 4.16. It means that the classification of the activities will be better using
the distances as input, but regarding the automatic segmentation of the signals, it will not be that easy using only this information.

**Figure 4.16.** This figure shows the trajectory analysis of distances between the hand and the detected object produced by the same activity *Put something somewhere*. The top part shows a zoom analysis of the trajectories that represent the same activity.

Similar analysis can be carried out using the joint angles of the human, orientation of the objects, etc., where a similar output will be obtained. This indicates that analyzing trajectories is not a decisive factor to correctly segment and recognize human behaviors. Some questions arise from this:

- Does the correct human activity depend on having the right input information?
- Or does it depend on having a better way of interpreting the incoming data?

We need to define a general abstraction from the data in order to recognize the intentions of the humans from observable data. Manual annotated data helps to determine the significant feature information from videos which we need to meaningfully segment or/and interpret from the perceived data. These manual annotations are very important to determine the general features that can be found in different videos, i.e., the level of abstraction of the data. Additionally, the manual segmentation is very useful especially to generate the ground truth of the observations.
The important aspect of the perception module is to define what will be considered as relevant information. In other words, what could be the highest level of abstraction in order to recognize the intentions of the humans from observable data. For example, in the case of the hand motions, we are able to segment them into mainly three categories: move, not move or tool use. Notice that these kind of motions can be recognized in different scenarios, but they can not define an activity by themselves. Therefore, we need to add the context information, for example which objects are in the scene and what their properties are. The information of the motion together with the object properties has more meaning than considering each entity as its own. The properties that can be easily recognized from the observations are: ObjectActedOn ($o_a$) and ObjectInHand ($o_h$).

Then, the outcome of the perception module is the recognition of the low-level human motions ($m$) from the observations:

- Move
- Not move
- Tool use

As well as the properties of the perceived object(s):

- objectActedOn ($o_a$)
- objectInHand ($o_h$).

The low-level motions and object properties will determine the current state of the system ($s$), which is defined as the triplet $s = \{m, o_a, o_h\}$. The definition and some examples of the identified motions and object properties can be observed in Table 4.2.

The proper combination of low-level motions and object properties will define a high-level human activity. This indicates that such activities can not be observed, rather they will be inferred. Some examples of the analyzed high-level activities together with their respective definition obtained from the Oxford online dictionary\(^4\) are:

- **Reach:** Stretch out an arm in a specified direction in order to touch or grasp something.
- **Take:** Lay hold of (something) with one’s hands.
- **Put something somewhere:** Move to or place in a particular position.

SECTION 4.4

Proposed levels of abstraction

Table 4.2. General definition of low-level hand motions and object properties.

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
<th>Formula</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move</td>
<td>The hand is moving</td>
<td>$\dot{x} &gt; \varepsilon$ and $\varepsilon$ is a threshold, see Section 4.6.1.1</td>
<td>Moving from position A to position B</td>
</tr>
<tr>
<td>Not move</td>
<td>The hand stops its motion</td>
<td>$\dot{x} \to 0$</td>
<td>Holding a bread</td>
</tr>
<tr>
<td>Tool Use</td>
<td>Complex motion, it uses two objects, one is used as a tool and the second is the object that receives the action</td>
<td>$o_h(t) = \text{knife}$ and $o_a(t) = \text{bread}$</td>
<td>Cutting the bread where the objects are knife and bread</td>
</tr>
<tr>
<td>Object acted on</td>
<td>The hand is moving towards an object</td>
<td>$d(x_h,x_o) = \sqrt{\sum_{i=1}^{n}(x_h-x_o)^2} \to 0$</td>
<td>Reaching for the bread, where $o_a(t) = \text{bread}$</td>
</tr>
<tr>
<td>Object in hand</td>
<td>The object is in the hand, i.e. $o_h$ is currently manipulated</td>
<td>$d(x_h,x_o) \approx 0$</td>
<td>Hold/take the bread, where $o_h(t) = \text{bread}$</td>
</tr>
</tbody>
</table>

- **Release**: Allow (something) to return to its resting position.
- **Idle**: Spend time doing nothing or move aimlessly or lazily:
- **Cut**: Make an opening, or incision in (something) with a sharp-edged tool or object.
- **Pour**, etc.

In the following sections we will demonstrate that the recognition of the low-level motions are very accurate compared to the recognition of the high-level activities. Therefore we propose a method that uses the information from the low-level abstraction to infer the high-level activities. In the remainder of this document we will refer to high-level human activities (such as: reach, take, release, etc) as a set of low-level human motions (i.e. move, not move and tool use).
4.5. Recognition of 3-class problem using ISA

Since the obtained results for a 9-class using ISA (see Section 4.2.3) are not very useful to recognize human activities, we define a new methodology which we call two-stage method and it is shown in this section. In the first stage, we recognize from videos the more abstract human motions as well as object properties. In the second stage, we combine these results into a reasoning engine which finally obtains the desired recognition of human activities (see Section 5.3.4).

In this subsection, we present the results of using the methodology shown in Section 4.2.2, to solve the 9-class problem applied in the proposed 3-class problem. We demonstrate that the recognition is more accurate when trying to recognize the 3-class problem than the 9-class problem. Therefore, we use the same pipeline, the same training and testing video data sets as shown in Section 4.2.2 to make a fare comparison. We focus on solving the 3-class problem into the human motions, i.e. to classify the abstract low-level human motions:

1. move
2. not move
3. tool use

The obtained results using the ISA method for the 3-class problem show a huge increment of the recognition accuracy up to 66.67%, this accuracy is much higher than the accuracy of the single-stage method (~25%). Please refer to Table 4.3 b) to see the obtained confusion matrix. We can observe that tool use motions are misclassified as move motions, this indicates that tool use could be considered as a subclass of move. Therefore we may need the information of the objects to help the system to distinguish between these two different motions. More details about how to include these results into the reasoning engine can be found in Chapter 5.

Afterward, we use the learned features from the first subject and we test them on the second subject (see Fig. 4.5). The classification accuracy for the low-level human motions is around 65%, which is high considering that we used different video sets for training and testing. Different video sets imply that the visual features between the videos are different, for example, the subjects may have shirts of different colors. Such features have not been trained previously. This represents a very important aspect of learning and it is considered as a self-taught learning problem. To the best of our knowledge the state-of-the-art techniques for action recognition is about 51.5% of accuracy for the self-taught learning problems [Le et al., 2011].

Since, we demonstrate that there is a major improvement in the recognition of human motions using a 3-class method, then we test another scenario of the pancake making video data
4.6. Recognition of 3-class problem using Color-Based techniques

To improve human recognition, we decide to combine the video information with the trajectory information in order to extract the relevant features from the human motions. In other words, from the videos we can identify the important features, e.g. color-based features (see Fig. 4.18). The current position of the feature is identified and stored, in order to retrieve a

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Classified as</th>
<th>a) Pancake making</th>
<th>b) Sandwich making</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move</td>
<td>Move</td>
<td>50</td>
<td>61.9</td>
</tr>
<tr>
<td></td>
<td>Not Move</td>
<td>50</td>
<td>9.58</td>
</tr>
<tr>
<td></td>
<td>Tool Use</td>
<td>0</td>
<td>28.57</td>
</tr>
<tr>
<td>Not Move</td>
<td>Move</td>
<td>0</td>
<td>9.09</td>
</tr>
<tr>
<td></td>
<td>Not Move</td>
<td>100</td>
<td>81.81</td>
</tr>
<tr>
<td></td>
<td>Tool Use</td>
<td>0</td>
<td>9.09</td>
</tr>
<tr>
<td>Tool Use</td>
<td>Move</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Not Move</td>
<td>33.33</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Tool Use</td>
<td>66.66</td>
<td>100</td>
</tr>
</tbody>
</table>

5This confusion matrix is computed for each sub-video, i.e. for the pancake testing scenario we have 25 sub-videos of different length.
6The experiments were carried out with a PC-desktop with 8GB RAM and Intel® Core™ i7.
This section will explain the methodology to segment and recognize the human motions using the color-based technique. The perception component from our system depicted in Fig. 4.18 represents the first module of our system. The perception module will enable robots to perceive different types of goal-relevant information by observing humans. To achieve this, the visual features \( f_v \) obtained from the environment (raw video) are analyzed and preprocessed.

### 4.6.1. Pipeline of the Color-Based recognition

In order to identify the low-level motions and object properties from video sequences, we implement a well-known and simple color-based algorithm to detect and track the position of hands and objects. We use the OpenCV (Open Source Computer Vision) library [Bradski and Kaehler, 2008] to obtain the visual features \( f_v \) in order to get the hand position \( x_h \) and to compute its velocity \( \dot{x}_h \). A similar procedure is performed for recognizing the objects and its trajectory for that feature through time.
FIGURE 4.18. These figures show the expected outcome of the perception module using a simple Color-Base method. The visual features extracted from the video are the state of the system: \( s = \{m, o_a, o_h\} \).

properties in the scene.

The methodology is summarized in three algorithms.

1. We convert the color space of the original image from BGR to HSV, since it is more suitable for color based image segmentation. Then, we obtain a binary image using the function `cvInRange()`, which uses the upper and lower boundary array for thresholding the image. The boundaries are obtained off-line and they represent the maximum and minimum limits of the HUE, SATURATION and VALUE of the object to be detected, i.e. the \(\text{HSV}_{\text{min}}^{\text{max}}\) color space, which is obtained heuristically in this work.

As a result, the obtained image contains the recognized area(s) of interest represented as white isolated objects. After that, we smooth the binary image using the function `cvSmooth()` with the method \(\text{CV\_MEDIAN}\). Then, we use the function `cvCanny()` to find the edges of the smoothed image, followed by the function `cvFindContours()` to obtain the area enclosed by the recognized contour, where the position of the identified object \(x_o\) is retrieved. The above process is depicted in Fig. 4.19 and Algorithm 4.2 presents the general steps, which are needed to detect a specific color area in a given image.

2. We notice that in step 17 of Algorithm 4.2 a low-pass filter is used to smooth the signal and remove some of the noise. We use a simple technique called the moving average, where an array of raw (noisy) data \([y_1, y_2, \ldots, y_N]\) is converted to a new array of
smoother data. This algorithm consists of replacing each data point with the average of the neighboring data points defined within a time span (moving window). This process is equivalent to a low-pass filter with a response given by the following equation:

\[
y_s(i) = \frac{1}{2N+1} (y(i+N) + y(i+N-1) + \ldots + y(i-N))
\]

where \(y_s(i)\) is the smoothed value for the \(i\)th data point, \(N\) is the number of neighboring data points on either side of \(y_s(i)\), and \(2N + 1\) is the size of the moving window, which must be an odd number. Afterward, the normalized velocity is computed as follows:

\[
\dot{x} = \frac{dx}{dt} \text{ and } \dot{y} = \frac{dy}{dt}
\]

\[
vel = \sqrt{\dot{x}^2 + \dot{y}^2}
\]  

3. We define a velocity threshold (see Fig. 4.20) to segment between move or not move. This procedure is explained in Algorithm 4.3 which automatically segments and recognizes the hand motions into the two possible classes: move, not move. However, to recognize the tool use motion we need to identify the object properties, i.e. ObjectActedOn or ObjectInHand, which is explained in Algorithm 4.4.

4. Finally, Algorithm 4.4 presents the steps to identify the objects of interest in the scene and object properties such as: ObjectActedOn or ObjectInHand. Such object properties are determined by the variation of two threshold values as explained in Algorithm 4.4. In the next subsection a further analysis on the definition of these thresholds is performed.

It is important to note that the recognized object \(o_i\) can only satisfy one of the above object properties, i.e. \(o_a(t) = o_i\) or \(o_b(t) = o_i\) but not both in the same frame \(t\). Nevertheless, it is possible to have more than one object on the scene, for instance \(o_1 = \text{pancake}\)
Algorithm 4.2 General Color-Based algorithm.

Require: \( \text{img\_src} \): input video.
\( \text{hsv\_min}[i] \): minimum value of the foreground.
\( \text{hsv\_max}[i] \): upper bound of the foreground.

1: while true do
2: \( \text{cvCvtColor}(\text{img\_src}, \text{hsv\_img}, \text{CV\_BGR2HSV}) \) \{Change the color format of the image from BGR to HSV\}
3: for \( i = 1 \) to \( N \) step 1 do
4: \( \text{hsv\_mask}[i] = \text{cvCreateImage}(\text{cvGetSize}(\text{hsv\_img}[i]), 8, 1) \) \{Create a mask HSV image\}
5: \( \text{cvInRangeS}(\text{hsv\_img}[i], \text{hsv\_min}[i], \text{hsv\_max}[i], \text{hsv\_mask}[i]) \) \{Obtain a binary image using the upper and lower limits, to recognize the area(s) of interest\}
6: \( \text{cvSmooth}(\text{hsv\_mask}[i], \text{hsv\_mask}[i], \text{smoothtype}, \text{size1}, \text{size2}, 0, 0) \) \{Smooth the binary image using a preferred smooth type\}
7: \( \text{cvCanny}(\text{hsv\_mask}[i], \text{hsv\_mask}[i], 1, 3, 5) \) \{Find the edges of the areas of interest\}
8: \( \text{cvFindContours}(\text{hsv\_mask}[i], \text{storage}, &\text{contours}[i], \text{sizeof(CvContour)}, ..., \text{cvPoint}(0, 0)) \) \{Compute the contours from the binary images\}
9: while \( \text{contours}[i] \) do
10: if \( (\max_x > \text{contour}[i].x \geq \min_x) \) then
11: \( \text{contour\_new}[i] = \text{contour}[i] \)
12: if \( \text{contour\_new}[i] \) then
13: \( [\text{X}[i], \text{Y}[i]] = \text{getPos(\text{contour\_new}[i])} \) \{Obtain the position of the contour\}
14: \( [\text{smooth\_x}[i], \text{smooth\_y}[i]] = \text{smoothPos}(\text{X}[i], \text{Y}[i]) \) \{Apply a low pass filter to smooth the position data (see Formula 4.9)\}
15: if \( i = 1 \) then
16: \( \text{vel} = \text{getVelocity}(\text{smooth\_x}[i], \text{smooth\_y}[i]) \) \{Obtain the normalized velocity of the hand, i.e. \( i = 1 \)(see Formula 4.11)\}
17: \( \text{smooth\_vel} = \text{smoothVel}(\text{vel}) \) \{Apply a low pass filter\}
18: \( \text{motion} = \text{hand\_recognition}(\text{smooth\_vel}) \) \{See Algorithm 4.3 for the hand recognition process\}
19: \( [o_a, o_h] = \text{object\_recognition}(\text{smooth\_x}[i], \text{smooth\_y}[i]) \) \{See Algorithm 4.4 for the object recognition steps\}
20: return The state of the system \( S = \{\text{motion}, o_a, o_h\} \)
Algorithm 4.3 Hand tracking and segmentation algorithm.

Require:  
\text{smooth\_vel}: \text{velocity of the hand.} 
\text{threshold\_vel}: \text{velocity threshold.} 

1: if (\text{smooth\_vel} > \text{threshold\_vel}) then 
2: \quad motion = \text{MOVE} 
3: else 
4: \quad motion = \text{NOT\_MOVE} 
5: \quad return \ motion 

and \( o_2 = \text{spatula} \) where the object property \( o_a(t) = o_1 \) and \( o_h(t) = o_2 \).

When more than two objects are being recognized on the scene and one of them is true for \( o_a \) and the second one is true for \( o_h \) (as in the previous example), then that motion is classified as \textit{tool use}.

4.6.1.1. Threshold definitions

The above methodology (see Algorithm 4.4) requires the definition of three thresholds:

1. \( \text{threshold\_vel} \): velocity of the hand that determines if the hand is \textit{moving} or not.

2. \( \text{threshold\_distanceMax} \): determines the maximum distance between the hand and the object(s) on the scene. This threshold will determine if the object has the property of \textit{ObjectActedOn}.

3. \( \text{threshold\_distanceMin} \): defines the minimum distance between the hand and the object(s). When the distance between the object and the hand is lower than this threshold then, the object has the property of \textit{ObjectInHand}.

In previous literature, Takano and Nakamura [2006] proposed to segment human motions into short sequences of motions mostly using the information of the velocity of the analyzed limbs. The segmentation of the motions is done by setting the velocity thresholds heuristically, mostly to determine if the limbs were moving or not. In a similar manner, we have proposed a procedure to segment the human motions based on the velocity of the hand(s). The correct definition of these thresholds represents the only input parameters of our system. In order to correctly set those thresholds, we perform the following analysis.

\textbf{Velocity thresholds:}

- First, we plot the velocities produced by both hands during the execution of the same task. For example, we plot the velocity profile of the right and left hand during the task
Algorithm 4.4 Object properties recognition algorithm.

Require: \texttt{smooth\_x[i]}, \texttt{smooth\_y[i]}: position of the hand and objects detected.
\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ threshold\_distanceMax: maximum distance between the hand and object.
\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ threshold\_distanceMin: minimum distance between the hand and object.

1: \texttt{hand\_pos = [smooth\_x[1], smooth\_y[1]]} \{The hand is always the first object detected\}
2: \textbf{for} \ i = 2 \ \textbf{to} \ N \ \textbf{step} \ 1 \ \textbf{do}
3: \texttt{object\_pos = [smooth\_x[i], smooth\_y[i]]}
4: \texttt{distance[i - 1] = getDistance(hand\_pos, object\_pos)} \{obtain the distance between the hand and the identified objects\}
5: \texttt{smooth\_distance[i - 1] = smoothDistance(distance[i - 1])} \{Apply a low pass filter to smooth the data\}
6: \textbf{for} \ j = 1 \ \textbf{to} \ N - 1 \ \textbf{step} \ 1 \ \textbf{do}
7: \{Find the properties of the objects on the scene\}
8: \textbf{if} \ (\texttt{smooth\_distance[j]} < \texttt{threshold\_distanceMax}) \textbf{and} \ (\texttt{decrementa}) \textbf{then}
9: \texttt{oa = j}
10: \textbf{if} \ (\texttt{smooth\_distance[j]} < \texttt{threshold\_distanceMin}) \textbf{then}
11: \texttt{oh = j}
12: \textbf{if} \ (\texttt{smooth\_distance[j + 1]} < \texttt{threshold\_distanceMax}) \textbf{then}
13: \texttt{oa = j + 1}
14: \texttt{motion = TOOL USE} \{Tool use motion is defined if it has both properties \texttt{oa} and \texttt{oh}\}
15: \textbf{else}
16: \texttt{oa = NONE}
17: \textbf{else}
18: \texttt{oh = NONE}
19: \textbf{else}
20: \texttt{oa = NONE}
21: \texttt{oh = NONE}
22: \textbf{return} \texttt{motion, oa, oh}
of preparing a pancake. This plot is depicted in Fig. 4.20 (Top picture). Several things can be observed from this picture. The first one is that both velocity signals have similar patterns. Which indicates that the velocity produced by each arm is similar even though they perform different activities. Note that the frame rate of this video is 24Hz and the velocity was computed from the current positions of the pixels over the frames.

• Then, we heuristically define a threshold which best fits both signals. This threshold is shown by the cyan line in Fig. 4.20. We can observe that for the pancake task, the threshold is 3.0. However, this threshold will not correctly segment the human motions for the scenario of making a sandwich. This is due to the objects on the sandwich scenario are closer to each other and as a result the size of the image (pixels) is smaller. Consequently, we observe a lower velocity in the middle and bottom plots in Fig. 4.20. In this case a velocity threshold of 1.5 is chosen. It is important to remark that this threshold is the same for all the videos of the sandwich making task. In other words, even when different participants are considered performing the task under different conditions, the same threshold will remain the same, see Fig. 4.20. We also witness that the same activity performed by the same subject under two different conditions (normal and fast) result in two different velocity profiles. Which exemplifies the difficult task of automatically segmenting the human activities accurately.

Distance thresholds:
A similar procedure is performed for the definition of the distance thresholds. For example, we plot the distance between the pancake mix and the left hand (blue line in Fig. 4.21, Top) and the distance between the pancake mix and the right hand (red line in Fig. 4.21, Top). These distances are plotted together in the same figure and two thresholds are defined heuristically.

First, we define the maximum distance to determine the object property of ObjectActedOn (see the cyan line from Fig. 4.21). In this case the chosen threshold is 140 for the pancake task and 145 for the sandwich task. This means that the section of the signals that goes below this threshold will be segmented as ObjectActedOn (see the cyan box from Fig. 4.21). Then, the minimum distance is determined to identify the object property of ObjectInHand (see the magenta line from Fig. 4.21). Please note that the signal below this threshold will lose its property of ObjectActedOn and get the new property of ObjectInHand (see the magenta box from Fig. 4.21). The used thresholds for the minimum distance are 85 for the pancake task and 70 for the sandwich task. The determination of the velocity and distance thresholds can be automated using the Image-Based Learning Approach (IBLA) [Ramírez-Amaro, 2007; Ramirez-Amaro and Chimal-Eguia, 2012]. This is considered as future work.
Recognition of 3-class problem using Color-Based techniques

4.6.2. Results of hand motions and object properties

We tested our algorithm in two data sets: pancake making and sandwich making, as explained in Section 3.4. The experiments were performed only on a subset of the whole videos. For the pancake scenario, we segment the video until the pouring action was finished. While, for the sandwich scenario, we segment the video until the cutting the bread action has ended. After the segmentation of the video, we execute the algorithm for two conditions: normal and fast speed.

Quantitatively the results indicate that the human motions for both hands (move, not move, tool use) are correctly classified in the pouring the pancake mix scenario with an accuracy of 91% with respect to the ground-truth. Fig. 4.22 shows an example of the segmentation of the

---

7The ground-truth data is obtained by manually segmenting the videos into hand motions, object properties and human activities.
signals obtained from tracking both hands. This figure also depicts that the signals presented by each of the hands have different patterns which makes the segmentation and recognition a difficult problem, however our technique can overcome such problems. Fig. 4.22.a) shows the raw data obtained from the color-based perception module and Fig. 4.22.b) presents the automatic segmentation and recognition of that signal into three classes move (blue line), not move (green line) and tool use (red line).

Then, we tested our system for the new task of flipping the dough and the accuracy of recognition of the hand motions is 86.92%. The accuracy of segment the object properties is in average 90.65%. The results show a lower recognition accuracy specially for the object property ObjectActedOn due to the proximity in the observed 2D image at all times between the hand and the spatula.

Moreover, we also tested our method with the sandwich making scenario. The obtained
FIGURE 4.22. Plots of the original and clustered hand trajectories for the \textit{pancake making} scenario. The clustered trajectories are obtained using the color-based technique to segment the hand motions in three classes.

results of the automatic segmentation and recognition are shown in Fig. 4.23. Similarly, the left part of the figure shows the original 2D Cartesian trajectories obtained from the sandwich task and the right side shows the segmented and recognized trajectories. The accuracy of segmentation and recognition for this task is around 86.24%.

Regarding the recognition of the object properties (\textit{ObjectActedOn} and \textit{ObjectInHand}), the accuracy for the pancake making is around 96.22% and for the sandwich scenario it is 89.24% since the objects on the scene are smaller and get occluded by larger objects. We can notice that even though we are using a very simple algorithm to identify and track objects from videos, the obtained accuracy is high. Furthermore, the \textit{color-based} method can be applied for \textit{on-line} object recognition, as implemented in this work. Which means that the above results are obtained by the segmentation of \textit{on-line} videos. The obtained confusion matrix\footnote{This confusion matrix is obtained frame-wised.} is
Figure 4.23. Plots of the original and clustered hand trajectories for the sandwich making scenario. The clustered trajectories are obtained using the color-based technique to segment the hand motions in three classes.

shown in Table 4.4. Nevertheless, one of the limitations of the Color-Based method is that the object(s) and the background should have a significant color difference in order to successfully segment them and each object needs to have different colors.

Table 4.4. Confusion matrix results of low-level motions in % for pancake and sandwich making using the Color-Based technique.

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Right Hand</th>
<th></th>
<th></th>
<th>Left Hand</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a)Pancake</td>
<td>b)Sandwich</td>
<td>a)</td>
<td>b)Sandwich</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Not Move</td>
<td>80.5</td>
<td>19.46</td>
<td>80.46</td>
<td>10.59</td>
<td>8.94</td>
<td>96.42</td>
</tr>
<tr>
<td>Move</td>
<td>0</td>
<td>100</td>
<td>37.76</td>
<td>52.44</td>
<td>9.79</td>
<td>0</td>
</tr>
<tr>
<td>Tool Use</td>
<td>1.02</td>
<td>0</td>
<td>18.29</td>
<td>1.01</td>
<td>80.68</td>
<td>6.06</td>
</tr>
</tbody>
</table>

At this point our perception system is able to accurately extract the relevant information from the observations. This information defines the hand motions and two object properties. An example of the outcome of this perception module is depicted in Fig. 4.24. However, the segmented information is not enough for recognizing human activities such as reaching, taking, cutting, etc. For this reason an inference module is implemented and it is further
Figure 4.24. Results of the perception module in order to automatically segment the low-level human motions and the object properties. In a) the hand motion is segmented as *not move* and the object recognition does not have any relevant information. In b) the hand is *moving* and the object has the property $o_a(t) = \text{bread}$. c) and e) have the same segmentation values for the hand (*not move*) and the object has the property $o_h(t) = \text{bread}$. Finally d) has motion=move and the object property $o_h(t) = \text{bread}$.

The implemented color-based segmentation module is a basic visual process and it is possible to replace it with more sophisticated object recognition algorithms. Further discussions on perception algorithms are out of the scope of this thesis.

### 4.7. Summary and contributions of this chapter

Regarding the problem of action recognition, the robot should be able to correctly identify through observations the behaviors of the demonstrator including the information of the object(s) involved during the execution of the observed behavior. This represents a very important and hard problem since the process of visual input is expected to be as fast and accurate as possible to automatically extract and interpret the information from the incoming sources. For this reason, in this chapter we assessed different state-of-the-art techniques as well as the simplest and fastest methods proposed to solve the recognition problem for robots.

First, we have analyzed the human activity recognition problem from videos using an unsupervised state-of-the art learning algorithm based on Independent Subspace Analysis (ISA) [Le et al., 2011] to extract spatio-temporal features from videos. Even though this a very powerful technique, it does not perform as accurately as expected. Even if we use a similar methodology as proposed by the state-of-the art technique [Le et al., 2011] to recognize the human activities, the results have very low accuracy around 25%. This is mostly since the activities that we are trying to recognize are very granular and this represents a more difficult problem than the classical data sets used.

Therefore, what we propose in this Chapter is that instead of recognizing the desired human activities directly from the videos, we only recognize more general motions such as *move*,
not move and tool use, where the recognition accuracy increases to above 72%. The complex part of the identification of the more granular and specific human activities is then handled by a reasoning engine. However, the analyzed ISA algorithm uses off-line classification and the videos were manually split into meaningful classes. Additionally, the information of the object properties is obtained by manual labeling. The assessment of the ISA algorithm is the result of a joint collaboration between the Technical University of Munich (TUM) and Seoul National University (SNU), as part of the GenKo project and the obtained results have been published at an international conference [Ramirez-Amaro et al., 2013b].

Afterward, we tackle the recognition problem using the trajectories produced by the right hand to cluster a specific activity into different types of reaching motions. Section 4.3 presented the results of clustering the human trajectories into more specific classes. From these results we observe that the human motion is predetermined by the position of the object. We notice that even when all the analyzed trajectories belong to the same motion, in this case reaching, the obtained clusters show the single reaching motion as seven different activities. Even though this analysis is important, it is not useful when we want to generalize toward different scenarios, where the objects may be in different places as the trained ones. The trajectories used in this analysis were obtained from the TUM Kitchen Data Set\(^9\), which is publicly available. This trajectory approach indicates that the trajectories will be clustered based on the similarity of the shapes of the trajectories. We can also observe that the obtained results are very difficult to generalize toward different scenarios due to the fact that the obtained models are relative to the object position and object type, even when the object information is not considered in the model, i.e. the object information is implicit within the human trajectory.

Finally, in Section 4.6 we presented the results obtained from the color-based technique using our proposed methodology to automatically segment and recognize human motions and object properties. The obtained results suggest that we can recognize general motions and object properties with high accuracy above 85% depending on the complexity of the scenario.

The contributions of this chapter are mainly the evaluations of different ways of getting the perceived information of the environment. Three different sources were assessed, such as: video information, trajectory data and the combination of the video and trajectory data. Each of these methods have advantages and disadvantages and mainly we evaluate the robustness of these algorithms proposed in the literature tested with our new data sets. Additionally, we focus on determining if the obtained model(s) are generalizable toward different scenarios, which is a very challenging task. The findings presented in this chapter are very important

\(^9\)http://ias.cs.tum.edu/software/kitchen-activity-data
when we want to implement a complete system into a robot, where the response time is a key factor. Such constraints are also very important factors when evaluating the perception modules. In the next Chapter 5 we present our semantic representation method that greatly enhance the perception system using the levels of abstraction proposed in this chapter.
CHAPTER 5

Inferring Semantic Human Activities Reasoner (ISHAR)

Robots obtain information from external sensors and from their own sensors. However, it is very difficult to understand and interpret the observations on the basis of raw information using the current techniques (see Chapter 4). Then, we propose a method that equip robots with reasoning capabilities and this enables them to better integrate and interpret the observed information from different sources, e.g. cameras, lasers, actuators, virtual environments, etc.

This chapter represents the core and most important part of this thesis. Section 5.1 presents the proposed semantic reasoning framework. Section 5.2 explains our method to extract the meaning of observations via semantic rules. Then, section 5.3 presents the obtained results of recognizing human behaviors under different constraints. Section 5.4 introduces the knowledge and reasoning engine to enhance our system. Then, section 5.5 presents the obtained results for the knowledge and reasoning engine. Finally, section 5.6 concludes this chapter.
5.1. Inferring human intentions using ISHAR

The major challenge is to find mechanisms that explain the observed raw signals such as poses, velocities and distances, in a way that robots are able to make informative models that permit the understanding of what they are observing and infer how they could generate/produce a similar behavior and more importantly why did they infer that behavior?, in order to re-use the learned models into new situations. One way to abstract the raw sensor data is using hierarchical approaches based on semantic representations.

Semantics is the study of meaning and several methods have been used to determine the semantics in the domain of human behaviors such as: Linguistic Descriptions [Park and Aggarwal, 2004; Inamura and Shibata, 2008], Syntactic Approaches [Turaga et al., 2008; Aggarwal and Ryoo, 2011], Graphical Models [Wörgötter et al., 2009; Koppula and Saxena, 2013a], etc. Typically, the semantic representations in any of the previous methods are given a priori by an expert [Aggarwal and Ryoo, 2011; Tenorth and Beetz, 2013]. Therefore, in this Chapter we explain our propose method to automatically detect the semantics of human behaviors to interpret the interactions/relations between the low-level motions and the objects (see Fig. 5.1).

![Diagram](image)

**Figure 5.1.** This figure illustrates the inference module using the proposed two levels of abstraction. They range from general (low-level) to specific (high-level) human movements.

In this work, the semantics of human behavior refers to find meaningful relationships between human motions and object properties in order to understand the activity performed by the human, i.e. the semantic representations are used to interpret a visual input to understand
human activities. Then, to define the *semantics of human behavior*, we propose two levels of abstraction (see Section 4.4):

1. The *low-level*, which describes the motion primitives (obtained from the perception module, cf. Chapter 4, Section 4.4: module 1) such as: *move, not_move* or *tool_use*.

2. The *high-level* abstraction (obtained from this reasoning module), which represents the *human behaviors*, such as: *idle, reach, take, cut, pour, put* something somewhere, *release*, etc.

Additionally to identifying the human motions, it is needed the recognition of the manipulated object(s) during the observed activity. Then, in order to infer the human intentions, first we need to define mechanisms to interpret the input information. Examples of the different kinds of input information are presented in Chapter 4. The output of this interpretation is mainly the information of the hand motion segmentation (*m*) and the identified properties of the object(s) (*ObjectActedOn(o_a)* or *ObjectInHand(o_h)*). This triplet conforms what we call the state of the system (*s*). Then, using only this information we should be able to extract the meaning of the human motions. In other words, we should be able to recognize the high-level human activities such as: *reach, take, cut, etc.* In this work we propose to achieve that by extracting the semantics out of the observations. The major advantage of this method is that we can transfer the learned *meaning* of the activities into new scenarios.

Once the activity and object(s) are being recognized (see Chapter 4), the next problem is to interpret and understand why this activity is being performed. This *understanding* capability represents a big advantage compare to classical recognition systems, since we allow the robot not only to recognize the human activity but also to identify the human intentions. Additionally, this capability enables the robot to realize if the recognized activity represents a new activity or a previously learned activity. Such understanding of the current task enables the robot to generalize the learned skills toward new situations using the meaning of the activity. Therefore, we propose a hierarchical approach that uses the information from the general *motions* and object properties to infer the *human behaviors*.

Fig. 5.2 depicts the main components involved in this module, which are mainly three sub-modules: 1) Action Recognition, 2) Semantic Rules and 3) Action Understanding. The input for the sub-module 1 (see Chapter 4) is the information of the environment, which is processed and transferred to the sub-module 2. This second block automatically generates the human activity semantic rules that are later used in the third sub-module to grow the knowledge representation with the obtained reasoning rules. These reasoning rules are used to finally infer the human intentions.

106
Then, as shown in Fig. 5.2 we include knowledge-base to enhance the inference capability of our system. This is an important feature due to the fact that, if the inference fails (notice that this is the step where most systems fail Park and Aggarwal [2004].), then the system uses its knowledge base to obtain an alternative and equivalent state of the system which contains instances of the desired class. This represents a very important advantage of our system, since with the knowledge base we do not need to recompute the semantic rules every time we face a new situation, i.e. the generated rules are preserved even in different scenarios. This last step represents the novelty of our system compared to classical approaches.

In the next sections, we present the results of our approach to successfully recognize human actions from videos using semantic representations. Furthermore, we show quantitatively the enhancement of the activity recognition using our reasoning engine. The key factor in
our framework is the abstraction of the problem in two stages. First, by recognizing general motions such as: move, not move or tool use. Second, by reasoning about more specific activities (reach, take, etc.) given the current context, i.e. using the identified motions and the objects of interest as input information.

5.2. ISHAR algorithm description

A decision tree classifier is used to learn the mapping between the human motions and the human behaviors through object information. Prior literature has also proposed the use of a decision tree to define domain-specific rules to determine a meaningful semantic representation using spatial and temporal constrains [Park and Aggarwal, 2004]. However, that approach only recognizes interactions composed of two sequential activities, where the individual leaf node represents an interaction type, which can be reached with different combinations of \(\langle\text{agent} - \text{motion} - \text{target}\rangle\) triplets. That means that the semantics of a human behavior is not unique. Therefore, there is no guarantee that the learned rule can be used in a different situation. This point is very important when defining the semantics, i.e. the possibility of re-use the extracted meaning toward different scenarios. Hence, our system focuses on solving that problem using a decision tree classifier but with a different definition of the triples that makes the semantics of human behavior generalizable to new scenarios.

The methodology that we propose in this work consists of two steps:

1. The first step generates a tree that can determine the human basic activities in a general form, i.e. reach, take, put, release and idle, explain in Section 5.2.1.

2. Then, the second step extends the obtained tree to recognize more complex activities. We call this kind of activities granular activities, for instance cut, pour, spread, flip, etc. The major difference between this kind of behaviors is the context as it is explained in the subsection 5.2.2.

5.2.1. Basic human activities method

To learn the decision tree we require a set of training samples \(D\), which is composed of a set of instances \((S)\). The set of instances is a set of items over which the concept is defined. Each instance \((s \in S)\) describes a specific state of the system and it is represented by its attributes \((A)\). The concept or function to be learned is called the target concept which is denoted by \(c\). In general, \(c\) can be any \(n\)-valued function defined over the instances \(S\); that is
ISHAR algorithm description

$c : S \rightarrow \{0,1,\ldots,n-1\}$. In our case the target concept corresponds to the value of the attribute ActivityRecognition. When learning the target concept ($c$), the learner is presented with a set of training examples ($D$), each consisting of one instance $s$ and its target concept value $c(s)$. We refer to the ordered pair $\langle s,c(s) \rangle$ as the state-value pair which describes the training sample ($D$). In this work the training samples $D$ are described by the following attributes:

1. Hand_motion (with the possible values: move, not_move, and tool_use)
2. ObjectActedOn (with the possible values: Something, None)
3. ObjectInHand (with the possible values: Something, None)

and the target concept value:

- Class $c : ActivityRecognition : S \rightarrow \{\text{Reach, Take, Release, Put\_Something\_Somewhere, Idle, GranularActivity}\}$

Therefore, some examples of the state-value pair ($\langle s,c(s) \rangle$) are:

\[
\langle \{\text{not\_move }, \text{None }, \text{None } \} , \text{IdleMotion} \rangle
\]
\[
\langle \{\text{move }, \text{Something }, \text{None } \} , \text{Reach} \rangle
\]
\[
\langle \{\text{not\_move }, \text{None } , \text{Something } \} , \text{Take} \rangle
\]

The first three elements correspond to the current state $s$, whereas the final item represents the target concept value $c(s)$. Fig. 5.2.1 illustrates the training sample set.

As the Occam’s razor states: “It is preferable to choose the simplest hypothesis that fits the data” [Mitchell, 1997]. Therefore, in order to learn the target function $c$ from a set of training samples $S$, we use the C4.5 algorithm [Quinlan, 1993] to compute a decision tree, where shortest trees are preferred over longer trees and it is guarantee that the attributes closer to the root contain the highest information gain. This represents a good indicator of the generalization of the data. Additionally, decision trees represent a very reliable technique to learn top-down inductive inference rules due to its robustness to noisy data [Quinlan, 1993]. Also they can be represented as sets of if-then rules to improve human readability. Thus, making them transparent and intuitive for further analysis. The central core in the C4.5 algorithm is to select the most useful attribute to classify as many samples as possible by using the information gain measure:

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \cdot Entropy(S_v)$$ (5.1)
Inferring Semantic Human Activities Reasoner (ISHAR)

Figure 5.3. This figure depicts some of the state-value pairs used to generate the decision tree, on the top: three input attributes and its corresponding values per frame. Bottom: representation of the target concept values. Note that each column describes the duration of each activity in frames. For example the activity Reach takes 15 frames (from frame 10 to 25), whereas the activity Take is executed in 3 frames.

where Values(A) is the set of all possible values of the attribute A, and $S_v = s \in S | A(s) = v$ as a collection of samples for S, and the entropy is defined as:

$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2 p_i \quad (5.2)$$

where $p_i$ is the probability of S to belong to class $i$.

The methodology to infer human activities is presented in Algorithm 5.1 and an example of the process is depicted in Fig. 5.4. Notice that the steps 1-3 from Algorithm 5.1 are performed only once, since the obtained tree $T$ captures the rules that can later be used to infer human activities even in new scenarios. This brings to the second important point of the algorithm where steps 4-7 are performed whenever a new data set is being considered. This means, we do not learn a new tree, but instead we use the obtained tree $T$ to infer the target concept $c$ for the new states of the system.

An example of the above process is depicted in Fig. 5.4, where we can observe that the input information of this module is the current state of the system ($s$). Afterward, this module processes that information to generate as output the inferred human activities, the goal ($g$).
Algorithm 5.1 Inferring human activities

Require: Data set (D) {Notice that the Data set could be the training set or a new set}
1: if Training phase then
2: Compute_Tree(⟨s, c(s)⟩, A) {s represents the training instances, c(s) is the attribute whose value has to be predicted by the tree and A is a list of other attributes that may be tested by the learned decision tree. This function returns a decision tree that correctly classifies the given instances using eq. (5.1) and eq. (5.2).}
3: return T, the decision tree that best classifies s and determines the hypothesis h ∈ H such that h(s) = c(s) for all s ∈ S. {The set of hypotheses H is referred as the semantic rules}
4: else
5: for each current state s do
6: Determine: the target class c(s) from the set of hypotheses H obtained from step 3
7: return c(s), the inferred human activity

5.2.2. Granular human activities approach

In order to correctly infer complex activities such as: cut, pour, spread, flip, etc., more attributes have to be considered. For instance, we can take into account the type of object being manipulated, for example for the cut and spread activities, they both use the knife as a tool but they represent different activities. One distinction between these two activities is the object they are acted on (o_a), either the bread or the mayonnaise, respectively. Therefore, a second stage is needed in order to extend our obtained tree T and be able to infer those granular activities.

Then, for the second step, we use as input the activities clustered as Granular from the previous step and we learn a new tree, which represents the extension of our previous tree. The methodology that we follow is similar to the one explained in subsection 5.2.1. It means that the set of instances S is described by the same attributes A, but with different values. For example, the Hand_motion attribute has now only two possible values: move or not_move. The attribute ObjectActedOn could present the new possible values: pancake, dough, bread, cheese, electric stove, etc. Whereas the ObjectInHand attribute could have, for example, the following 4 possible values: bottle, spatula, knife and plastic wrap. Note, that the values of the last attribute are the parental classes of the objects. More details about the inclusion of the class definitions and the knowledge-base into our system are given in Section 5.4.
Some examples of the new state-value pairs \((s, c(s))\) are:

\[
\begin{align*}
\{ & \text{move, pancake, spatula} \}, \text{Slide\_out} \\
\{ & \text{move, bread, knife} \}, \text{Cut} \\
\{ & \text{not\_move, cheese, bottle} \}, \text{Sprinkle}
\end{align*}
\]

The two-steps method is depicted in Fig. 5.4. We observe that the two-steps method to generate the decision tree \(T\) is performed during the training stage.

**Figure 5.4.** Methodology to infer human activities. First, the input video is segmented into hand motions and two object properties (see Chapter 4). Afterward, if we are in a training phase then we compute the rules, if not, we used the obtained rules to infer human activities.

### 5.3. Results of semantic reasoning

The Weka data mining software is used to generate the decision tree [Hall et al., 2009] and the manual annotated data from the sandwich-making scenario is chosen as the training data set (see Section 3.4). This scenario is selected since, from the task examples presented in section 3.4, it represents the task with the highest complexity due to the several sub-activities that it contains. During the training stage, we divide the procedure in two steps. The first step generates a tree that can determine the human basic activities in a general manner. The second step extends the tree to recognize more specific activities, which we call granular activities.
For the first step, we use the information of the ground-truth data of the first subject during the normal condition while he is making a sandwich. We split the data as follows: 60% of the trails (instances of the data set $S$) was used for training and the remaining 40% for testing. Then, we obtained the tree $T_{\text{sandwich}}$ shown in Fig. 5.5. This learning process captures the general information between the objects, motions and activities. From $T_{\text{sandwich}}$ it is possible to observe that the following human basic activities can be inferred: idle, take, release, reach, put something somewhere and granular.

![Diagram of tree](sandbox_tree.png)

Figure 5.5. This figure shows the tree obtained from the sandwich making scenario ($T_{\text{sandwich}}$). Note that in this tree, the name of the objects is replaced by a general label (something), to infer basic activities.

It is important to notice that the first attribute that has to be correctly segmented is hand motion. If the hand attribute is not move, then we could predict that the activity is either taking or idle, which is defined by the object property $\text{ObjectInHand}$. This indicates that from the obtained tree we can determine six hypotheses ($H_{\text{sandwich}}$) which represent the semantic rules describing the basic human activities. Then, the obtained rules define the relationships between the image observations and representations, which are given by each branch of the obtained tree. The taxonomy of the tree represents the syntax of the representations. Following the syntax, we are able to construct machine/human-understandable descriptions of human activities, i.e. semantics. Some examples of such rules are:
if Hand(\textit{move}) & ObjectInHand(\textit{None}) & ObjectActedOn(\textit{Something}) \\
\quad \rightarrow \text{Activity(Reach)} \quad (5.3) \\
if Hand(\textit{not\_move}) & \text{ObjectInHand(\textit{Something})} \\
\quad \rightarrow \text{Activity(\textit{Take})} \quad (5.4) \\
if Hand(\textit{move}) & \text{ObjectInHand(\textit{Something})} \\
\quad \rightarrow \text{Activity(\textit{PutSomethingSomewhere})} \quad (5.5) \\
if \text{Hand(NotMove)} & \text{ObjectInHand(\textit{None})} \\
\quad \rightarrow \text{Activity(\textit{Idle})} \quad (5.6) \\
if \text{Hand(Move)} & \text{ObjectInHand(\textit{None})} & \text{ObjectActedOn(\textit{None})} \\
\quad \rightarrow \text{Activity(\textit{Release})} \quad (5.7) \\

Activities such as: cutting, sprinkling, spreading, etc. are expected from the sandwich making data set. However, during the first-step method, those activities were always clustered under the same class \textit{spread}¹:

\text{if Hand(Tool\_use)} \rightarrow \text{Activity(\textit{Spread})} \quad (5.8)

It means that the C4.5 algorithm is not capable of correctly separate the complex/granular activities into different rules. Thus, the second-step proposed in sub-section 5.2.2 is needed to correctly classify these granular activities.

From this step, we observe that we do not need to know the name of the objects, but rather we only need to know that there is an object on the scene, i.e. instead of using the real name of the objects, we replace them by their corresponding class (e.g. \textit{object=something} instead of \textit{object=bread} as explained in Section 5.2). The second step identifies the activities that were clustered as granular activity from the first step. In other words, all the complex activities are replaced in the input data set(s) with the label of \textit{Granular} and they are inferred with the following rule:

\text{if Hand(Tool\_use)} \rightarrow \text{Activity(\textit{Granular})} \quad (5.9)

Some examples of these complex activities are: cut, spread, unwrap, pour, flip, etc. This

¹The \textit{spread} activity has the most instances on the training data set, therefore most of the instances are correctly classified when this activity is chosen.
Results of semantic reasoning

In order to correctly infer complex activities more attributes have to be considered. For instance, we can take into account the type of object being manipulated, for example for the cutting and spreading activities, they both use the *knife* as a tool but they represent different activities. One main distinction between these two activities is the object they are acted on ($o_a$), either the *bread* or the *mayonnaise*, respectively. Therefore, a second stage is needed in order to extend our tree $T$ and to be able to infer those *granular* activities.

For this second step of our approach, we use as input the activities clustered as *Granular-Activity* from the previous step and we learn a new tree, which represents the extension of our previous tree. The final tree can be observed in Fig. 5.6, where the top part (magenta box) represents the general and most abstract level of rules to determine different *basic* activities (similar to the tree in Fig. 5.5) and the bottom part (purple box) presents the extension of the tree, which includes the current information of the environment. This means that, in order to identify which *granular* activity is being executed by the human, we need to know which objects (or classes) are being identified from the scene. Some examples of the extension of the tree are:

$$\text{if } \text{Hand(tool\_use)} & \& \text{ObjectInHand(Knife)} & \& \text{ObjectActedOn(Bread)} \rightarrow \text{Activity(Cut)} \quad (5.10)$$

$$\text{if } \text{Hand(tool\_use)} & \& \text{ObjectInHand(Bottle)} & \& \text{ObjectActedOn(Bread\_slice)} \rightarrow \text{Activity(Sprinkle)} \quad (5.11)$$

Observe that the second rule, see eq. (5.11), has as *ObjectInHand* the value *Bottle*, which presents the parental class of the identified object instead of the object itself, i.e. *pepper* has *Type Bottle*. This is an interesting concept and it is explained during the Knowledge and Reasoning section (see Section 5.4). Then, the next step is to test the accuracy of the obtained tree $T_{\text{sandwich}}$. In order to do that, we use the remaining 40% of the sandwich data set to test the accuracy of the obtained rules. In other words, we execute steps 4-7 from Algorithm 5.1, where we want to determine $c(n_{\text{sandwich\_test}})$, i.e. given the input attributes $n_{\text{sandwich\_test}} = \{\text{Move, Something, None}\}$ we determine $c(n_{\text{sandwich\_test}})$. Then, the *state-value* pairs from the test data set $n_{\text{sandwich\_test}}$ are of the form $(n_{\text{sandwich\_test}}(t),?)$, where $t$ represents the time (frames). Afterward, the target value is determined for each state of the system $c(n_{\text{sandwich\_test}}(t))$. Finally, the obtained results show that $c(n_{\text{sandwich\_test}}(t))$ was correctly classified with an accuracy of 92.57% compared with the ground-truth. To analyze the results we present the obtained confusion matrix in Table 5.1, where we can see that the
This figure shows on the top part (magenta box) the tree obtained from the sandwich making scenario \( T_{\text{sandwich}} \). On the bottom (purple box) is shown the extension of the tree, which represents the second stage of the learning process. Notice that in this branch of the tree, the name of the objects are used, since the current context defines the executed activity.

The main diagonal of the table indicates that human activities are correctly classified in most of the cases. Therefore, the semantic rules obtained from the tree \( T_{\text{sandwich}} \) generalize very well the human activities for this scenario. The remaining question is, can we use these rules to identify similar activities in a different scenario? This is analyzed in the following sections.

5.3.1. Advantages of the ISHAR method

Using a decision tree to obtain the semantic rules is not new. However, the pre-processing steps that we explained in Section 5.2 about the input of the attributes and classes to the system represent the novelty of our approach. For example, if instead of training the tree \( T_{\text{sandwich}} \) with the proposed triples, i.e. \( \{\text{move}, \text{Something}, \text{None}\} \), we would have trained with the input data obtained from the labeling tool (as it is usually done in the classical approach), then we would have obtained a highly complex tree. Particularly, if we would have used the...
Table 5.1. Confusion matrix (expressed in%) for high-level activities using labeled data from the sandwich making scenario. Where a=Reach, b=Take, c=Put something somewhere, d=Release, e=Granular and f=Idle.

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>86.9</td>
<td>3.4</td>
<td>3.4</td>
<td>0.6</td>
<td>0</td>
<td>5.4</td>
</tr>
<tr>
<td>b</td>
<td>2.6</td>
<td>76</td>
<td>8</td>
<td>13.3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>0.8</td>
<td>0.3</td>
<td>91.1</td>
<td>1.1</td>
<td>3.5</td>
<td>2.8</td>
</tr>
<tr>
<td>d</td>
<td>7.2</td>
<td>0.5</td>
<td>1</td>
<td>46.3</td>
<td>0</td>
<td>44.8</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
<td>1.2</td>
<td>0</td>
<td>98.7</td>
<td>0</td>
</tr>
<tr>
<td>f</td>
<td>2.6</td>
<td>0</td>
<td>0</td>
<td>2.3</td>
<td>0</td>
<td>94.9</td>
</tr>
</tbody>
</table>

Then, we would have obtained a decision tree similar to the one shown in Fig. 5.7. From this complex tree, we can observe rules such as:

\[
\text{if ObjectInHand(Blue_knife) \& Hand(not\_move)} \rightarrow \text{Activity(Take)}
\]

\[
\text{if ObjectInHand(None) \& ObjectActOn(Pepper)} \rightarrow \text{Activity(Reach)}
\]

\[
\text{if ObjectInHand(Cucumber) \& Hand(move)} \rightarrow \text{Activity(PutSomethingSomewhere)}
\]

It is possible to notice that this rules depend on the current context, i.e. the correctly identification of objects. Another disadvantage of these classical approaches to recognize the basic human activities compared to ours, is the fact that if a new data set is analyzed, then the obtained rules can not be re-used unless the same objects are used on the new data set.

For example, if we would like to infer the activities from the pancake making data set using the tree obtained from the sandwich making data set (see fig. 5.7), then this complex tree is
Figure 5.7. Illustration of the obtained decision tree according to classical methods [Park and Aggarwal, 2004]. We observe that the rules depend on the detected object and a generalization is not possible compared with the tree from Fig. 5.5.
not able to correctly classify the human activities. The main reason of the failure is that new objects such as \textit{pancake mix}, \textit{spatula}, \textit{pancake}, etc. are used and were not previously trained.

In the following, we demonstrate the above arguments quantitatively considering a 6-class problem for all the results shown here.

- **sandwich making vs. sandwich making, our approach**: First, we use our proposed methodology to infer human activities with our proposed triples using 60\% of the \textit{sandwich making} data set for training, we obtain the tree shown in Fig. 5.5. After that, we use the remaining 40\% of the data set to test the accuracy of the tree and the obtained recognition is 92.57\%. Afterward, we change the test option to 10-fold cross validation and the obtained tree is the same as the one shown in Fig. 5.5 and the recognition accuracy is 92.92\% \footnote{Notice, that the experiments performed in this subsection were done on the data that was manually annotated with our labeling tool for the complete task, i.e. more than 10,000 instances.}. The obtained tree using our proposed method has 6 leaves (number of rules) and the size of the tree is 10 nodes, which means that the complexity of the obtained tree is very low.

- **sandwich making vs. sandwich making, classical approach**: Second, we use the same parameters to train the tree using classical triplets and similarly we use the first 60\% of the data set for training and we obtain the tree shown in Fig. 5.7. Afterward, we tested this tree using the last 40\% of the data set and the accuracy is 91.65\%. Then, we change the test option to 10-fold cross validation and the obtained tree is similar to the one previously learned and the recognition accuracy is 92.17\%. This new tree has 60 number of leaves and the size of the tree is 70 nodes. Here, we can quantitatively demonstrate that the number of rules needed to correctly infer human activities increases 10 times compared with our proposed approach.

- **sandwich making vs. pancake making, comparative**: Finally, we performed a more complex test, which consist of using the obtained trees from Fig. 5.5 and Fig. 5.7, i.e. \textit{sandwich making}, to test a new data set in this case the \textit{pancake making} data set. The obtained results show that the recognition accuracy for the \textit{basic} activities using our method is 97.15\% whereas using the classical approach the accuracy decreases to 1.37\%. It is clear that the classical tree fails to identify the activities in the new scenario even though similar activities are executed. The main reason for this is that the branches of the tree from Fig. 5.7 depend on the trained objects and when a new data set with different objects is used, then a new tree has to be acquired due to the lack of generalization.
It is possible to highlight that with our proposed method we can exploit the simplicity and intuitiveness of the decision trees, which do not decrease the accuracy and it captures the general rules that can define such complex human everyday activities. Table 5.2 summarizes the above points.

**Table 5.2. Summary of the advantages of our approach compared to classical ones to compute trees.**

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy of recognition</th>
<th>Complexity of the tree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Testing with 40% of the</td>
<td>Rules generated</td>
</tr>
<tr>
<td></td>
<td>sandwich making data set</td>
<td></td>
</tr>
<tr>
<td>Classic</td>
<td>91.65%</td>
<td>60</td>
</tr>
<tr>
<td>Enhanced</td>
<td>92.57%</td>
<td>97.15%</td>
</tr>
</tbody>
</table>

**5.3.2. Generalization toward different scenarios**

The next challenge is to test the obtained semantic rules in unknown scenarios. In order to do that we use the set of hypothesis $H_{sandwich}$ to infer the human activities in new situations. Therefore we do not need to generate new hypotheses to infer human basic activities. Two new scenarios explained in Section 3.4 are used for testing: pancake making and setting the table (see Fig. 5.8). The terminologies for these new scenarios are $n_{pancake}$ and $n_{setTable}$, respectively. Therefore, we want to obtain the target values $c(n_{pancake}(t))$ and $c(n_{setTable}(t))$.

First, we input the data from $n_{setTable}$ into our Algorithm 5.1 and perform steps 4-7 and we obtained that $c(n_{setTable})$ was correctly classified with 91.53% of success and its corresponding confusion matrix is shown in Table 5.3a. This table shows that the best results are along the main diagonal, which indicates that most of the instances were correctly classified.

However, we can also observe that the off-diagonal elements of this table are higher than the results obtained for the sandwich making scenario (see Table 5.1). For example, for the activity Taking, 38.8% of the instances were wrongly classified as the activity PutSomething-Somewhere. From the obtained tree $T_{sandwich}$ (see Fig. 5.6) we can see that the difference between these two activities is the value of the attribute Hand_Motion, which in this case was incorrectly label as Move, therefore the activity was incorrectly classified as PutSomething-Somewhere.

The second new scenario $n_{pancake}$ was tested and we obtained that $c(n_{pancake})$ correctly
Section 5.3

Results of semantic reasoning

Figure 5.8. This figure shows some snapshots of the different data sets used to test the obtained semantic rules, refer to Section 3.4 for a better description of the data sets used. a) is the data set used to learn the decision tree; b) and c) are the new data sets used to test the obtained rules.

classifies 97.15% of the instances (see Table 5.3b). These results indicate that the obtained tree $T_{sandwich}$ generalize the definition of human basic activities in kitchen scenarios.

We would like to highlight that a similar tree is obtained if we would have used as training set the labeled information obtained from the setting the table or pancake making actions as shown in [Ramirez-Amaro et al., 2013a]. This implies that the obtained tree extracts the meaning of the human basic motions in a general form.

Table 5.3. Confusion matrix (expressed in %) of high-level activities from the set the table and pancake making scenarios. Where a=Reach, b=Take, c=Put something somewhere, d=Release, e=Granular and f=Idle.

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>a) Set the table</th>
<th>b) Pancake</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>a</td>
<td>86.9</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>5.5</td>
<td>44.4</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>f</td>
<td>1.6</td>
<td>0</td>
</tr>
</tbody>
</table>
5.3.3. Testing the robustness of the obtained semantic rules

The previous results were obtained using the labeled information of one human performing different activities using only his/her right hand. The labeling process was executed by one expert user. Therefore, in order to test the robustness of the obtained hypotheses $H_{sandwich}$, we need to conduct more experiments to collect more data and analyze the results. Hence, the following two experiments are evaluated:

1. **Different conditions and people using both hands:** In this experiment we tested the variability of possible styles to perform the same activity by analyzing the activities performed by several people (male and females) and by constraining the time conditions during the execution of the activities. This analysis was carried out using the scenario of making a sandwich since we have more subjects and different time constraints, see Section 3.4.2.

In this experiment 8 people not related to our project were instructed to prepare a sandwich. Half of the subjects were females and the other half were males (see Fig. 5.9). Additionally, each subject made the sandwiches under two conditions: normal and fast-speed. Fig. 5.9, shows that one of the male subjects (S8) was left handed, which implies that the granular activities (cut, sprinkle, etc.) were performed using the left hand instead of the right hand, in contrast to the other subjects. Additionally, we could notice that the meaning of the task making a sandwich was not interpreted the same way for all the subjects, for instance, in the case of subject 3, she prepared the sandwich with one bread slice (a half sandwich without toasting the bread).

![Figure 5.9](image)

**Figure 5.9.** Different styles of preparing a sandwich. Subjects 1, 2, 7 and 8 are males and the rest are females.

Therefore, in general, we can observe that each person has his/her own style and interpre-
Results of semantic reasoning

tation for the desired task, which in this case was to prepare a sandwich. Then, two questions arise: *do we interpret other humans motions the same way? is there a general principle that we as humans follow to execute basic activities?*

In order to answer the first question, we perform the following experiment. One person, different from the subjects that prepared the sandwiches, manually labeled the attributes of the human motions and the object properties. That was performed for the eight subjects, for the two conditions and for each hand. This means that in total 32 videos were labeled by the same person. Then, we use this information to test our obtained hypotheses $H_{sandwich}$.

The following results demonstrate that our system robustly identifies the performed activities even with all the above variations. The results suggest that even when the subjects perform the task in different ways, the obtained semantic rules are still valid with high accuracy, above 90%, this indicate that we are able to interpret other human motions quite accurately, which answers the first question.

Furthermore, it is possible to observe that the rules do not get affected if the subject is executing the task in a normal condition (94.75%) or under fast conditions (95.02%). Additionally, these rules are not gender dependent (Women=95.6%, Men=93.85%), this indicates that we all follow similar principles, which answer the second question (see Fig. 5.10). Moreover, the obtained rules are also valid when the motion of the left hand is being analyzed when an expert labeled the videos. The obtained average accuracy is 95.05% for this case.

2. **Labeling strategies by different persons:** In the previous experiment only one person, considered as an expert, labeled 32 videos. Then, the next challenge is to test if the labeling process performed by different persons affects the obtained results. The goal is to determine if we perceive the activities performed by other people in a different way and if this is reflected in the labeling. We expect that people uses different labeling strategies to segment human activities, but the obtained rules remain the same.

Therefore, we designed the next experiment. We asked four students\(^4\) that were not related to the project to label four videos of the same subject (S1): one under normal condition for the right hand, the second for the same condition for the left hand and the other two under fast condition, left and right arm. One important factor that has to be considered is that the students that labeled the videos were not instructed in any way about strategies to segment the videos. The students labeled the data using our provided labeling tool (see Section 3.4.4) to maintain the consistence between the motions and object properties and make a fair comparison.

---

\(^3\)These labels are later considered as the ground-truth and the person that labeled this data is considered an expert since it had a training stage.

\(^4\)The education of the students goes from high school to master students and two students were females and the rest males.
Figure 5.10. This figure presents the results of different people (male and females) making sandwiches under two conditions: normal and fast.

Then, we use the new labeled data as testing data set of our current tree from Fig. 5.6. The obtained results presented in Fig. 5.11, show that the accuracy of recognition decrease to 74.62% in the case of normal condition and to 78.75% for the fast condition. This results indicate that the obtained rules do not depend on the labeling process since different people perceive and segments the observations in different manners. These two percentages represent our base line for the recognition of human activities when an automatic perception system is implemented. It is interesting to mention that both cases correspond to the labels for the right hand. Specifically, these results indicate that a robot is not expected to automatically segment these motions at 100%, but to perform at least as an non-expert human does. Additionally, we could observe that the labeling strategy that women and males follow is pretty similar, as we expected.

The results obtained from testing human activities from both hands provide completeness to our approach. We can observe that the rules obtained from the right hand ($H_{sandwich}$) are also valid to infer the activities that humans perform with the left hand. This indicate that the obtained semantic rules are independent of the hand that is performing the action, due to the fact that the meaning remains the same.
5.3.4. Enhancing activity recognition using ISA and ISHAR

The next step is to test the robustness of the obtained tree (see Fig. 5.6) using as input data the classification results obtain from the ISA algorithm (see Section 4.5) instead of the manually labeled data. Therefore, we use as new input the classification results for the sandwich making videos using the ISA algorithm. This means that we use the ISA method in order to recognize the low-level motions such as: move, not move and tool use, using the video from the first subject preparing a sandwich in a normal condition to train the spatio-temporal features for a 3-class problem.

Afterward, we use the learned features from the second subject under a fast speed condition to test the observed new visual features and the recognition accuracy is to 66.67%. Then, using the recognized motion results, the correctly classified instances using the tree from Fig. 5.6 is 81.1%. The accuracy of these results is lower than the manually labeled results, since the accuracy is affected by the errors produced by the incorrect classification of the human motions generated by the ISA algorithm (the accuracy of the ISA for the high-level motions is
66.67%). The confusion matrix is depicted in Table 5.4.

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Idle motion</td>
<td>37.50</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>25</td>
<td>37.5</td>
</tr>
<tr>
<td>b) Reach</td>
<td>7.84</td>
<td>74.5</td>
<td>0.39</td>
<td>3.13</td>
<td>4.7</td>
<td>9.4</td>
</tr>
<tr>
<td>c) Take</td>
<td>0</td>
<td>4</td>
<td>56</td>
<td>12</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td>d) PutSomethingSomewhere</td>
<td>0</td>
<td>0</td>
<td>6.5</td>
<td>62.32</td>
<td>0.13</td>
<td>31.02</td>
</tr>
<tr>
<td>e) Release</td>
<td>0</td>
<td>0.96</td>
<td>0.96</td>
<td>2.88</td>
<td>69.23</td>
<td>25.96</td>
</tr>
<tr>
<td>f) Granular</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.85</td>
<td>0</td>
<td>98.14</td>
</tr>
</tbody>
</table>

Afterward, we used the motions recognized as granular activities (see Table 5.4) and reclassified them following the ISA methodology. In the case of the sandwich making data set, the granular activities are one of the following categories: cutting, unwrapping, spreading and sprinkling, and the classification accuracy is 72.2%. Then, we used the motion recognition results obtained from the extended ISA algorithm for the pancake making videos. Please take into account that the tree was generated from a 60 fps video and the new test set contains information at 24 fps, which means that the life-time of the activities is shorter than the training data. Nevertheless, the correctly classified instances are 93.60%. This high accuracy is due to the fact that the high-level activities were correctly classified 81% of the time.

The final human activity recognition after the two stages is around 87.3%, i.e. the first stage comes from the recognition of the ISA algorithm and the second stage consist of using this recognition as input data into the semantic rules and the average of those results is the final recognition accuracy. A video where more details about these experimental results can be found in the following link: https://www.youtube.com/watch?v=jHf338db6gE

Using this methodology the complexity of action recognition decreases, since we proposed to first classify the low-level activities and then use that information to infer the high-level activities. If we want to classify these high-level activities, the classification accuracy decreases

---

5This confusion matrix is obtained frame-wise.

6In this experiment the videos from subject one during the 2nd take (S01/T02) were used as training data set for the motion recognition, while subject one take one (S01/T01) was used as testing. Therefore we use the results from the testing data set.
substantially, due to the fact that certain activities like reaching are misclassified as releasing due to their similarity.

The following two important aspects of our work needs to be highlighted. On one hand, if the human action recognition for complex and real-world scenarios like the ones presented here is classified using the classical approach, i.e. trying to recognize the high-level activities, such as: reaching, taking, releasing, cutting, sprinkle, etc., then the classification accuracy is around 25% (see Section 4.2.3). This is the classical approach that is used for action recognition, where the state-of-the-art techniques for goal-directed activities reported a maximum accuracy of about 43.6% [Le et al., 2011]. On the other hand, if our method is used, i.e. split the classification problem into first recognizing the three high-level motions (move, not move and tool use), and later use the classification output as input for the reasoning engine, then the action recognition accuracy increases to approximately 82% as an average of the two tested data sets.

Therefore, with our proposed framework the activity recognition accuracy increases by 55% compared to the state-of-the-art method [Le et al., 2011] using the same data sets (see Table 5.5). These results indicate that our two-stage approach is two times better than the best results reported in the literature and more than three times better than the average results of the state-of-the-art methods.

**Table 5.5.** This table summarizes the recognition results of the ISA implementation without and with our proposed system.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>ISA for 9-class problem</th>
<th>ISA with ISHAR for 9-class problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sandwich making</td>
<td>25%</td>
<td>81%</td>
</tr>
<tr>
<td>Pancake making</td>
<td>NA</td>
<td>93%</td>
</tr>
</tbody>
</table>

Even though the results presented here are for off-line classification, the extension to on-line classification is possible. This is due to the fact that once the video features are extracted from the training process, the classification results are obtained in seconds.

### 5.3.5. Activity recognition using Color-Based technique and ISHAR

The next important step is to extend our system from off-line to on-line recognition using the Color-Based technique as presented in Section 4.6. This implies that we use as input the data obtained from the automatic segmentation of human motions and object properties in order to test the on-line recognition (see Section 4.6.2).
First, we applied the learned rules obtained from the training data set to the same data set, i.e. sandwich making. In order to test the semantic rules we use a different subject than the one used for the training and two conditions were tested: normal and fast speed. The results show that the accuracy of recognition is about 81.74% (normal condition = 79.18% and fast speed condition = 83.43%). One example of the obtained confusion matrix from this scenario is depicted in Fig. 5.12(a), where the output of the left hand recognition is shown.

![Confusion Matrix](image)

**Figure 5.12.** The left figure shows the confusion matrix of the left hand recognition for the scenario cutting the bread. The right figure shows the confusion matrix of the new scenario pouring the pancake mix for the right hand recognition.

The obtained errors from the activity recognition are generated by the misclassified objects from the perception module, especially for the sandwich scenario, when the object knife is occluded by the hand and the bread, as shown in Fig. 5.13.

Then, we tested the semantic rules on a new scenario pouring the pancake mix, in which the activity pour has not yet been learned. Nevertheless, the system is able to identify that a new activity has been detected since the observed object properties are not in the learned model. Then, the system via active learning asks the user to provide a name for the unknown activity. After the new activity has been learned the system can correctly infer it. The results indicate that the accuracy of recognition is around 88.27% (see Fig. 5.14). For this new scenario, the obtained confusion matrix of the recognition of the right hand activities is depicted in Fig. 5.12(b).

After that, we tested our system with a different task of flipping the dough, similarly to the above example this demonstration has not being trained to recognize the new granular activity of flipping. The obtained results, shown in Fig. 5.15 demonstrate that our system is able to learn via active learning the new demonstrated activity with an accuracy of 76% [Ramirez-Amaro et al., 2015]. This lower accuracy is due to the errors from the perception system during the identification of the object properties.
Results of semantic reasoning

**Figure 5.13.** These figures show the *on-line* generated signals and the immediate inference of human activities performed by the right hand of the *sandwich making* scenario when the subject is in the fast condition. a) shows the signals obtained by the Color-Based technique where the vertical lines indicate the automatic segmentation and recognition of the human activities for the right hand. Whereas, b) shows one snapshot of our system to infer the human activities *on-line.*
FIGURE 5.14. These figures show the on-line generated signals and the immediate inference of the human activity performed by the right hand of the pancake making scenario. a) shows the signals obtained by the Color-Based technique where the vertical lines indicate the automatic segmentation and recognition of human activities for the right hand, whereas b) shows one snapshot of our system to infer human activities on-line.
The important contribution of these results is the definition of rules that make the inference of human activities in different scenarios possible, with an overall accuracy of 85%, considering known and unknown activities. The above is possible even when a very simple hand and object recognition system is employed to segment the motions and object properties automatically. Another, very important feature of our system is the possibility of recognizing human activities of both hands at the same time as depicted in Fig. 5.16. It is possible to observe that the same tree is used to recognize activities for both hands without further modifications.

Fig. 5.17 presents the obtained results under all the tested constraints, where the highest accuracy is obtained by the human expert label with 94.89% and the automatic segmentation and recognition obtained by the Color-Based system is very close to this accuracy with 89.79% for the pancake making data set. Notice, that the accuracy obtained by the non-expert persons
that label the data is 81.39\% for the \textit{sandwich making} data set. Finally, the lowest accuracy is obtained by the ISA algorithm using our two-step method with an accuracy of 74\%. This is due to the fact that such technique is mainly to recognize more general activities, e.g. move, not move, tool use and not for specific activities as the ones that we have analyzed in this work, i.e. reach, take, pour, cut, etc. Nevertheless, the method with the lower accuracy improves the performance of the best result of the state-of-the-art method.

![Analysis between the system and the human recognition](image)

**Figure 5.17.** This figure shows the results of recognition of the activities made by: a) Human experts, b) Non-expert humans, c) the Color-Based method and c) the ISA algorithm for the data sets of \textit{pancake} and \textit{sandwich making}. It is possible to notice that the automatic recognition done by the color-based system is closer to the human expert recognition and more accurate than non-expert human.

It is interesting to notice that the best results were obtained using a very simple technique (e.g. Color-Based) to track the hands and to determine the object properties that were acquired \textit{on-line} automatically. This presents the first step towards a generalization of those kinds of activities.

Another problem that needs to be addressed is regarding the knowledge that robots should have about objects, places and actions when performing more and more complex tasks. They must be able to recognize objects, i.e. know what they are, and how they can be used. Having grounded symbolic models of its actions and related concepts allows the robot to reason about its activities and improve its problem solving performance. In order to use action-related
Knowledge and reasoning engine

In the field of robotics, especially in human-robot interaction, it is important to enable robots with decision-making capabilities in order to increase their adaptability and flexibility to different environments. These reasoning capabilities depend on the correct recognition of the current human motion in order to predict the next motion. One powerful tool to achieve that is through Semantic Reasoning. The semantics (construction of the meaning) can be enhanced if a Knowledge-Based and a Reasoning Engine are integrated into the system [Russell and Norving, 1995]. Semantics can be divided into two parts: Construction of the Meaning and Knowledge Representation. In the previous sections the construction of the meaning (semantic rules) was explained and tested. This section focuses on the Knowledge Representation and its Reasoning Engine.

Knowledge and reasoning play a crucial role in dealing with partially observable information. This is possible since they are capable to infer or predict different behaviors, in a way as we (humans) do and expect. This is partly obtained due to the fact that the knowledge base system can combine general knowledge with the current perception of the world to infer hidden aspects of the current state. The key factor here is to define mechanisms to obtain appropriate reasoning rules to be able to infer meaningful relationships from the perceived environment. Therefore, the following questions arise:

- How those rules have to be defined?
- Who defines those rules, an expert?
- How can we guarantee that these rules are going to be valid for different situations? or What is the level of generalization of those rules?

These questions need to be answered properly since a good Knowledge-Reasoning system should be able to adapt and be flexible to changes in the environment by only updating the relevant knowledge.

Developing a proper knowledge base requires a careful process of analysis of the domain, choosing the appropriate vocabulary and encoding the reasoning engine to obtain the desired
infereces [Russell and Norving, 1995]. The last point is very important since the main goals of the reasoning process are: 1) to create representations of the world, 2) use a process of inference to derive new representations about the world, and 3) use these new representations to deduce what to do next.

Representing abstract concepts such as action, space, time, physical objects, among others, is sometimes achieved with an ontology representation. A language commonly employed to represent knowledge is the Web Ontology Language (OWL) which is an action representation similar to Description Logics (DL) such as Prolog queries. This language uses semantic data modeling based on triplets such as Resource Description Framework (RDF), which is an underlying technology of OWL. There is an OWL and RDF API for Prolog, which simplifies the interface of the ontology with the logic engine. We use this API to develop our Knowledge Based framework.

In this work, the knowledge is represented in OWL. This XML-based format allows to formally describe relational knowledge in a Description Logics (DL) dialect. For formally modeling knowledge, it is very useful to distinguish between general relations and environment-specific information. In OWL, this is reflected by the distinction between classes and instances. Class knowledge is described in the so-called TBOX (terminological box); knowledge about instances of these classes are contained in the ABOX (assertional box). The relation between classes and instances is similar to object-oriented programming.

5.4.1. Knowledge framework

We used KnowRob [Tenorth and Beetz, 2013] as our base line ontology, which is an extension of the OpenCyc ontology, a general upper ontology that covers a broad range of human knowledge. KnowRob is implemented in Semantic Web Implementation (SWI) Prolog [Wielemaker et al., 2008] (see Fig. 5.18), which means that, Prolog is used for loading, storing and reasoning about the knowledge which is represented in OWL. The OWL files are loaded into the system, i.e. these are internally stored as triples. Finally, the results are visualized in java applets as shown in Fig. 5.18.

It is important to note that KnowRob separates knowledge about the world (which is, as far as possible, represented in OWL) from implementation issues and deduction rules (which are implemented in plain Prolog). The formal semantics of OWL natively exhibits this separation. In order to benefit from the semantic properties of OWL, it is necessary that the information, which is read from external sources by Prolog predicates, is transformed into a representation that is compatible to the OWL knowledge.

We mainly use two branches of the KnowRob ontology: the TemporalThings and the Spa-
5.4 Knowledge and reasoning engine

Figure 5.18. Knowledge framework called Knowrob which consist of different modules such as reasoning, knowledge, visualization, etc.

The first one contains the important subclasses of Actions and the second one describes the abstract spatial concepts such as places and object classes. KnowRob loads the information of the ontology as RDF triples and the reasoning is done with the Thea OWL parser library [Vassiliadis et al., 2008].

Reasoning with Computables is another important characteristic of KnowRob, since it provides the possibility of compute new relations during the reasoning process (on demand) instead of defining them manually. This is important since the environment where we evaluate our system is mostly dynamic, i.e. the states change over time. There are two kinds of computables: Computable Classes, which create instances of their target class, and Computable Properties, which compute relations between instances (see Fig. 5.19).

Computable Classes are more straightforward to define, since they retrieve stored instances from a data set, such as from MySql or they are created when a new instance is perceived by
any external sensor (e.g. cameras) of the system. On the other hand, to define Computable Properties is not an easy task, since they determine new relationships between objects and actions. Therefore, in this work we focus our attention on the appropriate definition of those properties.

5.4.2. Proposed algorithm for expanding the ontology

In order to define meaningful relationships between actions and objects, we use the obtained semantic rules described in Section 5.3. These rules generate new individuals and new relationships between individuals (objects properties). Those object properties are obtained by the definition of new computables. Then, the created instances and relationships are added to the ontology as part of the inferred knowledge base. A very general way of representing the inputs and outputs of the knowledge and reasoning system is shown in Algorithm 5.2.

In the previous subsections, we have demonstrated that the obtained semantic rules of the human activities do not change under different constraints, e.g. time constraints, different sce-
Algorithm 5.2 General representation of inputs and outputs of the knowledge and the reasoning system.

Require: $KB \leftarrow \text{KnowRob}$

1: \textbf{TELL} ($KB, \text{Perceive\_Object}(O_{\text{instance}}))$
2: \textbf{TELL} ($KB, \text{Perceive\_Action}(A_{\text{instance}}))$

{In this step the \textit{Computable Classes} are executed and new instances are generated into the KB}

3: \text{human\_activity} $\leftarrow$ \textbf{ASK} ($KB, \text{Relation\_Query}(?)$)

{In this step the \textit{Computable Properties} are obtained from the KB}

4: \textbf{return} \text{human\_activity}

narios, gender or the hand used to execute the activity. In this subsection, we use the obtained semantic rules to improve our reasoning engine and the knowledge base. For example, the inferred goal $c(s)$ obtained from the semantic rules is validated in such a way that if the system succeeds to infer a human activity, then this value defines the inferred goal ($g$). In the case of failure, the system executes the Knowledge Base submodule and this generates an equivalent state of the system to infer the correct concept value. In this case we refer to this value as the enhanced concept value $c(s_{\text{enhanced}})$, which is later used as the inferred goal $g$, see Fig. 3.2.2.

The rules obtained from the decision tree can be easily programmed in any kind of language. Nevertheless, some first order logic languages such as Prolog could enhance the system with more inference and reasoning capabilities. By reasoning we mean that some facts are derived (inferred). These facts are not necessarily expressed in the ontology or in the Knowledge Base explicitly. Therefore, as part of our framework, those rules represent an important part of our Reasoning Engine, since they help to recognize human everyday activities. Hence, we introduce new features to our reasoning engine to infer new relationships between motions, objects and activities.

One key point from our reasoning engine is that through inference mechanisms, new representations about the world are derived, i.e. those are not manually included into the Knowledge Base. To enhance our ontology, first we made some modifications on the KnowRob ontology, mainly to reduce its complexity. In general, to obtain an instance of the human basic activities, such as: \textit{reach}, \textit{take}, \textit{release}, etc., six node levels are necessary to travel in the KnowRob ontology path, and those classes are manually included into the knowledge base. Whereas, in our ontology the maximum distance is 4 node levels (see Fig. 5.20).

Another important advantage of our enhanced ontology compared to KownRob is that such \textit{basic} human activities are inferred and not manually included in the ontology. In other words, our ontology does not contain the classes of \textit{reaching}, \textit{taking}, etc., rather it computes them on...
Figure 5.20. Comparison between KnowRob and our modifications to the ontology. Top: It depicts the KnowRob ontology, the human activities (pink boxes) such as: reach, release, take, etc are part of the basic knowledge base, i.e. they are manually included. Bottom: Our proposed ontology has a new structure, which is less complex since it needs less classes (defined meaningfully) and less steps to create an instance of the activity.

demand as instances of the class motion. This means that new classes are added on demand (see Fig. 5.21). Then, our basic ontology is optimized since it grows on demand. In order to allow this dynamic growing we use the computable property from KnowRob and new Class Computables need to be defined.

The structure of our ontology is defined mainly based on the obtained semantic rules. The human motions are classified as move, not_move or tool_use and the inferred activities define new sub-classes on demand. This is very important if we want to predict the intentions of people. For example, if the activity is a sub-class of not_move, then the human activity are either take or idle according to the obtained semantic rules.

One important aspect in the reasoning engine is the grounding, which defines the connections between the reasoning process and the real environment. The state of the environment is given by the sensors of the system (robot), which in this work is emulated by the labeling information which is stored in a data base (MySql).

Then, we define three Class Computables, one for each subclass of the class Motion. Af-
terward, we can retrieve the generated instances as follows:

\[
\text{rdf\_instance\_of}(\text{?InstM, comp\_humAct \text{'}Motion'}). \tag{5.17}
\]

where, \(?\text{InstM}\) is the obtained instance of the class \text{Motion} and the output is similar to: \(\text{?InstM} = \text{not\_move\_0}, \text{?InstM} = \text{move\_1}, \text{?InstM} = \text{tool\_use\_6}, \text{etc.}\) This represents the second step of the Algorithm 5.2. An illustration of the obtained instances can be seen in Fig. 5.21, right.

**Figure 5.21.** Illustrative example of the semantic description between activities and objects. It is possible to observe the dynamic growing of the knowledge base (Inferred knowledge) using the inference mechanism described and implemented in this work.

The next step is to define the two properties of the objects: \text{ObjectActedOn} and \text{ObjectInHand}. Hence, we create one \text{Computable Property} for each of them. Then, we can query the properties of the individuals as follows:

\[
\text{rdf\_triple}(\text{comp\_humAct \text{'}objectOn\text{'}, ?F, ?V}). \tag{5.18}
\]

\[
\text{rdf\_triple}(\text{comp\_humAct \text{'}objectIn\text{Hand'}, ?F, ?V}). \tag{5.19}
\]
where objectActOn and objectInHand represent the target names of the defined Computable Properties. \( ?F \) represents the selected frame and \( ?V \) corresponds to the value of the property, which is the name of the object. Those queries are equivalent to the first step from Algorithm 5.2. Possible output values from query (5.19) are: \( ?F = 'o-1' \), \( ?V = 'none' \), which means that for occurrence 1, the value of ObjectInHand is none. On the other hand, the output from the query (5.18) is: \( ?F = 'o - 1' \), \( ?V = \text{PancakeMix} \), this means that for the occurrence 1 the ObjectActedOn is of the class PancakeMix.

The next step is to semantically relate the instances from the class Motion and the object properties previously described. This is achieved by using the rules obtained from section 5.3. For example, to infer the activity Take, we have the following prolog predicate:

\[
\text{humanAct}(?Occ,\text{take}) : - \\
rdfs\text{-instance\_of}(?\text{Inst}\text{M}, \text{comp\_humAct} : '\text{Motion}'), \\
\text{InstM} = '\text{StandingStill}', \\
rdf\_\text{triple}(\text{comp\_humAct} : \text{objectInHand}', Occ, ?V). \\
(\text{V} = '\text{Something}'; \text{V} \backslash = '\text{none}').
\]

where ?Occ is the occurrence number we want to infer and take is the name of the inferred class. From the above predicate we can see how the instances of the class Motion and the objects with the property of ObjectInHand are semantically described and represented. This exemplifies the last step of our Algorithm 5.2. Similar prolog predicates are defined for the remaining rules. An illustration of the growing of the knowledge base with the inferred instances from the computed relationship between the Motions and the objects properties can be seen on the right side of Fig. 5.21.

### 5.5. Results of knowledge base and reasoning engine

Our system is enhanced by adding the knowledge and reasoning sub-modules, since they can deal with incomplete information to infer human activities, even when we use the semantic rules in an untrained scenario. Fig. 5.2 depicts the combination of the semantic module and its improvement using the Knowledge and Reasoning Engine. For example, given the perceived state of the environment, our system infers the concept value \( c(s) \) using the semantic rules. If the system succeeds to infer the observed human activity, then this value defines the inferred goal \( g \). In the case of failure, the system executes the Knowledge Base sub-module and this generates an equivalent state of the system to infer the correct concept value, in this case we
refer to this value as the enhanced concept value $c(s_{enhanced})$, which later is used as the inferred goal $g$.

The above procedure is possible since our system is able to backtrack the knowledge-based ontology until it identifies a similar parental class to the one perceived. Therefore, the inference using this new instance is possible. For instance, if we take the sandwich scenario and lets assume that the perception module recognizes the hand motion $m=tool\_use$ and a new object pepper with the object property $o_h(t)$. Then, for this example, our system fails to infer the human behavior since the object pepper was not defined. However, our system compensates this failure by looking into the knowledge-based to find the parental class of this new object, which in this case pepper belongs to the class Bottle. This produces a new instance bottle_1, which represents the equivalent state of the system $s_i$. Using this equivalent state $s_i$, our system is able to correctly infer the human activity using the corresponding predicate from the Reasoning Engine. In this example the possible outputs could be activity=$Pouring$ or activity=$Sprinkle$, depending on the value of $o_a(t)$ (see Fig. 5.6, purple box).

In order to semantically represent the objects inside our working environment, we need to describe them in a structured and articulated form. This is achieved with the description of the semantic map, which contains information about the properties and relations between the objects inside the environment. This is described using OWL properties, meaning that the objects defined in the semantic map are described as instances of the classes inside the ontology, i.e. each object instance inherits the properties and relationship of its class. For example, Pancake_1 is an instance of the Class Pancake (see Fig. 5.21), which is defined in the semantic map and inherits the motion-object relationship described in the queries (5.18) and (5.19).

For example, we can ask the system the following query: humanAct(?Occ,?Activity) and the possible outputs are as follows (see Fig. 5.22):

![Figure 5.22](image_url)

**Figure 5.22.** Examples of the inferred activities and highlighted objects on the semantic layer. The numbers above the pictures indicate the time where the activities are inferred, i.e. the Occurrence number (Occ). For example, Occ=0 infers the activity=$idle$, Occ=1 activity=$reach$, Occ=2 activity=$Take$ and Occ=3 activity=$Put\ something\ somewhere$. 

141
When $\text{Occ} = 0$, we would like to know what is the inferred activity from the occurrence number 0. The output is $\text{Activity} = \text{IdleMotion}$ as shown in the 1st column of Fig. 5.22. Additionally, we retrieve the value of the property $\text{ObjectInHand}$, which according to the semantic rule should be equal to 'none' as we obtained from the query. Therefore, none of the objects in the semantic map is being highlighted.

When $\text{Occ} = 1$, then the output for the next inferred activity is $\text{Activity} = \text{Reach}$, see the 2nd column of Fig. 5.22. In this figure we can observe that the instance of $\text{PancakeMix}_1$ is being highlighted in a yellow color, which indicates that this instance has the property of $\text{ObjectActedOn}$. On the other hand, none of the objects had the property of $\text{ObjectInHand}$. This means that when a reaching activity is being identified, first we look at the object that we will manipulate, but we do not have that object in our hands yet. This represents an important finding since this information helps to decrease the search space for the perception module. This implies that we could focus on the object that the human is currently seeing and infer that, most probably, it is the object that is going to manipulate in the close future. Further analysis on this topic is being considered as future work.

When $\text{Occ} = 2$, then the inferred activity is $\text{Activity} = \text{Take}$, see the 3rd column of Fig. 5.22. This indicates that the object that is being colored in red has the property of $\text{ObjectInHand}$. We can notice that as expected, the object that in the previous motion was identified as $\text{ObjectActedOn}$ becomes the object being manipulated (i.e. $\text{ObjectInHand}$).

When $\text{Occ} = 3$ then the inferred activity is $\text{Activity} = \text{PutSomethingSomewhere}$, see the 4th column of Fig. 5.22. It is possible to observe that the object $\text{PancakeMix}_1$ is still being highlighted in red, which means that even when the human is performing a different activity from the previous occurrence ($\text{Occ} = 2$), the object remains with the same property of $\text{ObjectInHand}$.

Then, instead of asking about the inferred activity, we could also ask the system, the time that certain activity was performed, by giving the value of the desired activity, for example $\text{Activity} = \text{release}$ and we could have as a result all the occurrences where this activity was inferred, such as $\text{Occ} = 11$ or $\text{Occ} = 17$ or $\text{Occ} = 23$ and the properties of the objects are retrieved as well. In this case during all occurrences, they were equal to $\text{ObjectActedOn}=\text{none}$ and $\text{ObjectInHand}=\text{none}$, therefore, the visual output is similar to the 1st column of Fig. 5.22.

From the above examples we could observe that if we infer the activities performed in
consecutive instances, those activities follow a pattern that can define the plan that the human follows during the execution of the final task. For example, we notice that humans first reach the target, then take the object. Such mechanisms help to improve the vision recognition systems, since the distinction between two similar activities such as: Reach and Release can be detected given its meaning and the time when they are performed. It is important to notice that the differentiation between the activities of Reaching and Releasing is very challenging since the Cartesian trajectories of both activities are very similar.

We could conclude that representing abstract concepts such as action, space, time, physical objects, among others, are better achieved with an ontology representation, as we have presented in this section. Such types of representations are very powerful when we need to control a robot in a dynamic physical environment, since those representations give the system the capability of being more adaptive and flexible to new situations due to its ability of generalize and abstract the problem.

The contributions of this work regarding the Knowledge-Base and Reasoning Engine are:

- The description of a new model for the semantic environment, for the pancake-making scenario.
- The representation of a new hierarchical structure to define new classes on the ontology in a meaningful manner.
- The definition of new Computables Properties to extract properties of an object, such as: computeObjectActedOn, computeObjectInHand, etc.
- The implementation of new prolog predicates to relate the new obtain instances to the object properties in order to infer the human activities.

### 5.6. Summary and contributions of this chapter

The main goal of this chapter is to present the module responsible of interpreting the data obtained from the previous Chapter 4, which can handle different sources such as: videos, motion capture systems, virtual environments, etc. These input sources are considered as the perception of the environment. This indicates that our system process these information in order to infer the human intentions. Particularly, the current Chapter is responsible of identifying and extracting the meaning of human motions by generating semantic rules that define and explain these human behaviors, i.e. it infers the high-level human activities, such as: reach, take, release, pour, cut, etc.
This means that complex human activities, such as *Put something somewhere* could be demonstrated in different forms as shown in Fig. 5.23. Even when the patterns (positions or/and velocities) are different, our reasoning system is able to interpret those observed signals as the same behavior. This represents a big advantage of our system respect to the current state-of-the-art (see Section 2) since our system first observes the demonstrator and recognizes the learned activity by *understanding* the principles that define the observed activity via different levels of abstraction.

![Diagram](image)

**Figure 5.23.** This figure shows the signals obtained from the observations, with a zoom analysis of the obtained signals under the same activity, e.g. *Put something somewhere* is shown over time. We show that even when those signals have different patterns, they are correctly classify as the same activity due to the extraction of their semantics.

This module represents the core of our framework. It is responsible for identifying and extracting the meaning of human motions by generating semantic rules that define and explain these human motions, i.e. it models the high-level human activity value \( c(s) \) such as: *reach*, *take*, *pour*, *cut*, etc. Afterward, it evaluates that \( c(s) \) was inferred correctly. If this evaluation fails, then the system uses its knowledge base to obtain an alternative triplet \( s_i = \{ m_i, a_{i1}, o_{ih} \} \) which contains instances of the desired class or its parent class in the case of untrained objects.
For example, if during the classical training method we use $o_a = \text{something}$ and $o_h = \text{none}$, then the system is restricted to only accept those values in order to correctly infer $c(s)$. However, we enhance the system by adding the knowledge base, which means that, if we have new values $o_a = \text{pancake}$ and $o_h = \text{spatula}$, then the system looks for the corresponding class and infer the activity value $c(s_{\text{enhanced}})$. This is a very important feature of our system, since with the knowledge base we do not need to recompute the semantic rules every time we have a new situation, i.e. the generated rules are preserved even in different scenarios.

First, sub-section 5.3 initially presents the semantic rules obtained off-line using as training the sandwich making data set using ground-truth information. Later, in sub-section 5.3.2, the obtained rules are tested off-line in two new scenarios: pancake making and setting the table data sets. Then, sub-section 5.3.3 demonstrates the robustness and re-usability of the obtained tree using the sandwich making data set under different constrains, e.g. time conditions, labeling strategies, etc. To the best of our knowledge, such validations are not presented in any other work, but they represent a very important aspect when discussing about the generality of the obtained results. Additionally, the presentation of these results are highly relevant to explain that the proposed method can be implemented in on-line systems, which is explained in the next Chapter 6.

We also include a validation of the used ground-truth in sub-section 5.3.3, where the segmentation of the ground-truth made by “an expert” is tested using random people to segment the data. In this regard, the work by Koppula and Saxena [2013b] presents an interesting analysis on the ambiguity in temporal segmentation, where they considered various types of segmentation concluding that the segmentation method proposed by them was the closest segmentation to the ground-truth. However, no further analysis has been made to include the different segmentation made by random people. Then, our findings complement those results by including that missing analysis. Our obtained results demonstrate that the segmentation of human motions is not the same due to the fact that different people have their own strategy to segment human behaviors. Therefore, we can not expect a 100% accuracy when an artificial system is performing this segmentation automatically. Then, the obtained results suggest that the worst expected performance should be similar to the one obtained by random people segmenting motions from others (around 76.68% accuracy). Hence, the obtained results from sub-section 5.3.5 are better than expected (average of 87.44% accuracy) using the simplest and on-line Color-Based algorithm enhanced with our semantic representations.

Finally, sub-section 5.5 shows the results of enhancing our system with the inclusion of the Knowledge-Based and the Reasoning Engine to effectively improve the dynamical grow of the ontology-base knowledge representation.
We validate our proposed semantics and reasoning engine in a robotic platform under highly complex and different scenarios. This integration consists of different steps. First, the on-line integration of different input sensors (see Chapter 4). Then, using the extracted observations, our system infers the observed human behaviors (see Chapter 5). Finally, this Chapter explains the integration of the above steps in a robotic platform, in this case the iCub.

First, Section 6.1 introduces the main components of our system in the robot. Then, Section 6.2 explains the libraries and processes used to incorporate our framework in the control loop of the iCub. Section 6.3 presents the assessment of our framework that allows robots to make on-line inferences. Then, Section 6.4 describes the enhancement of our system with egocentric views. Section 6.5 demonstrates the robustness of our system using virtual environment as new demonstration scenario. Finally, Section 6.6 concludes this chapter.
6.1. Transferring the inferred task to robots

This final module integrates multiple levels of sensory input observations using our framework. These input sensors include: videos of humans performing the desired activity, egocentric observations of the activities, virtual environment demonstrations, etc. Specifically, our framework combines the perception and inference modules into the control loop of a robot. In order to achieve this, we have to address several challenging and difficult problems. For instance, one important factor to consider for the integration of our system into a robot is the transition from off-line learning to on-line learning systems. The perception and semantic modules can easily be implemented for off-line systems as we have shown in Section 5.3.4. Nevertheless, the inclusion of such off-line systems into a robotic platform is not useful since it is not reactive and will make the robot very limited in terms of its functionality.

Therefore, it is required of perception and semantic modules to work on-line as we have presented in Section 5.3.5. For such on-line systems, we have to consider the possibility of learning new activities on-demand as we proposed in Section 5.3. Additionally, the perception and semantic systems need to be as fast and accurate as possible. In other words, the communication between the perception and inference modules have to be in the real-time since these modules have to be implemented together with the control loop of the robot (see Fig. 6.1).

**Figure 6.1.** This figure shows the general overview of the steps needed to transfer the observed task into the iCub robot.

Fig. 6.1 shows the different modules that conform our framework. The module *Execution of the goal*, takes as input the inferred human activity \( g \) from module 2 (blue rectangle from Fig. 6.1). Thereafter, the robot executes the skill planer, which indicates the motion primitive that should be executed in order to achieve a similar task as the one observed. For example, after the robot has inferred the human behavior \( g \), it accesses the *skill plan* module,
Assessment and integration of our framework in complex scenarios

where a list of sequence of primitives is stored. This list of primitives defines specific tasks. The output of this module consists of executing the primitives \((p_n)\) from a library within the retrieved plan, where \(n\) is the number of executed primitives. For instance, if the inferred activity is reaching, then the skill plan commands the primitives: \(p_1 = \text{find the desired object},\) \(p_2 = \text{approach the object},\) etc. This commands the motions of the robot \((q_d)\), which is the feedback information for the control loop. This control is executed until the desired task has been successfully achieved. Then, the next primitive is loaded and executed in the same form. This process is executed until the last primitive is generated. The control loop includes different control schemes depending on the definition of the primitives. Examples of the controls used in this thesis are: Position-based Visual Servoing, Joint-Space control using Inverse Kinematics, Gaze control, etc.

The following sections explain in further detail the modules, applications, resources and libraries that were used and implemented in order to achieve the above steps.

6.2. Implementation on the iCub humanoid robot

As the experimental platform we use the iCub robot which consists of 53 DOF [Metta et al., 2008] (see Section 3.5). In this work, we used a total of 25 DOF, in specific, we used 16 DOF of the right arm, 3 DOF of the torso and 6 DOF of the head (see Fig. 6.2).

Regarding the software, we used Yarp [Metta et al., 2006] and the iCub libraries [Pattacini, 2010]. The iCub software is developed on top of Yarp, which is a set of libraries that takes care of defining a soft real-time communication layer to connect with the hardware interfaces. The Yarp system is defined in terms of protocols, which address inter-process communication between ports and devices. The ports are able to deliver messages of any size across the network using a number of underlying protocols (TCP/IP, UDP, shared memory, etc.). These ports could connect or disconnect at run-time due to its decoupled definition of parameters. For example, a port grabber can send images to multiple listeners for parallel processing.

Another interesting feature of Yarp is that the connections between the ports have a similar structure as a directed graph, where ports are the nodes and the connections are the edges. Which means that each port is assigned with a unique name. For example, /icub/cam/left is the port created for the left eye of the iCub, as shown in Fig. 6.2. Furthermore, ports can move data from one thread to another (or several others) across different processes.

The iCub libraries are a set of open-source libraries developed by the RobotCub consortium and are publicly available\(^1\) to the wide community of iCub users. These libraries are constantly

updated to facilitate the collaboration between iCub users and to promote the developing of new algorithms like the one presented in this work. In other words, there exists a collection of libraries and modules targeting different problems, such as: vision processing, motor control, gaze control, etc.

Fig. 6.2 depicts a general overview of the implemented Yarp modules, input/output resources, applications and iCub libraries used during the development of our system. From this figure we can observe that our application, iCubActionRecognition (green rectangle), has a direct communication with the iCub, as well as with three Yarp modules (blue rectangles), i.e. CartesianController, iKinCartesianSolver and iKinGazeCtrl. Furthermore, our application communicates with the camera interfaces of the iCub, i.e. camera_resources (orange rectangle) and it uses some iCub libraries such as the iKinChain.

![Diagram showing iCub Action Recognition module](image)

**Figure 6.2.** Illustration of all the implemented applications, modules, libraries and the communication between them. We can observe that our main application called iCubAction-Recognition will infer and execute the observed activity.

We can observe that Fig. 6.2 not only depicts the name of the used resources from Yarp and iCub libraries, but also shows the created ports and the possible connections between them.
For example:

- The `CartesianController` module creates two local ports: `/primitive/right_arm` and `/primitive/torso`. The first one is connected to the iCub (remote) port `/icub/right_arm` and the second is connected to `/icub/torso` as soon as the `iCubActionRecognition` thread starts. This module commands the motion of the robot joints to the desired position.

- The `iKinCartesianSolver` creates one remote port `/icub/cartesianController/right_arm` in real-time, which will be connected with the new local port `/primitive/arm_client`. In other words, the remote port will move the end-effector of the target arm to a desired position in Cartesian space. The goal of this module is to control the right arm of the robot in the operational space, which is expressed as a combination of a 3D point to be attained by the end-effector (in this case, we considered the center of the palm as the end-effector).

- The `iKinGazeCtrl` is used to control the robot gaze through a YARP interface. This module creates a local port `/primitive/gaze_client`, which is connected with the remote port `/iKinGazeCtrl`. The remote port is created and initialized when the Cartesian client controller is launched.

- Our application also uses Yarp resources, i.e. `camera_resources`. This means that we can create a port for reading or writing images using the proper port definition. In this case with the command `BufferedPort<ImageOf<PixelBgr>> portImgIn, portImgOut`, which defines two ports with the property of sending/receiving images in BGR² format. The new defined ports will have the names `/primitive/img:i` and `/primitive/img:o`, to receive and send data respectively. The first port will receive the images from the iCub camera, i.e, this port will be connected with the port `/icub/cam/left` of the iCub. The second port (`/primitive/img:o`) will send the data to the YarpViewer (visualizer) through the port `/marker_viewer`.

- Besides the Yarp modules and resources, we use additional iCub libraries, such as the `iKinChain`. This library is a base class for defining a serial link chain between the iCub root Coordinate Frame and any limb of the iCub, for example the iCub arm (`iCubArm()`) or the iCub eye (`iCubEye()`).

The above Yarp modules, resources and iCub libraries are used inside different internal functions of the developed application, i.e. `iCubActionRecognition`. We can observe from

²BGR belongs to the RGB pixel type, with pixels stored in reverse order.
Fig. 6.2 that our application has two main functions: `inferActivity()` and `executePrimitive()`. A better description of our module which is connected with Yarp, iCub and other external libraries is explained in the next subsection.

**6.2.1. Design of iCubActionRecognition module**

The main goal of our system is to enable robots to correctly segment, recognize and reproduce the observed human behaviors from different scenarios during *running-time*. Our application is implemented within a thread from the class `yarp::os::RateThread`. This class allows to write a control loop with a specific frequency. Within this class, three main methods are called: `threadInit()`, `run()` and `threadRelease()`.

- The `threadInit()` is called once when the thread starts (before `run()` is executed). This method initializes the robot devices/interfaces and their corresponding instances that are used to configure and connect the created ports. This method is better explained in Algorithm 6.1.

<table>
<thead>
<tr>
<th>Algorithm 6.1 Implementation of the <code>threadInit()</code> function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: begin CtrlThread::threadInit()</td>
</tr>
<tr>
<td>2: icart = Open_iKinCartesianSolver() {This function opens the Cartesian Interface}</td>
</tr>
<tr>
<td>3: igaze = Open_iKinGazeCtrl() {This function opens the Gaze Interface}</td>
</tr>
<tr>
<td>4: armDevice → view(posArm) {Initializes the motor and encoder information}</td>
</tr>
<tr>
<td>5: armDevice → view(encArm)</td>
</tr>
<tr>
<td>6: eyeChain = eye → asChain() {Creates and defines a serial chain between the world CF of the robot and the desired limb}</td>
</tr>
<tr>
<td>7: armChain = arm → asChain()</td>
</tr>
<tr>
<td>8: Port_img = Open_cameraPorts() {Creates and initializes the camera ports for input/output access}</td>
</tr>
<tr>
<td>9: startMovie() {Defines the initial parameters of the video input and the output window of the created tree}</td>
</tr>
<tr>
<td>10: return The devices are valid and the needed interfaces are initialized</td>
</tr>
</tbody>
</table>

- Then, the `run()` function is called periodically every 20 ms. Within this method, we mainly call two processes, i.e. `inferActivity()` and `executePrimitive()`, as shown in Algorithm 6.2. The input/output data between these processes is depicted in Fig. 6.3 and the flow of the data within the control loop of the robot is summarized as follows:

  1. We implemented the process `inferActivity()`, which automatically segments and recognizes human activities in *real-time*. First, we display and pre-process the video stream that shows the desired human activity (Online Visual Module). The
Algorithm 6.2 The run() method within the thread class.

1: begin CtrlThread::run()
2: activity = inferActivity() \{This function segments and infers the observed human activities, see Algorithm 6.3\}
3: executePrimitive(activity) \{This function commands the desired primitive to the iCub robot, see Algorithm 6.5\}
4: return The iCub moves its limbs to achieve a similar motion as the one observed

output of this module, i.e. the state of the system $s = \{m, o_h, o_o\}$ represents the segmented high-level motions and object properties. The obtained states of the system ($s$) are used in the semantic system (Goal Inference), which are retrieved by the inferred activity ($g$).

2. Then, we developed the process executePrimitive(), which takes as input the inferred activity, i.e. the goal ($g$). This indicates that $g$ triggers the Skill Planner system, which executes the motion Primitives obtained from a Library in order to achieve a similar goal as the one observed. When the primitive is finished, then step 1 is executed again to infer the next observed activity. It is important to notice that all the modules receive inputs and produce the desired outputs on-line.

A more detailed explanation about the two main implemented processes: 1) inferActivity() and 2) executePrimitive(), within the thread is given in the next subsections.

- Finally, the thread executes threadRelease() when exiting. This implies that we stop the Cartesian, the Gaze and the motor interfaces that we have created and initialized in Algorithm 6.1, e.g. we close the Cartesian client using icart → stopControl(). Finally, we restore the controller context to its original state with the command icart → restoreContext(startup_context_id).

Thus, with the above processes we have achieved the desired behavior of the robot, i.e. the robot first observes the human, then it understands the activities performed by the human and finally, it executes the corresponding motion until the whole task is finished (the avi file stops streaming data). In the next subsections, we explain in more detail the functions and algorithms implemented to achieve the above steps on the iCub.

6.2.1.1. Description of the function inferActivity()

This process has video images of the external camera (src) as input data. This function generates as output the inferred activity. Particularly, this process automatically segments and
Figure 6.3. This figure shows the data flow between the processes through Yarp, iCub and other external libraries. We can observe the two principal applications developed: inferActivity() and executePrimitive().

infers the observed human activities. The inferActivity() application is subdivided into several sub-applications. For instance, processAViMovie(), getHumanMotions(), getObjectProp(), getActivity(), etc. These are better explained in Algorithm 6.3 and depicted in Fig. 6.4.

It is possible to observe that the implementation of the inferActivity() module, is very similar to the one described previously in Section 4.6.1, where a color-based implementation is presented to segment and track the human hands and objects from a video file. This means that this algorithm was adapted into the control loop of the robot without losing its robustness. Within this module the function getActivity() is considered as the last step of this algorithm. It represents a very important part of our implementation. This new function is described in Algorithm 6.4.
Algorithm 6.3 Implementation of the function inferActivity()

Require: 1plImage * src: avi file with the human demonstration.

1: src = cvQueryFrame(CvCapture*) {OpenCV function to read the avi file frame by frame}
2: if (src) then
3: 1plImage * src_segmented = ProcessAviMovie() {This function applies the OpenCV color base algorithm previously shown in steps 1 – 20 of Algorithm 4.2}
4: motion = getHumanMotions(src_segmented) {This step segments the human motions into two categories move or not move. This function has been previously described in Algorithm 4.3}
5: [oa, oh] = getObjProp(src_segmented) {This function gets the object properties, i.e. ObjectActedOn and ObjectInHand respectively as shown in Algorithm 6.8}
6: activity = getActivity(m, oa, oh) {This function retrieves and displays the inferred activity, see Algorithm 6.4}
7: return activity

Algorithm 6.4 Definition of getActivity() algorithm.

Require: m: human motions, e.g. move, not move or tool use
         oa : ObjectActedOn property
         oh : ObjectInHand property
         memory : memory file that contains new learned activities
1: if (m == 'not move') and (oh == 'none') then
2:  activity = Idle
3: else if (m == '?m') and (oh == '?oh') and (oa == '?oa') then
4:  activity = ?a {Replace the content of ?m, ?oh, ?oa using the corresponding information of the obtained semantic rules shown in Fig. 5.5, where ?a could be take, release, reach or put}
5: else if (m == 'tool use') and (oh == 'knife') and (oa == 'bread') then
6:  activity = Cut {Notice that for the definition of the granular activities shown in Fig. 5.6, we require the context information}
7: else
8:  newAct = find_newActivity(memory, oh, oa) {This means that a new activity has been detected, e.g. pour. Then, first we look into the memory file to find out if the rule has been already learned}
9:  if newAct == ' ' then
10:     newAct = askUser() {If the new activity is not in the memory file, then this function will display a message (during execution time) with the identified values of oh and oa, then the user will be asked to add the name of the new activity (not the rule)
11:     newRule = createNewRule(newAct, oh, oa) {The system will automatically generate the new rule}
12:     memory = saveNewActivity(memory, newRule) {The new rule will be asserted into the memory file similar to step 5 of this algorithm}
13: else
14:     activity = newAct
15: highlight_branch(activity) {Highlight the branch of the tree that corresponds to the inferred activity.}
16: return activity
It is possible to notice that steps 1 – 7 of Algorithm 6.4 are simple if – then rules that can easily been programmed in any programming language. However, if a first-order logic program is used, then the system is more robust than the typical ones Ramirez-Amaro et al. [2013b]. Steps 8 – 13 represent the possibility of incrementing new branches to our obtained tree (see Fig. 5.6). This is achieved using a memory file that can save this new detected rules.

Another important fact is that the user can decide to re-use the new learned rules when the system is used in a similar scenario. In other words, when the systems starts, i.e., during the threadInit(), a message appears that asks if the rules stored in the memory should be uploaded. The steps of Algorithm 6.3 and Algorithm 6.4 are illustrated in Fig. 6.4, where the inputs/outputs of each sub-functions are shown.

**Figure 6.4.** Main applications implemented on the iCub to infer the human activities from observations.

### 6.2.1.2. Description of the function `executePrimitive()`

Once the observed activity is inferred, the function `executePrimitive()` is called. This function defines a plan execution library that, given the inferred activity \( g \), selects the primitives \( p_n \)
from the library repertoire that need to be performed by the robot. Specifically, there is a skill plan for each activity such as reach, take, put, release, etc. This procedure is depicted in fig. 6.5 and it is explained in Algorithm 6.5. It is important to notice that when the skill plan is finished, i.e. the robot finished the execution of the inferred activity, then the robot waits until the next activity is inferred. The above is explained in step 5 of Algorithm 6.5.

**Figure 6.5.** Illustration of the applications, modules and libraries used to perform the `executePrimitive()` function.

From the execution plan, we obtain $n$—primitives ($p_n$) that the robot needs to execute. Those primitives are retrieved from the *primitive library*. For example, lets assume that the inferred activity is reach, which means that the system executes step 2 from Algorithm 6.5. The function `doReach` executes the next steps (see Algorithm 6.6):

1. **moveHead**. This represents the first primitive $p_1$ from the reaching plan, which triggers the iCub’s gaze controller [Pattacini, 2010] to randomly move the 6DOF of the iCub’s head.

2. **searchMarker**. Then, $p_2$ looks for the target ($o_1$), which is identified with a visual AR marker. This means that $p_2$, looks for the object using our implemented perception module to feedback the control system with visual information of the pose (position and
Algorithm 6.5 Description of the function \texttt{executePrimitive()}

**Require:** \texttt{activity}: the inferred activity from Algorithm 6.4.

1. if \(\texttt{activity == 'Reach'}\) then
2. \texttt{doReach()} \{Call the skill planner function of \texttt{reach}, see Algorithm 6.6\}
3. else if \(\texttt{activity == '?'a'}\) then
4. \texttt{do '?'} \{In a similar manner as step 2, the corresponding plans (\texttt{do '?')} are called depending on the inferred activity (?'a')\}
5. \texttt{activity = inferActivity()} \{When the robot finishes the primitive, then the system asks for the next activity, i.e. Algorithm 6.3 is executed again for the rest of the video\}
6. \texttt{return The robot executes the desired primitive}

Orientation) of the marker. We use the ArUco library, which is based on OpenCV to detect the markers. Every marker provides 2D image features for 4 corner points. When the marker is detected, the 3D Cartesian pose of the marker with respect to the camera Coordinate Frame (iCub right eye) which are obtained from the image features. The ArUco library uses the intrinsic parameters of the camera to compute the pose of the marker. This primitive obtains visual feedback (detection of the object), which means that this camera sensor provide information about the current state of the environment.

3. \texttt{get3Dpoint}. Then, \(p_3\) transforms the 3D Cartesian position of the marker into the iCub world Coordinate Frame (CF). This is achieved by obtaining the rotational-translation matrix from the world CF \(R_{\text{root}}^{\text{eye}}\) to the eye CF using the Denavit-Hartenberg convention explained in Section 3.5. Which means that the following equation is used:

\[
p_{\text{root}} = R_{\text{root}}^{\text{eye}}p_{\text{eye}}
\]  

(6.1)

4. \texttt{moveGaze}. Later, \(p_4\) uses the IKGaze controller to move the iCub’s head toward the desired fixation point, i.e., the 3D Cartesian point.

5. \texttt{moveArm}. Finally, \(p_5\) enables the Position-Based Visual Servoing interface of the robot to control its arm in joint space \(q\). This primitive consists of three stages. The first stage, obtains the Cartesian pose of the target using the visual marker detector described in step 2. The second stage commands the arm to move to a desired position. This position is defined using the current \(x\)– and \(y\)–positions of the hand and uses the \(z\)–position from the target. Then, when the first motion is finished, the system executes the third stage, which moves the iCub’s arm to the desired 3D Cartesian position of the target. All these motions use the Inverse Kinematic interface to transform the desired 3D Cartesian positions to desired joint positions and move the arm’s end effector to \(p_{\text{root}}\).
Algorithm 6.6 Skill planner of reaching

Require: Port_img : Images from iCub’s camera.
        igaze : gaze interface.
        icart : cartesian interface.
        eyeChain, armChain : kinematic chain from the world CF to the corresponding limb

1: 1) moveHead:
   2: igaze → lookAtAbsAngles(ang) {Move the head of the robot in a random way to locate the marker}

3: 2) searchMarker:
   4: eye_3Dpoint = marker.detect(Port_img, markerInfo) {This step retrieves the 3D Cartesian position of the identified marker using the ArUco library}

5: 3) get3Dpoint:
   6: roto_translationEye = eyeChain → getH() {get the roto-translation matrix from the world CF to the eye CF in Denavit-Hartenberg notation}
   7: world_3Dpoint = roto_translationEye * eye_3Dpoint {Obtain the 3D point in the world coordinate frame}

8: 4) moveGaze: {move the gaze to the desired 3D position}
   9: igaze → lookAtFixationPoint(world_3Dpoint)

10: 5) moveArm: {obtain the current hand Cartesian and joint position}
   11: icart → getPose(cHand, cHandJoint)
   13: icart → goToPositionSyn(cHand) {Move the right arm using inverse kinematics}

14: wait(delay)
15: icart → goToPositionSyn(world_3Dpoint) {Move the arm toward the marker position using IK.}

16: return reaching motion

When the reaching motion is finished, the function inferActivity() is executed again (see Algorithm 6.3) to retrieve the next observed human activity. This process is repeated until the information of the video is finished. In the case that a new activity is detected and learned, for instance the activity cut, then the function executePrimitive() calls the activity none, which means that the robot does nothing. This activity has been created to prevent undesired robot motions. Particularly, a primitive is executed only when that activity has been already programmed to the robot. The primitives can be programmed directly by the user or can be automatically generated using robot exchange knowledge tools as proposed in RobotEarth [Tenorth et al., 2013a]. An example that depicts the main steps of the function executePrimitive() is shown in Fig. 6.5.

Beside the pick and place activities, we have implemented another motion on the iCub, which is the pouring activity. This activity has not been explicitly included in the semantic rules and it will be identified, named and learned on-demand by the system. However, the
robot has the plan and primitives to execute this activity in advance. The skill plan implemented for this activity is explained in Algorithm 6.7. We can notice that this motion mainly uses the motor interface information, which means that we command the desired $q$ and $\dot{q}$, i.e. joint positions and velocities of the robot arm. This motion consists mainly of two steps: 1) the Pour motion, which moves the 5th joint position (a wrist joint) to a desired $q = 85^\circ$ and the value of the other joints is preserved. Then, 2) goBack, returns the arm joint position to its original values. When this last motion is finished, the pour plan function is terminated and the function inferActivity() is called.

Algorithm 6.7 Definition of the function doPour()

Require: posArm: arm motor information.
          encArm: arm encoder information.
          handVel = 10.0 : reference speed of the joints.
          1) Pour
          2: armVelocity = handVel
          3: posArm → setRefSpeeds(armVelocity.data()) {set the reference speed to 10 degrees/second}
          4: encArm → getEncoders(armEncoders.data()) {retrieve the current value of the encoders}
          5: armPosition = armEncoders {the new joint position of the arm will be the same as the current ones}
          6: armPosition[4] = 85.0 {change the value of the 5th joint of the arm to achieve the pour motion}
          7: posArm → positionMove(armPosition.data()) {move the arm to the desired joint position}
          8: 2) goBack
          10: posArm → positionMove(armPosition.data()) {move the arm to its original position}
11: return pour motion

6.3. Experimental validation on the iCub

Several experiments were performed to assess and evaluate our work on a humanoid robot in realistic scenarios. To illustrate the different contributions of this work, first we show the results of the proposed framework for the pancake making scenario. Summarizing the previous section, our system consists mainly of two subsystems, 1) activity observation and interpretation of human intentions (red highlighted square from Fig. 6.6) and 2) activity execution by the robot. Those subsystems were implemented into the control block of the iCub and the flow
of the inputs/outputs of our system is depicted in Fig. 6.6.

![Diagram](image)

**Figure 6.6.** Integration of the system into the control block of the iCub. This process includes information from external views obtained from videos and environment information obtained from the camera of the iCub. The environmental constraints are added to the desired joint position to control the robot movement.

Some examples of the results of our system implemented on the iCub are shown in Fig. 6.7. In this figure, the first row shows that the perception system will retrieve the recognized high-level motion and the object properties \(o_a\) or \(o_b\) from the pancake video. In the case of the first row of Fig. 6.7, \(motion=move\) and \(objectActedOn(o_a(t))=pancakemix\). Then, immediately the goal of the human is inferred. This indicates that one branch of the obtained tree from Fig. 5.6 is executed as shown in the first row of Fig. 6.7.b. In other words, the human activity is inferred by the system, in this case \(activity=Reach\) is sent to the robot to be executed. Then, the \(doReach()\) execution plan is performed, which is depicted in the first row of Fig. 6.7.c.

Similar to the previous example, Fig. 6.7 depicts more activities inferred and executed by the iCub in real-time, e.g. the second row from Fig. 6.7 shows the activity of **Putting something somewhere** and the final row shows the **pouring** activity, which by the way, represents a new activity to the system for this case. This means that this activity was learned on-demand. Please notice that all the implemented applications and modules of our system are running during execution time.

In order to evaluate the system’s response time for observing and inferring the human activities, we first analyze the average life time of each of the observed activities. Therefore, Table 6.1 shows the average life time of the activities in Frames\(^3\) for the three analyzed tasks, i.e.,

\(^3\)Note, that the frame information comes from the manual annotated videos made by a human expert, i.e. from
Experimental validation on the iCub

Figure 6.7. Examples of the activity recognition and execution made by the robot for the pancake making scenario. First the robot observes the motions of the human from a video, then it infers or learns the human activity and finally the iCub execute a similar activity.

sandwich making under normal condition, sandwich making under fast condition and pancake making. It is important to highlight that the duration of the videos is different for each task as well as the frequency of the videos. For example, the sandwich making under normal condition has a duration of 20 s and a frequency of 60 fps. On the other hand, the sandwich making under fast condition has a duration of 7 s and the video is a 60 fps. Finally, the pancake making task has a duration of 10 s and the video frequency is lower to 24 fps.

From Table 6.1 we observe that the shortest activity takes about 7 frames for the idle activity. In other words, the minimum number of frames that an activity lasts is approximately 7 frames. This number represents a good indicator to assure that the inferred activity has been correctly identified.

Afterward, we analyzed the duration of the observed activities in Seconds. Table 6.2 shows the ground truth data.
**Table 6.1.** Average activity life time in Frames for the *pancake making* and *sandwich making* scenarios.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Sandwich Normal</th>
<th>Sandwich Fast</th>
<th>Pancake</th>
<th>Inference time of our system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reach</td>
<td>39</td>
<td>16</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>Take</td>
<td>177</td>
<td>68</td>
<td>52</td>
<td>7</td>
</tr>
<tr>
<td>Put</td>
<td>61</td>
<td>38</td>
<td>21</td>
<td>7</td>
</tr>
<tr>
<td>Release</td>
<td>26</td>
<td>13</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>Idle</td>
<td>82</td>
<td>7</td>
<td>35</td>
<td>7</td>
</tr>
<tr>
<td>Cut</td>
<td>787</td>
<td>280</td>
<td>N/A</td>
<td>7</td>
</tr>
<tr>
<td>Pour</td>
<td>N/A</td>
<td>N/A</td>
<td>97</td>
<td>7</td>
</tr>
</tbody>
</table>

The corresponding time life of each of the analyzed activities. As expected, these activities differ in time for each task and as a result the execution speed (the frame rate of the videos) is different. Interestingly, we observe that the shortest time is 0.12 s. Therefore, the robot can make an informative decision after this time to guarantee that the observed activity has been correctly inferred and executed. Even when the robot is able to infer a new activity each frame, the robot only makes the execution of the inferred activity if this has been the same activity for the last 7 frames (i.e. 0.12 s). The reason of this waiting time is to compensate the errors coming from the perception system.

**Table 6.2.** Average activity life time in Seconds for the *pancake making* and *sandwich making* scenarios.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Sandwich Normal</th>
<th>Sandwich Fast</th>
<th>Pancake</th>
<th>Inference time of our system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reach</td>
<td>0.65</td>
<td>0.27</td>
<td>0.42</td>
<td>0.12</td>
</tr>
<tr>
<td>Take</td>
<td>2.95</td>
<td>1.13</td>
<td>2.17</td>
<td>0.12</td>
</tr>
<tr>
<td>Put</td>
<td>1.02</td>
<td>0.64</td>
<td>0.88</td>
<td>0.12</td>
</tr>
<tr>
<td>Release</td>
<td>0.44</td>
<td>0.22</td>
<td>0.63</td>
<td>0.12</td>
</tr>
<tr>
<td>Idle</td>
<td>1.37</td>
<td><strong>0.12</strong></td>
<td>1.46</td>
<td>0.12</td>
</tr>
<tr>
<td>Cut</td>
<td>13.12</td>
<td>4.66</td>
<td>N/A</td>
<td>0.12</td>
</tr>
<tr>
<td>Pour</td>
<td>N/A</td>
<td>N/A</td>
<td>4.04</td>
<td>0.12</td>
</tr>
</tbody>
</table>

When comparing the time performance of our system (0.12 s.) with the most recent state-of-the-art techniques such as OACs presented by Wächter et al. [2013], we can observe that our
approach infers the observed activity faster than previous approaches. In order to demonstrate this fact, we performed the following comparison. From the video that comes with the most recent published paper of the OACs concept [Wächter et al., 2013], we determine the time that their system needs in order to infer the observed activity. First, we observed that this time is variant since it will depend on the life time of the activity, mainly since the system needs to observe the effects/consequences of the executed activity. Therefore, the inference time of the OACs approach goes from $0.53 \text{s}$ up to $1 \text{s}$ depending on the type of activity. Whereas our system can make an inference in $0.12 \text{s}$ for every observed activity. For example in the case of the pouring activity the average life-time is around $4.04 \text{s}$, which means that the OACs method will infer this activity after $4.04 \text{s}$, whereas our system can infer this activity in $0.12 \text{s}$.

It is important to mention that in this work the observations of humans are obtained from pre-recorded videos, but our framework is not limited to this. This means that we can replace this input information with live stream videos obtained from the cameras of the robot. This is possible due to the modular architecture of our framework which allows to replace any module to acquire more complex behaviors, e.g. the vision module can be replaced for a more advance detection system or the control approach can be substituted by a more robust and adaptive control law, e.g. [Dean-Leon et al., 2004, 2006].

A video where more details for all these experimental results can be found in the following link: http://web.ics.ei.tum.de/~karinne/Videos/ramirezK_experimentalValidation-Iros14.mp4

Additionally, the implementation of recognition of parallel hand activities and the flipping the dough activities can be observed in the next link:
http://web.ics.ei.tum.de/~karinne/Videos/AR14ramirezK.avi

### 6.4. Enhancing ISHAR using egocentric views

Cognitive and neurophysiological studies have demonstrated that there is a strong coupling between the gaze and hand movements to understand human behaviors. Inspired by that fact, in this section we present the results from the recognition of human activities via semantic representations using external cameras and the object recognition from the gaze camera (see Fig. 6.8). On one hand, the gaze leads the hand toward the grasped object, the next target location of the manipulated object, landing sites, etc. On the other hand, the hand movements need to be observed using external cameras to distinguish between different hand motions as well as object information.

Mainly in this Section we present: a) the enhancement ISHAR method by combining the information from different sources using semantic representations; b) we demonstrate that
the gaze-hand coordination enhances the recognition of human activities; c) our presented framework is fully implemented in a humanoid robot that recognizes and executes the inferred activity. The above contributions are presented in more detail in the next subsections.

6.4.1. Approach to combine the external and gaze information in ISHAR

The method used to recognize human activities is an extension of the one presented in Section 5.2 based on semantic representations. This abstraction method does not directly attempt to classify human activities, but rather, it infers the activities based on the observed human motions and the information of the object of interest. To achieve this goal, we will combine information from the environment and information of human motions. First, we segment the continuous video streams from external and gaze cameras into meaningful classes using a simple Color-Based and Template matching techniques. Whereas the second part handles the difficult problem of interpreting the perceived information into meaningful classes using our inference module.

Three primitive human motions are segmented into three categories: move, not move or tool use. However, we extended the object properties to be recognized. For our previous experiments shown in Section 6.3, we used two object properties: ObjectActedOn and ObjectInHand. Now, our model contains a third property which is:

- Object Seen ($o_3$): This represents the object that the human is looking at. This informa-
tion comes from the gaze camera.

Now, the output of the current state of the system \(s\), is defined as the triplet \(s = \{ m, (o_a ; o_b), o_h \}\). The definition and some examples of the motions and object properties are similar to the ones explained in Section 5.2, with a slight modification. For example:

\[
\langle \{ \text{not}\_\text{move}, \text{None}; \text{Something}, \text{None} \}, \text{IdleMotion} \rangle
\]
\[
\langle \{ \text{move}, \text{Something}; \text{Something}, \text{None} \}, \text{Reach} \rangle
\]
\[
\langle \{ \text{not}\_\text{move}, \text{None}; \text{None}, \text{Something} \}, \text{Take} \rangle
\]

In order to obtain the hand motions (move, not move, tool use) and object properties (ObjectActedOn and ObjectInHand), we used the Color-Based technique as explained in Section 4.6. Note that the information used for these motions and properties comes from one external video. However, to obtain the new object property ObjectSeen, we implemented the Template Matching algorithm that uses the information from the gaze camera or first person view. The results from the Template matching are obtained from a joint collaboration with our colleague Dr. Humera Noor.

Object recognition from first person view offers a unique set of challenges, e.g. motion blur, hand occlusion and multiple objects (see Fig. 6.9); however it also gives an extra benefit in that the object under consideration is usually at the center of view and stands out of the other objects in the scene.

![Blurry image](image1)
![Hand occlusion](image2)
![Multiple objects](image3)

**Figure 6.9.** Top row shows examples of the problems faced with the egocentric view for the *pancake making* scenario and the bottom row depicts similar problems in the *sandwich making* scenario.
We use the simple Template Matching technique to identify the object property \textit{ObjectSeen}. We implement the function cvMatchTemplate() from the OpenCV library to match the pre-stored object models with the current input image obtained by the gaze direction.

The original source image is searched in a defined window in both \(x\) and \(y\) direction. As a result, the function retrieves the score mapping between the input image and the used template. If the matched score is above a threshold (heuristically determined), then the location and score are saved. This process is repeated over a number of model elements covering the varying viewpoints and occlusions. Thus, the objects in each frame are identified along with their coordinates.

Fig. 6.10 shows some examples of the used templates to recognize the \textit{blue knife} from the sandwich scenario. We can observe the complexity of the visual input, since several objects appear within the gaze view (see Fig. 6.9). Therefore, our system is able to recognize more than one object on the scene and the obtained results suggest that the accuracy of recognition is around 57%, when only the object with the highest score is considered.

![Fig. 6.10. This figure shows some sample model templates used to recognize the knife from the gaze camera.](image)

Regarding the \textit{pancake making} scenario, our system presents a better accuracy of 79.9% and some examples of this recognition can be observed in Fig. 6.11. We identified the object(s) seen at any instance and were able to select the single most-relevant object in each frame via thresholding. As may be observed, having a multi-image template model allows to identify the partial objects with varying viewpoints.

![Fig. 6.11. Examples of the \textit{objectSeen} recognition for the \textit{pancake making} scenario: (a) pancake Mix (b) Spatula (c) Dough.](image)

This section describes our method to combine the results obtained from the external cameras
with the gaze cameras using semantic representations. Particularly, this module interprets the visual data obtained from the perception module and process that information to infer the human intentions. This indicates that it receives as input the hand motion segmentation \((m)\) and the object properties \((o_a, o_h, o_s)\).

In order to identify and extract the meaning of human motions, we generate semantic rules to define and explain these human motions. We use a decision tree using the C4.5 algorithm [Quinlan, 1993] and the Weka software. The sandwich making scenario was chosen as the training data set, since it has a high complexity due to the several sub-activities and different constraints that it contains. First we generate a tree that can determine the human basic activities in a general manner. Then, as presented in [Ramirez-Amaro et al., 2014a] we extend the tree to recognize the granular activities based on the current context.

For the first step, we use the information of the ground-truth data of a subject during the normal condition while making a sandwich. We split the data as follows: 60% was used for training and 40% for testing. Then, we obtain the tree \(T_{sandwich}\) shown in Fig. 6.12 where the following human basic activities can be inferred: idle, take, release, reach, put something somewhere and granular\(^4\). This learning process will capture the general information between objects, motions and activities.

![Decision Tree](image)

**Figure 6.12.** This figure shows the tree obtained from the sandwich making scenario \((T_{sandwich})\). We can observe that the attributes ObjectActedOn and ObjectSeen represent the same node on the tree.

We can observe from Fig. 6.12 that the object properties ObjectActedOn and ObjectSeen belong to the same node for the found tree \(T_{sandwich}\). This indicates that there is a strong correlation between these two properties. We made further experiments to test the robustness

\(^4\)Granular activities define classes such as cut, pour, flip, etc. These activities are difficult to generalize due to the fact that they depend on the context.
of the found tree. For instance, first we remove the \textit{ObjectSeen} property from the training data set and we found a similar tree. Then, we remove the \textit{ObjectActedOn} property while the \textit{ObjectSeen} was enable and the tree was exactly the same as the one in Fig. 6.12. Even when these two properties are highly correlated, the property \textit{ObjectSeen} exhibits its own important features which are analyzed in the next section. A similar tree is obtained when the \textit{pancake making} data set is used for training as presented in [Ramirez-Amaro et al., 2013a]. This strongly indicates that the gaze information first sees the object that will be acted on (manipulate), as previously suggested in the Neurophysiological studies [Johansson and Flanagan, 2009].

From Fig. 6.12 we can observe that the activities: \textit{cut}, \textit{sprinkle}, etc. are inferred using the same rule:

\[
\text{if RightHand(Tool\_use)} \rightarrow \text{Activity(GranularActivity)}
\]  

(6.2)

Specifically, these activities need more information in order to be correctly classified. Those activities do not represent human basic activities, here we will call them \textit{granular activities}. The methodology to extend the tree to infer such activities is presented in Section 5.2.2.

6.4.2. Results of human activity recognition using gaze input

First, we test the obtained semantic rules (see Fig. 6.12) under a new scenario, in this case the \textit{pancake making} scenario. The obtained results are presented in three parts:

1. We present the inference results only using external cameras.

2. We analyze the correlation between the properties \textit{ObjectActedOn} and \textit{ObjectSeen}.

3. Finally, we present the enhanced human recognition using the combination of the external and gaze output.

First, we use the object recognition results from Section 4.6 with a small variation to correctly recognize the human activities. In this case, we try to simulate a perception module with a bad performance. In order to do this we lower the \textit{threshold\_distanceMax} from 140 to 100. We repeat this experiment for two takes using the same subject and the average of recognition accuracy is about 77\%\(^5\) without considering the gaze information. Fig. 6.13 shows the obtained confusion matrix of the first take. Notice, that these results only considered as input the information from the external cameras.

\(^5\)As we explained in Chapter 4, the quality of the perception system affects the recognition accuracy, in this case we observe that the accuracy presented before has decreased.
Figure 6.13. Confusion Matrix of the inferred activities for the pancake making scenario obtained using the information from the external cameras. The blue highlighted rectangle indicates the activity with the worst accuracy.

Then, we analyze the relationship between the object properties of ObjectActedOn and ObjectSeen. Inspired by the literature findings [Johansson and Flanagan, 2009], humans first look at the object that they intend to manipulate. Then, we tested this hypothesis in our real pancake making scenario. The results shown in Table 6.3 indicate that humans first look at the target object between 12 and 75 frames before acting on the target object. From this table we can also validate the strong correlation between the properties ObjectActedOn and ObjectSeen. For example, the object PancakeMix had the property ObjectActedOn for 11.5 frames and the property ObjectSeen for 87.5 frames. The ObjectSeen property was detected 7.5 frames before the property ObjectActedOn and the number of frames that PancakeMix had both properties was 9 frames. These results are the average of the two takes.

Finally, since we proved that ObjectActedOn can be replaced with the value of ObjectSeen, we are able to combine the information of the external cameras and the gaze camera to enhance the recognition of human activities. Then, with the inclusion of the gaze information, the recognition of human activities was improved to 82%. This is considering a perception module with low accuracy.
Assessment and integration of our framework in complex scenarios

Table 6.3. Analysis of the ObjectActedOn and ObjectSeen results (in frames) for the pancake making scenario.

<table>
<thead>
<tr>
<th>Object</th>
<th>ObjActedOn avg. length</th>
<th>ObjectSeen avg. length</th>
<th>First property detected (Nr. frames in advance)</th>
<th>Nr. frames with both properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>PancakeMix</td>
<td>11.50</td>
<td>87.5</td>
<td>ObjSeen (7.5)</td>
<td>9</td>
</tr>
<tr>
<td>Spatula</td>
<td>14.75</td>
<td>95</td>
<td>ObjSeen (78.3)</td>
<td>7.75</td>
</tr>
<tr>
<td>Dough</td>
<td>65</td>
<td>68.5</td>
<td>ObjSeen (15.5)</td>
<td>53</td>
</tr>
<tr>
<td>Pancake</td>
<td>37.5</td>
<td>153</td>
<td>ObjSeen (72.5)</td>
<td>37.5</td>
</tr>
</tbody>
</table>

It is important to highlight that the activities affected by the property ObjectActedOn are reach and release as shown in Fig. 6.12. This means that the improvement will only be reflected on those two activities and the others will remain the same. Fig. 6.14 shows the comparative results between the ground-truth and the inferred activity with and without the information of the gaze during the first 65 frames of the pancake making scenario.

![Figure 6.14](image)

Figure 6.14. Obtained results from the inference of human activities under two constraints. Top: Ground-truth data. Middle: Enhanced inference combining external and gaze cameras. Bottom: Inference using only the external input.

We can observe that the recognition using the gaze information is significantly better than without. Additionally, the obtained confusion matrix reflect this improved results, especially
for the reach activity that previously was 23.08% (see Fig. 6.13) and now with the gaze is 84.62% accurate (see Fig. 6.15).

![Confusion Matrix](image)

**Figure 6.15.** Confusion Matrix of the inferred activities the pancake making scenario obtained using the information of the external cameras and gaze. The reach and release activities are improved using the gaze information as indicated by the blue rectangle.

Interestingly, we can observe that the information of the human gaze within the semantic representation, greatly help on the prediction of the perceived human activity. Additionally, the obtained results suggest that if the external cameras fail to correctly detect the property of **ObjActedOn**, then the property of **ObjectSeen** can be used due to its strong correlation. Further analysis on this aspect is still under investigation.

An important remark regarding **ObjectSeen**. The above results seem to indicate that the importance of the **ObjectSeen** is higher than **ObjectActedOn**. Even more, we might wrongly assume that we can replace the property **ObjectActedOn** with the property **ObjectSeen**. Nevertheless this could lead to the following problems. First, the availability of the gaze sensor. It is clear that it is more common to have access to external cameras than the specialized gaze tracking system. Second, the property **ObjectSeen** switches between the **ObjectActedOn** and **ObjectInHand** as can be seen in Table 6.4. Moreover, from Table 6.3 we can notice that the number of frames that **ObjectSeen** can completely replace **ObjectActedOn** is lower than the
life-time of \( \text{ObjectActedOn} \). This analysis indicates that the property \( \text{ObjectSeen} \) can be used to enhance the accuracy of recognition of \( \text{ObjectActedOn} \) and not as its replacement.

**Table 6.4.** Analysis of the \( \text{ObjectInHand} \) and \( \text{ObjectSeen} \) results (in frames) for the *pancake making* scenario.

<table>
<thead>
<tr>
<th>Object</th>
<th>( \text{ObjectInHand avg. length} )</th>
<th>( \text{ObjectSeen avg. length} )</th>
<th>Nr. frames with both properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>35</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>PancakeMix</td>
<td>245.5</td>
<td>87.5</td>
<td>80</td>
</tr>
<tr>
<td>Spatula</td>
<td>163.25</td>
<td>95</td>
<td>30.25</td>
</tr>
</tbody>
</table>

### 6.4.3. Enhancing the reasoning and knowledge engine with gaze input

As we previously mentioned in Section 5.4.2, we introduce new capabilities in our reasoning engine. Now, we include the property \( \text{ObjectSeen} \) into our ontology. The main goal of this experiment is to use the information from the gaze images (eye view) as well as the external cameras. With this we have the first person as well as a third person views, which gives two different points of view when we learn new activities. The results presented in this work are only for the activity of *pancake making*.

First, we generate a new semantic environment model for the *pancake making* scenario. Second, we define new SQL computables and classes. For example, we can ask the following: *what object(s) do I see before I Reach my goal?*. The prolog query will be\(^6\):

\[
\text{objSeenBeforeAction}('Action', ?ObjectSeen, ?ObjGoal):-
\text{rdf_triple}(	ext{knowrob:'before', 'Action', ?ObjectSeen}),
\text{rdf_triple}(	ext{knowrob:'objectActOn', 'Action', ?ObjGoal}).
\]

where \( \text{Action} \) is replaced by *reaching*, then the output will be: \( ?\text{ObjectSeen} = \text{Spatula} \) and \( ?\text{ObjGoal} = \text{Spatula} \) or \( ?\text{ObjectSeen} = \text{Pancake} \) and \( ?\text{ObjGoal} = \text{Pancake} \) or \( ?\text{ObjectSeen} = \text{Spatula} \) and \( ?\text{ObjGoal} = \text{Pancake} \), etc. This means that in most of the cases, we first look at the object that we will reach afterward.

This represents an important contribution since this new information will help to decrease the search space for the perception module. The reason for this is that we could focus on the

\(^6\)The presented prolog queries are simplified and they are used only for illustration purposes.
object that the human is currently seeing and infer that, most probably, it will be the object that is going to be manipulated in the close future. In this case, we can consider the property \textit{ObjectSeen} as the probability that some object could be \textit{ObjectActedOn} in the near future. Further analysis on this topic is being considered as future work.

We could also infer from our system the current activity that the subject is performing when the pancake mix is being manipulated. Also, it is possible to answer if the human focuses his attention to the manipulated object or he/she is seeing something else, with the following prolog query:

\begin{verbatim}
actionObjSeen(?Action,'ObjInHand',?ObjSeen):-

rdf_triple(knowrob:'objHand',?Action,'ObjInHand'),

rdf_triple(knowrob:'detecObject',?Action,?ObjGoal).
\end{verbatim}


Therefore, we can observe that most of the time, people look at the object that he/she is manipulating, but we could also notice that during certain actions, such as pouring, a new object appear (pancake) and we most likely will focus our attention on that new object. Those inference rules, could help the robot during the planning process.

6.4.4. Transferring the obtained models into humanoid robots

As a final step, we validate our enhance framework on the same platform, the iCub humanoid robot. The implementation of our proposed framework within the control loop of the robot, follows a similar procedure as explained in Section 6.2.

The improvements of our system are presented specially when the Color-Based recognition fails to detect the \textit{ObjectActedOn} property that the Template-matching algorithm is capable to detect from the gaze video, see the last remark of Section 6.4.2. This procedure is better explained in Algorithm 6.8 and the obtained results are depicted in Fig. 6.16.

\textbf{Algorithm 6.8} Switching between ObjActOn and ObjSeen.

\begin{verbatim}
1: if (objectActOn == none) then
2:   if (objectSeen)! = ObjInHand then
3:     objectActOn = objectSeen
4: return objectActOn
\end{verbatim}

175
The important contribution of these results is the definition of rules that make it possible to infer basic human activities using different input sources, such as external and gaze cameras. Additionally, the obtained semantic representations are still valid (without any further training) under different scenarios with an accuracy above 82%. This presents the first step towards the generalization of those kinds of activities.

![Diagram of activity recognition and execution](image)

**Figure 6.16.** Examples of the activity recognition and execution made by the robot with the inclusion of the gaze information. First the robot observes the motions of the human from the external and the gaze videos, then it infers or learns the human activity and finally the iCub executes a similar activity.

This Section introduced the combination of the information from external cameras with the information from the gaze cameras. We demonstrate that the object that the human sees is typically the object that is intended to be manipulated. The object properties, i.e. *ObjActedOn, ObjInHand* and *ObjectSeen* are used to generate semantic rules that capture the meaning of the human everyday activities, e.g. *reach, take, release*, etc.

The obtained semantic rules have the important characteristic that they can be used in different scenarios and the accuracy to correctly infer human activities is above 82% when combining the external and gaze information (see Fig. 6.17). This represents our approach to find rules that could generalize basic human activities.
These figures show experiments executed on the iCub. The top part shows how the system fails to detect the activity *reach* and instead the robot detects the activity *release*, therefore the robot did not reach the targeted object. The bottom part depicts when the robot successfully detects the activity *reach* using the gaze information.

A video where more details for all these experimental results are illustrated in the following link: http://web.ics.ei.tum.de/~karinne/Videos/ramirezK_enhancedWithGaze-humanoids14.avi
6.5. Testing ISHAR in more complex scenarios

Enabling robots to learn new tasks typically requires that humans demonstrate the desire task several times [Bentinegra et al., 2006]. The observed motions should capture the human pose to further create models that identifies the demonstrated task. However, this implies high costs due to the preparations of capturing new scenarios and those observations are limited to few tasks. Nevertheless, the problem of the acquisition of data can be (partially) solved using virtual environments when new scenarios and different conditions can be rapidly tested in a larger scale for more diverse scenarios than conventional means (see Fig. 6.18).

Figure 6.18. Overview of our approach using as input virtual environment signals.

Advancements in Virtual Reality have enabled well-defined, accurate and consistent virtual environments that can capture even sequences of complex scenarios, such as human everyday activities. Another advantage of the virtual simulators (such as SIGVerse\(^7\)) is the fact that they are designed to be a user-friendly mechanism between virtual robots/agents and real users, allowing a better interaction. We envision that such rich scenarios can be used to train robots to learn new behaviors and improve the human-robot interaction, specially in human everyday activities where a diverse variability can be found.

\(^7\)http://www.sigverse.org
The construction of hypothetical interaction models of humans should be designed to be large and preferably based on big data to make it more scalable and to achieve a more natural Human Robot-Interaction (HRI). The research on HRI is done within a close laboratory, under very controlled scenarios, e.g. the light conditions are controlled, the location of the cameras, among other factors. However, that limits the exploration of more complex and difficult tasks normally analyzed in social and embodied interaction to build robust and general models about the studied interactions.

This need was explicitly stated in the Robohow\(^8\) project, where the goal is to enable robots to autonomously perform a large set of complex everyday manipulation tasks in real settings using websites, visual instructions and haptic demonstrations as primary information sources. However, integrating and combining those heterogeneous pieces is not a trivial task and it is still an unsolved problem.

In this section, we present a new framework that utilizes the SIGVerse virtual simulator to record a robot performing everyday tasks (such as, cleaning the table). Then, enable a real robot, i.e. an iCub, to process the signals from the virtual environment. This is achieved via the understanding of the activities performed by the observed robot. We show that we are able to address the challenging aspect of this work by inferring the behaviors of the new activity demonstrated by the virtual robot. This was realized using the previous knowledge and experiences that have been learned from observing human activities.

### 6.5.1. The SIGVerse simulator

Virtual environments (VE) are human-computer interfaces in which the computer creates a sensory-immersing environment that interactively responds and is controlled by the behavior of the user. For example, SIGVerse is a simulator environment, which combines dynamics, perception, and communication for synthetic approaches to investigate the genesis of social intelligence [Inamura et al., 2010].

Using this virtual simulator has several advantages, for example fast and low-costs set-up of new environments, different points of view of the scene, multi-user interaction, embodied interaction between the virtual avatars and the real user, etc. In other words, such VE are important tools specially when several human behaviors are investigated such as cooking, cleaning, etc. They provide a more complete and synchronized information about the executed task and the elements in the environment that will greatly help in the understanding of human behaviors without the need of further expensive extra sensors.

\(^8\)http://www.robohow.eu
The SocioIntelliGenesis simulator (SIGVerse) was principally developed for the RoboCup@Home simulation challenge [Inamura et al., 2010]. SIGVerse enables better and straightforward HRI experiments due to the fact that people and virtual robot agents are able to interact socially and physically. Additionally, users can arbitrary join virtual HRI experiments through the Internet to enhance the interaction. SIGVerse has three main modules:

- Dynamics are used to simulate physical properties of objects and agents.
- Percepcion provides the senses of vision, sound, force and touch to enhance the HRI.
- Communication between the available services. One very interesting feature of this system is the possibility to communicate different agents within the virtual environment but also the virtual environment can interact with users in the real world.

SIGVerse is a client/server system consisting of a Linux server and a Windows client application. The server is in charge of running the dynamics calculations and of controlling the behaviors of the robot and human avatars. Whereas the Windows client system is used to access the user interface in real time. In this work the server system was implemented to access the environment information, specially to obtain the position of the objects in the scene and the robot encoders information that can be recorded during the executed task as shown in Fig. 6.19.

For our experiment, we choose the task of cleaning up, which is one of the challenge task according to the rule-book of the RoboCup@Home competition [Inamura et al., 2013]. During this task the robot has to grasp a piece of trash targeted by the user and place it in a receptacle. Several problems are tested in this task and one of them is the understanding of the meaning of the instruction by speech recognition or by image processing of pictures captured with a camera. In this section we will focus on the second problem.

The SIGVerse simulator is a state-of-the-art system that has recently gained attention at the RoboCup@Home competition in the simulation league due to its robust functionality. During this challenge users were able to control the robot through a Joystick or a Kinect device. Specifically, the controller that determines the behavior of the robot used in this work was executed directly by random users. The information from the VE is obtained thanks to the collaboration with the National Institute of Informatics (NII) through Prof. Tetsunari Inamura.

### 6.5.2. Extracting information from SIGVerse

We extended and tested our framework to segment and recognize human behaviors from virtual scenarios. Fig. 6.18 depicts our proposed system, which first trains models to correctly
SECTION 6.5 Testing ISHAR in more complex scenarios

![Diagram of SIGVerse software components]

**Figure 6.19.** Principal components of the SIGVerse software. Additionally, we can observe the communication between the server and client services. The output log file obtained from the virtual scenario is stored in a file that is used for our semantic system (red block).

identify human behaviors while preparing a sandwich from real cameras. Then, the semantic representations and reasoning engines are obtained using the observed cooking tasks, as explained in Chapter 5. The challenging part is the transference of the obtained models into a new scenario, where instead of observing a real human, the data comes from a virtual environment and a mobile robot is demonstrating the whole new task of cleaning the table. Finally, our system is fully implemented in a robotic platform that gathers the information of the VE and extracts the semantics of the demonstrated activity to understand the behavior of the virtual robot to execute a similar task in a real scenario.

It is important to highlight that for this experiment we used the obtained semantics shown in Section 5.3. This means that for this scenario, we did not perform a training process, since the system uses the learned experiences from the previous scenarios.

One of the main advantages of using VE is the fact that the location of agents and objects within the environment are known and can be acquired without any further perception system, which safes time when analyzing the data. However, previously our system presented in Section 4.6 only considered the information of 2D images. This implies that we need to adapt our system to also include 3D data that comes from the Virtual Environment. This adaptation does not represent a problem due to the fact the modularity properties of our framework. These changes are only reflected on the segmentation of the robot hand motions during the execution.
of the task cleaning, i.e. on the computation of the right end-effector velocities of the mobile virtual robot. One important factor to mention is that for the new cleaning task, we tested the robustness of our system specially when no adjustments on the semantic rules were performed.

Since the obtained velocities of the robot end-effector presented some noise, we implement a 2nd order low-pass filter to smooth the obtained velocities. We choose the digital Butterworth filter with normalized cutoff frequency $\omega_n$.

$$H(z) = \frac{b(1) + b(2)z^{-1} + \ldots + b(n + 1)z^{-n}}{1 + a(2)z^{-1} + \ldots + a(n + 1)z^{-n}}$$\tag{6.3}

where $b$ and $a$ are row vectors that contain the filtered coefficients in length $n + 1$, where we choose $n = 2$ and with coefficients in descending powers of $z$. This filter was also used to smooth the distance signal $d(x_h, x_o)$ before computing the object properties $o_a$ and $o_h$. Some examples of the smoothed signals are shown in Fig. 6.20.

**FiguRe 6.20.** This figure shows the segmentation results from the SIGVerse signals.

Quantitatively the results of segmenting the motions (move, not move or tool use) of the virtual robot while executing the cleaning task are 77.83% accurate compare to the ground-
Regarding the recognition of the two object properties observed in this virtual scenario, the results indicate that the property of \textit{ObjectActedOn} is detected with an accuracy of about 88.46% and for the recognition of the property \textit{ObjectInHand} is 90.5%.

### 6.5.3. Results of activity recognition of a virtual robot using ISHAR

The next step consists of testing the accuracy of the obtained tree $T_{\text{sandwich}}$ from Fig. 5.6 applied to this new virtual demonstration using the VE. Notice that we did not perform any kind of training with the Virtual input. In this work, we are testing the robustness of our system when new input data is plugged-in into our framework, for example: 3D pose of the virtual robot, which involves data analysis with higher complexity.

Additionally, the agent that demonstrates the behavior is a virtual robot which is moving around the kitchen, which makes the recognition even harder, specially when no previous training has been applied to this new situation.

The obtained recognition results are shown in Fig. 6.21, where we can observe the recognition of human activities that our system obtained for the new activity \textit{clean}. From Fig. 6.21, we can notice that the activities \textit{idle} and \textit{release} get wrongly recognized by our system compared to the ground-truth.

![Figure 6.21](image)

\textbf{Figure 6.21.} Illustration of the recognition of human activities for the \textit{cleaning} task. We can observe that the \textit{cleaning} task has several sub-activities: \textit{idle}, \textit{reach}, \textit{take}, \textit{put} and \textit{release}, identified by different colors. The x-axis denotes the time numbered in frames. The y-axis first indicates the ground-truth recognition, then we present the recognition of our system in each frame and finally (up to the top) we present the recognition of our system with a 4 frames delay, which was used by the robot to execute the inferred activity.

However, if we observe the obtained tree one more time (see Fig. 5.6), then we notice that

---

\textsuperscript{9}The ground-truth data is obtained by manually segmenting the videos into hand motions, object properties and human activities.
in order to recognize the \textit{idle} or \textit{release} activities, we use the rules:

\begin{equation}
\text{if \ Hand(NotMove) and ObjectInHand(None)} \\
\quad \rightarrow \text{Activity(Idle)} \tag{6.4}
\end{equation}

\begin{equation}
\text{if \ Hand(Move) and ObjectInHand(None)} \\
\quad \text{and ObjectActedOn(None)} \rightarrow \text{Activity(Release)} \tag{6.5}
\end{equation}

We can notice that in both activities the only difference is the motion of the hand, which is either \textit{move} or \textit{not move}. This is a very interesting phenomenon since it suggest that these two rules describe the same activity. This aspect did not pop up before since previously we have tested our system in different cooking scenarios, where the human is mostly standing in the same place, which is in front of a table. However, in this new scenario, we notice that the demonstrator, in this case the virtual robot, is moving around the kitchen, which stressed this situation.

Then, we considered the case when these two rules describe the same activity and we tested the \textit{sandwich making} scenario again with this new assumption and the recognition improved from 92.57\% to 95.43\%. Thus, demonstrating that these two rules are equivalent. Following a similar procedure, we computed the accuracy of recognition of our system into the new scenario and the quantitative results are 80\% accurate compare to the ground-truth, which is a very high accuracy specially since no training was performed in this new scenario and the train and test scenarios are very different. This means that from Fig. 6.21 the black and magenta sections represent the same activity. Even more, this evidence was also detected in the labeling process performed by random users where they labeled the \textit{idle} and \textit{release} as the same activity.

\section*{6.5.4. Transferring the acquired models into humanoid robots}

As a final step, we validate our framework on the same robotic set-up following a similar procedure as explained in Section 6.2. However, we needed to include new procedures to adapt our code to the new scenario where more than one object is detected. Which means that several objects can have the same property at the same time. Then, to avoid that case, we improved our system specially during the recognition of \textit{ObjectActedOn} and \textit{ObjectInHand} properties. This procedure is better explained in Algorithm 6.9.

The results of the recognition of the human activities can be observed in Fig. 6.22. Our
Algorithm 6.9: Determine \( \text{ObjActOn} \) and \( \text{ObjInHand} \).

**Require:**
- \( \text{distance}[\text{n}] \): store the distance between the hand and objects detected (n).
- \( \text{threshold}_{\text{OA}} \): \( \text{ObjActedOn} \) threshold.
- \( \text{threshold}_{\text{OH}} \): \( \text{ObjInHand} \) threshold.
- \( \text{threshold}_{\text{OA2}} \): \( \text{ObjActedOn} \) threshold for the second object

1: for \( i = 0 \) to \( N \) step 1 do
2: if \( \text{distance}[i] \leq \text{threshold}_{\text{OH}} \) then
3: \( \text{OH\_vector.push\_back}(i) \)
4: else
5: if \( \text{distance}[i] \leq \text{threshold}_{\text{OA}} \) then
6: \( \text{OA\_vector.push\_back}(i) \)
7: Find the value with lower distance from \( \text{OA\_vector} \) to choose \( \text{ObjActOn} \)
8: for \( \text{index} = 0 \) to \( \text{OA\_vector.size()} \) step 1 do
9: if \( \text{distance}[\text{OA\_vector}[\text{index}]] < \text{min} \) then
10: \( \text{ObjActOn} = \text{OA\_vector}[\text{index}] \)
11: \( \text{min} = \text{distance}[\text{OA\_vector}[\text{index}]] \)
12: A similar procedure is followed to find \( \text{ObjInHand} \), where we can define the motion of tool use
13: for \( \text{index} = 0 \) to \( \text{OH\_vector.size()} \) step 1 do
14: if \( \text{distance}[\text{OH\_vector}[\text{index}]] < \text{min} \) then
15: \( \text{ObjInHand} = \text{OH\_vector}[\text{index}] \)
16: \( \text{min} = \text{distance}[\text{OH\_vector}[\text{index}]] \)
17: else
18: if \( \text{distance}[\text{OH\_vector}[\text{index}]] < \text{threshold}_{\text{OA2}} \) then
19: \( \text{ObjActedOn} = \text{OH\_vector}[\text{index}] \)
20: \( \text{motion} = \text{tool\_use} \) {This procedure defines the motion of tool use for granular activities}
21: return \( \text{ObjActOn}, \text{ObjInHand}, \text{motion} \)

results also suggest that in order to execute a complex task such as cleaning the table it is possible to only recognize simple and basic human activities, such as: reach, take, put, release/idle. In other words, in this context we did not use the motion value of tool use, which also proves the robustness of our semantic representations.

In this section, we present our framework to bootstrap humanoid robot skills using virtual reality means via extraction of semantic representations for the recognition of human activities. Our semantic representations are obtained by segmenting low-level human motions, i.e, move, not move or tool use and two object properties, i.e. ObjActedOn and ObjInHand. We prove that our semantic rules captures the meaning of the human everyday activities, in a completely new scenario, i.e. cleaning the table without any further training with an accuracy of recognition around 80% [Ramirez-Amaro et al., 2014b].
6.6. Summary and contributions of this chapter

The experimental integration and validation of the acquired cognitive behavior into a humanoid robot are very important, essential and a challenging task. This integration provides robots with capabilities of activity recognition via understanding. Specifically, this allows robots to cooperate with humans in a more natural and robust manner. This represents another key factor of our framework since we integrate the perception and semantic reasoning capabilities to a humanoid robot, in this case the iCub. In other words, our work is not limited to a theoretical domain, but rather, to provide a functional system capable to interact in real sce-
Summary and contributions of this chapter

This chapter addresses the integration of high-level control (decision making modules) with low-level control (motion control) to generate a functional system. The challenge is significant, as it requires implementation of interfaces between high-level control and low-level control. We integrate and assess our system in multiple scenarios with different levels of complexity, i.e., we tested under multiple input sensor information and under complex and new scenarios as follows:

- **One External video**: first, we integrate the perception obtained from Chapter 4 with our semantic and reasoning system presented in Chapter 5 into the control loop of the robot. Since the above integration is done in the robotic control loop, some challenges were considered during the integration. For example, once the robot has correctly inferred the human activity, we have to consider the on-line camera input of the robot, motor commands and other sensors of the robot to imitate and replicate the inferred behavior, etc. The results shown in Section 6.3 suggest that the robot is capable to understand new observed human activity in 0.12 s with an accuracy of 85%, without any further training.

- **External video and Gaze input**: then, we extend our semantic and reasoning engine by including the gaze information as observable information. The results shown in Section 6.4 indicate that the semantic representations are enhanced using the gaze information, specially when the perception module of the external cameras fails to correctly recognize the object property $\text{ObjectActedOn}$. The improvement of our system is reflected specially in the recognition of the activities $\text{reach}$ and $\text{release}$, where the accuracy went from 23.08% to 84% in the case of the recognition of the activity $\text{reach}$.

- **Virtual reality input**: finally, we tested our system in an even more complex scenario using different input representation, i.e. 3D signals obtained from a Virtual Environment scenario. In this experiment, we did not perform any training for the new data to test the robustness of our system. Then, the obtained results presented in Section 6.5 show that our framework is able to extract from the SIGVerse virtual simulator the meaning of the observed motions with 80% accuracy of recognition by obtaining the objects relationships given the current context via semantic representations to extract high-level understanding of those complex activities even when they represent different behaviors.

This indicates that our proposed system is designed in a way that accepts different inputs without further modifications and without the need of further training to correctly extract the semantic of complex human activities.
Transferring the learned models toward new scenarios or for different input sources was not possible using the current methods. Then, the need for a framework capable to integrate different observations obtained from different sensors is fundamental and very important to autonomous robots. In this thesis, we presented a modular system that builds up state-of-the-art techniques to correctly recognize human activities from different sources of observation and for different scenarios implemented in a humanoid robot.

First Section 7.1 presents a summary of the previously presented modules that conform our system. This section is divided in several subsections, starting with Section 7.1.1, where we provide conclusions about the key components of our system; then, Section 7.1.2 presents some final remarks regarding the enhancement of our system by including new input sources; after that, in Section 7.1.3 we explain the robustness of our system when more complex input sources and scenarios are tested. Section 7.1.4 states the obtained overall achievements of this thesis. Finally, Section 7.2 explains the outlook of this work.
7.1. Summary and contributions

Segmenting, recognizing and understanding human activities from observations typically requires a very complex and sophisticated perception algorithms [Poppe, 2010; Le et al., 2011; Patterson et al., 2005] or motion capture systems [Azad et al., 2004; Wächter et al., 2013], or virtual environments [Inamura et al., 2010] or other sources. However, such systems would be unlikely implemented on-line into a physical system, e.g. a robot, due to the pre-processing step(s) that those vision systems usually demand. This illustrates the complexity of the problem tackled in this thesis.

For example, if the robot learned human everyday activities from a sandwich making scenario, then the robot should be able to extract the semantics of the observed activities in order to recognize these learned models into a new scenario. The robot should be also capable to handle demonstrations that are acquired from different sensory inputs. Moreover, it should be flexible and adaptable enough to re-use the learned models in multiple scenarios with different levels of complexity, e.g. the preparation of a pancake, cleaning a table, etc. Fig. 7.1 illustrates this ideal robotic system. One of the main conclusions that we draw from this work is that the meaning of the human behaviors can be extracted and expressed using relationships between human motions and the observed objects. In this work, we use this conclusion as a definition for semantics of human behaviors.

![Learning and Testing](image)

**Figure 7.1.** This figure shows examples of the transference of experiences between scenarios achieved in this work. Two cases are depicted: a) shows the learning stage for the automatic segmentation and recognition of human activities; and b) shows that even when a different scenario is observed the semantics of the observed activity remains the same.

In this work we proposed a new hierarchical recognition method for the activities of human behaviors into atomic units, which we called primitive motions. Then, we associate these primitives with general information of the object in the scene to enhance the recognition of
very complex dynamic human behaviors. These primitives can be acquired from several sensory input data robustly and on-line. Consequently, enabling the system to be fast, reliably, adaptable and intuitive, which are key components for intelligent autonomous robots when learning new complex tasks during everyday situations.

Hence, with our proposed hierarchical approach such complex human behaviors are possible to recognize. Our method consists of two levels of abstraction:

1. The low-level abstraction, which describes the primitive motions, such as: move, not move or tool use, which can be observed under different scenarios and their definition will remain the same through different scenarios. This low-level also includes two object properties: ObjectActedOn and ObjectInHand, which represent the minimal information required from the environment to infer more complex activities.

2. The high-level abstraction, which represents the complex human activities, i.e. the activities that we wish to automatically segment and recognize from the observations, such as: reach, take, release, pour, cut, etc. These high-level activities are inferred using the low-level information via semantic representations.

These two levels of abstraction have several advantages compared to other techniques as shown in Table 2.1. For example, we demonstrate that our system performs very accurately (around 85%) even when new activities are tested; thus demonstrating that the inferred representations do not depend on the performed task. Furthermore, the system is able to recognize new activities and learn the correct rule(s) on-line, which means that we do not need to include all possible activities to the system, since this is not feasible in real applications.

Moreover, our framework enables robots with the capabilities of on-line segmentation, recognition, understanding and execution of the observed human behaviors under realistic scenarios. Further advantages of the central components of our system are presented in the next subsection.

7.1.1. Key elements addressed in this work

Our system can be utilized for the difficult and challenging problem of tasks and skills transfer from any agent to humanoid robots by extracting semantic representations of the observed behaviors. Specifically, this abstraction will capture the meaning of the observed human activities, which represents a challenging and difficult problem, specially for complex activities such as: reach, take, cut, release, etc. We experimentally validate our framework on a real humanoid robot, an iCub. Additionally, we are able to reuse different constraints to validate the obtained semantic rules under different conditions, such as:
• Four different and complex kitchen activities: 1) making a pancake; 2) making a sandwich; 3) setting the table; and 4) cleaning the table. Notice, that we use the information of the sandwich making to train our model and the rest of the activities were used for testing.

• Different sources of the observations, such as: one external camera, three external cameras, the combination of the external cameras and the egocentric views, as well as information from Virtual Environments.

• Several execution styles for similar activities, for example in the case of the sandwich making scenario the people performing the activities had different time constraints. As a consequence the execution of the activities has high variability.

### 7.1.1.1. Importance of the new data sets

Beside the research contributions, during this work we recorded two new data sets from natural and challenging scenarios, i.e. making pancakes and making sandwiches. Our new data sets contain information of 5 cameras. The first three cameras capture the human motions from the 3rd. view perspective, while the last two cameras record the self- or 1st. view perspective using wearable devices. These new recordings allow us to test our system with realistic scenarios using as inputs: one external camera [Ramirez-Amaro et al., 2014a], three external cameras [Ramirez-Amaro et al., 2013b], gaze camera [Ramirez-Amaro et al., 2013a], combination of the above 6.4.

Our suggested scenarios have several goal-directed movements that can be observed, which could be typically perceived in such natural environments. Such aspects can not be observed in control environments. Our data sets, help us to explore different input sources at the same time and for the same activities, For example, Fig. 6.8 shows that during the pouring activity, the gaze is fixated in the dough while the human is grasping the pancake mix. One fundamental question that inspired such recordings was to find out if the information of the gaze enhances the recognition of human activities as demonstrated in the Neuroscience studies [Flanagan and Johansson, 2003] but instead of using controlled environments, we used noisy and real environments.

### 7.1.2. Enhancing ISHAR with egocentric views

In this work, we evaluated our proposed method using the combined information of external cameras together with gaze cameras to enable robots to robustly recognize and understand hu-
man everyday activities (see Fig. 6.8). For this case we also used our new data sets explained in Section 3.4. Our findings, shown in Section 6.4, suggest that the combination of these two visual sources enhanced the recognition of human behaviors. Finally, our findings were implemented on a humanoid robot, which is able to recognize human activities with better accuracy, for example the activity reach increase from 23% (only using external cameras) to 84% by including the gaze information.

Then, we added new capabilities into the reasoning engine, which makes possible to compute new relationships between objects manipulated, objects seen and activities. This improved the dynamic growing of the knowledge base in a meaningful manner, which is required since we can not guarantee that the knowledge that has been manually stored in the system will be valid under different constraints/scenarios.

### 7.1.3. Assessment of ISHAR using complex and new scenarios

We have tested our system under different conditions, for instance using one static camera as presented in [Ramirez-Amaro et al., 2014a] or several external cameras, e.g. [Ramirez-Amaro et al., 2013b] and by combining external and egocentric views [Ramirez-Amaro et al., 2013a] to enhance the recognition of activities of daily life for robotic systems. However, those recordings are limited to the analysis of the specific scenarios where they were recorded. Then, the acquisition of a new tasks (new scenario) will require whole new set-ups, recruiting participants that will demonstrate the new tasks, accurate sensors located around the scenario, etc., which in long term represents a very costly and limited solution. Therefore, a better alternative is the use of virtual simulators that allow a long-term Human-Robot-Interaction due to its large scale capabilities as proposed by [Inamura et al., 2013].

Then, we integrate the information from such Virtual Environment into our framework. The obtained results show that our framework was able to extract from the SIGVerse virtual simulator the meaning of the demonstrated motions with 80% accuracy of recognition. This was successfully done by obtaining relationships between object properties and motions via semantic representations. This allows to extract high-level understanding of those complex activities even when they represent different behaviors.

In summary, the main contributions of this section are: a) we proposed a framework that combines real and virtual data using semantic representations; b) we proved that the learned models, obtained using 2D images, are valid even when we use as input the 3D information of the virtual scenario; c) we shown that our system is not limited to standing tasks but it is also valid for tasks when the demonstrator is moving around its environment; d) our presented framework is fully implemented in a humanoid robot that obtains information from either
cameras or VE and transfer its learned semantics to recognize and execute the inferred activity.

### 7.1.4. Contributions and overall achievements

The acquired Semantic Representations can be used to effectively learn in an intuitive and dynamic manner new skills. Hence, this method can be used flexibly across different demonstrators, situations and constraints – to infer and achieve a similar goal than the one observed. Furthermore, the inference capability introduced in this work has been integrated in different control schemes for a humanoid robot. These control approaches are able to take into account information from external sources and the sensors built in the robot to trigger on-line transitions between the inferred activities and the robot executions. Then, the robot exhibits similar behaviors as the demonstrator.

One fundamental contribution of our framework is the extraction of abstract representations of the observed task, which represents a big advantage of our framework compare with other approaches, e.g. [Ikeuchi and Suchiro, 1992; Park and Aggarwal, 2004], due to the fact that the current approaches learn models of the task for a specific scenario or a robot. Then, the obtained model only contains the information for that specific task, where the generalization is not possible.

Some other approaches strongly depend on pre- and post-conditions of the perceived demonstration to recognize human activities e.g. [Wörgötter et al., 2009; Jäkel et al., 2010; Aksoy et al., 2011; Wächter et al., 2013]. However, even when these systems are accurate, they need to observe the whole activity (from beginning to end) to correctly recognize it and typically a very sophisticated perception system is needed. Whereas, our system works with a very naive perception system and it can infer the demonstrated human activity each frame.

Our framework overcomes such methods by including Semantic Representations, Reasoning Engines and Knowledge-Based systems, which enhance the system with the ability to deal with partial information of the perceived environment. This is possible since the ontology representation exhibits a hierarchical structure of the knowledge base, where the inferred instances inherit the properties and relations of the parental class. In other words, it is possible to infer new properties between instances that were not manually provided. This represents a big advantage since our proposed system is more adaptable and flexible than classical approaches [Turaga et al., 2008; Park and Aggarwal, 2004].

In summary, our system has made the following achievements:

- We designed a modular framework that allows a better adaptability of the learned models to new environments by considering multi-level sensory input (see Chapter 3). This
greatly helps on the generalization of human activity understanding toward new situations.

- We assessed the state-of-the-art perception systems and the results showed in Chapter 4 suggested that using our proposed levels of abstraction it is possible to split the recognition complexity of such systems. Specifically, we enhanced the perception of human motions from 25% up to 72% using the same data sets and the state-of-the-art ISA technique without and with our proposed framework respectively.

- We proposed a new method to generate semantic rules in order to capture the essence of the human activities. The results obtained from Chapter 5 indicate that the best recognition accuracy obtained by our system is 92%. Whereas the worst recognition is obtained by a random human labeling the same demonstrations with accuracy of 74.62%. This means that our framework is able to extract higher/abstract level representations from the demonstrator.

- Our system has been integrated in a humanoid robot as presented in Chapter 6. The obtained results demonstrate that our robot is capable to infer new observed human activities in 0.12 s with an accuracy of 85%, without any further training. Moreover, the robot is able to include new sensory input data such as egocentric views where the recognition accuracy went from 23.08% to 84% in the case of the activity reach. Moreover, the robot without any further training was capable to identify the demonstrations of a virtual robot with an accuracy of 80%. Thus demonstrating the robustness of our proposed framework.

The proposed methodology to generate and extract semantic rules from human observations is presented and tested under different constraints. This implies that new activities were able to be learned and identified on-demand from new scenarios in an intuitive manner. This is achieved due to the fact that we proposed to recognize human behaviors via their understanding. Therefore, the learned models are reliable, generalizable, fast and scalable. A comparison between our proposed system and the state-of-the-art techniques in the area of Semantic Representations can be found in Table 7.1.
Table 7.1. This table summarizes the advantages of our system compared with the analyzed related work on human activity recognition.

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Accuracy</th>
<th>Re-usability of learned models</th>
<th>On-line recognition</th>
<th>Multiple input sensors</th>
<th>Naive perception system</th>
<th>Fast recognition</th>
<th>Implemented in a robot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affordances [Jäkel et al., 2010]</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>OACs [Wörgötter et al., 2009; Wächter et al., 2013]</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>SEC [Aksoy et al., 2011]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Linguistic approaches [Park and Aggarwal, 2004; Aggarwal and Ryoo, 2011; Inamura and Shibata, 2008]</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓ ✓</td>
</tr>
<tr>
<td>ISHAR (Our system)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

7.2. Outlook

Correctly identify human activities from observations is a challenging task in the robotics community, and its solution is very important since it is the first step toward a more natural human-robot interaction. In this work we presented a methodology to extract the meaning of the human basic activities by combining the information of the hand motion and two object properties. This information is used as input to our framework that can handle multiple sources of input information as well as several new scenarios. Then, the output of our system is the inferred human activity that is executed by a robotic system.

Testing algorithms into new scenarios has several challenges, such as: recognition of new behaviors, re-usability of models, accuracy, etc. Our system is able to handle most of those
Conclusions

These figures exemplify the predefined paths that humans follow while preparing pancakes and sandwich scenarios.

challenges. However one limitation is the recognition of a new activity, which is first identified as granular. Thus, as future work we are analyzing mechanisms to learn on-demand new behaviors. Our first attempt to solve this issue is with the inclusion of memory and an active-learning module as presented in [Ramirez-Amaro et al., 2014a]. However, the inclusion of the knowledge-based ontology proposed in this work will also help to control the growing of the decision tree, this is still under investigation.

We noticed that the performed activities follow predefined paths (see Fig. 7.2). These paths could be obtained from the semantic representation that we propose in this work. The obtained graphical structures shown in Fig. 7.2 are a strong indicator of the different styles that humans follow to perform the desired task. Such graphs could also help for detecting errors on the recognition of human activities. For example, if first our system infers the activity flip and in the next frame it infers the activity release that could indicate that either the object in the hand has been dropped or that the tracker system failed to detect the hand motion. Therefore, the obtained semantic rules could enhance the planning process of robotic systems, the error detection and possibly the error recovery. These three topics represent key elements for autonomous robots of the future.
APPENDIX A

Optimization principles to transfer human-like motions to robots

The robotic motions implemented in Chapter 6, were performed using the Inverse Kinematic algorithm. However, that technique is not very robust, therefore as part of this thesis we, in conjunction with our partners from the mathematics department, analyzed a mapping between the human motions vs. the robot motions, i.e we explore the embodiment problem, where the mapping between the human and robot motions is not straight forward. To tackle this problem we use physically inspired optimization principles to imitate human reaching motions.

In this section we are going to present an end-to-end framework which equips robots with the capability to perform reaching motions in a natural human-like fashion. First, a markerless, high-accuracy, model-based human motion tracker is used to observe how humans perform everyday activities in real-world scenarios (see Fig. 4.7 from Sec. 4.3). Then, the obtained trajectories are annotated and clustered to represent different types of manipulation and reaching motions occurring in a kitchen environment. Afterward, we use bilevel optimization methods to build models based on a combination of physically inspired optimization principles such as the minimization of jerk to determine which principle describes the human motions best. Finally, the obtained models are used directly to generate reaching motion trajectories for humanoids like the iCub which are both similar to human behavior and respecting the individual requirements of the robotic hardware. The pipeline of this framework is depicted in Fig A.

Imitation is an important learning mechanism, since it allows humans to first show the robot the motions that he/she needs to do in order to achieve a certain task and that means that the robot by mimicking that motion is defining the joint angles that it needs to do to follow the teacher’s motions. These motions are known as motion primitives Schaal [1999]. A motion primitive is a sequence of actions that can accomplish a certain movement goal.
primitives can be formalized in the form of control policies, where a control policy \( u \) is a function that maps the state \( x \) of the movement system and its environment into an appropriate action \( a \) for a particular task on time \( t \), i.e. \( u = p(x, t, a) \). Therefore we can store in a library the motion primitives produced by the robot from imitating the human motions and a combination of those primitives will define a complex task.

The human will demonstrate the task in terms of the desired motion and the robot will try to imitate the performed motions by using the joint angles that will be obtained from the human tracker system. Since the length of links of the humanoid robot and the human are different, the robot will not achieve the same goal as the teacher, because the task workspace of robot and human is different, that is the reason why learning is necessary and very important in this project. Initially the robot will follow the exact motion that the human did in order to perform the desired task, when the motion is finished then the robot will analyze if the goal-motion
was achieved via visual information. On the other hand, if the goal was not achieved, then the robot should continue with the next possible movement by trying at the same time to continue with the posture as similar as possible to the demonstrated posture of the teacher. Using this method, the robot should learn how to manipulate a stochastic, dynamic environment within a few trials.

Our contribution is a framework which is able to represent and generate motions based on an optimal combination of physically inspired principles obtained by analyzing human motions recorded in an unconstrained observation setup. In the experiments discussed here, subjects are observed while performing the everyday activity of setting the table without prior instructions. The underlying principles of the human motions are determined by a bilevel optimization approach, i.e. we identify parameters in an optimal control problem describing the motion planning, and a human-like control is generated for a humanoid robot based on these identified parameters. This approach has the important property to generate motions that fit different kinematic/dynamic chains while also considering variations in the task space goal. The generated motions are similar to human behavior as they are based on the same optimization principles. This is an important requirement in human-robot interaction scenarios where humans are expecting a certain behavior from the robot.

As an example of everyday movements we focused on reaching motions, which consist in moving one hand from an initial position in the workspace to the desired goal position. The presented framework is the combination of several procedures. The starting point are video sequences of humans performing everyday manipulation tasks. A markerless human motion tracker is used to extract the posture of the human in each frame and then this data set is sorted into the characteristic human sub-movements by a clustering approach based on the algorithms of Dynamic Time Warping and k-mean, these steps are better describe in Chapter 4.3. Given a characteristic movement, a novel bilevel optimization approach yields the one out of a given family of cost functions which reproduces the data best. For the specific dynamics of the humanoid robot’s arm an optimal trajectory for this cost function is computed (see Figure A).

A.1. Using optimization principle analysis

A common assumption of many approaches analyzing human motion is that humans try to minimize an unknown cost function while doing everyday manipulation tasks. This assumption also forms the basis for the strategy discussed here. It has to be noticed that the main purpose of this approach is to describe human behavior based on physical models rather than to explain it from a biological or psychological perspective [Engelbrecht, 2001]. The opti-
Optimization principles to transfer human-like motions to robots

Optimization principles implemented in this section were mainly developed by our partners of the Mathematics department. Therefore, the details of such analysis are beyond the scope of this thesis document, rather can be found in the following work [Albrecht et al., 2011].

The dynamics of the human arm are modeled by connecting rigid body dynamics of the links with a simple dynamical model of the muscle characteristics. The arm model possesses two joints (shoulder and elbow) having five degrees of freedom and seven lumped muscle pairs altogether. The resulting ordinary differential equation is discretized by using a single-step method to obtain the constraints of the type \( h(x) = 0 \).

The goal of our approach is now to find the weight vector \( w \) minimizing the deviations between recorded human data and the quantities corresponding to the solution of the lower level problem (llp). We choose to compare the two by their hand positions in Cartesian space. Consequently, a function \( p^{comp} \) is introduced which returns the hand positions belonging to the llp solution \( x^*(w) \) and \( p^{data} \) are the recorded human hand positions. The applied distance measure dist uses points of equal paths length. Summing up, the following optimization problem is obtained, the so-called upper level problem (ulp):

\[
\min_w \text{dist}(p^{data}, p^{comp}(x^*(w)))^2
\]  

(Figure A.2. Comparison of tracked wrist and elbow positions (black/grey) and trajectories computed from the optimal combination of cost functions (blue) (units in meter) [projection to the x-z-plane])
subject to
\[ \sum_i w_i = 1 \text{ and } x^*(w) \text{ solves l}

In the bilevel optimization of the three stereotypic arm movements recorded in the kitchen we use the generalizations to three dimensions of some common cost functions: Minimization of hand jerk [Flash and Hogan, 1985], joint jerk [Rosenbaum et al., 1995] and torque change [Uno et al., 1989] and additionally, we consider variants like planar hand jerk based only on the jerk of the x- and y-component.

Here the optimization results for three motions (reaching for a cup in an upper kitchen unit, reaching for a placemat on top of the oven and picking up a spoon inside a drawer) are presented. The recorded trajectories of human wrist and elbow positions and the corresponding optimal results of the bilevel optimization are displayed in Figure A.2. It can be observed that it is possible to find weight vectors \( w \) such that the computed hand paths get close to the recorded ones.

The optimization results show that no single criterion alone can correctly reproduce the trajectories. Figure A.3 shows the root of the values of the upper level cost function, which is the root mean squared deviation (RMSD) between the recorded data and the computed trajectory, for some of the used basic cost functions and the optimal combination. Thus, demonstrating that the optimal combination of the underlying physical principles yield significantly better trajectories than single principles alone (see Figure A.3).

![RMSD between observed and optimal trajectories for some basic l

As the RMSD values already suggest, the dominant cost function in the optimal combination for the reaching motion to the cup is torque change where for motion to the placemat joint jerk dominates. In the third case, i.e. the motion to the spoon, a combination of joint jerk with hand jerk and planar hand jerk is optimal.
A.2. **Experimental validation of the human-like motions on the iCub**

We use the iCub [Metta et al., 2008], a 53 degrees of freedom humanoid robot, to transfer and execute the human motions. The strong humanoid design of the iCub gives us an appropriate testing platform to show similarities between the original human motions and the human-based optimized robot motions. Strong anatomical human-robot similarities can be appreciated on the shoulder and elbow joints, which are precisely the joints we are concentrating on.

Using the cost function found by the bilevel optimization for a characteristic movement in the kitchen scenario, the kinematics, dynamics, and joint limits of the iCub, and selectable initial and final Cartesian positions, an optimal control problem similar to the llp mentioned above is solved to generate the controls for the iCub. The obtained motion is therefore human-like with respect to the optimization principles used by the human for that specific action.
(human to robot optimization transfer). Additionally it is also adjustable to any desired initial state and final goal within the workspace of the robot. Therefore the robot is able to execute the same type of intentional action but with the flexibility of choosing an initial state and final goal to fit the current situation.

For evaluation purposes in our experiments we chose final and initial states for the robot similar to the ones from the human but scaled down in order to fit inside the workspace of the robot. Execution time was also scaled to fulfill the velocity restrictions of the robot.

It is important to note that our approach does not copy human movements directly but generates trajectories for the robot’s arm based on a combination of dynamic principles used by the human. Additionally, in our case more restrictive hardware constraints in the kinematic chain of the robot are met (Figure A.4). The generated optimal motions are angle trajectories that are fed to the robot at regular intervals and executed using an independent P controller 

\[ \dot{q} = k_p(q_{ref} - q) \]

for each joint. This controller introduces a delay and the robot itself has a motor velocity controller that introduces other types of transformations to the signals.

Figure A.5 shows the human and the iCub robot executing the reaching motion for the placemat from a similar viewpoint. The elbow of the robot has to guarantee a certain distance to the torso in order to avoid self-collision. This limits the maximal similarity between robot and human motion, but on the other hand it shows that the difficulties arising in the context of transferring control strategies can be handled by our approach.
A.3. Summary of this appendix

This section discusses a framework to record and analyze human arm motions in an environment of everyday-life, in our case a kitchen scenario, and then to use this knowledge to control a humanoid in a human-like way. Various steps are needed within this framework: Starting with tracking the human motions and then clustering into relevant sub-movements, data is assembled which clearly shows the stereotypical human motions. A bilevel optimization approach is then used to analyze this data and obtain a cost function describing the human arm movement best, considering the nonlinear dynamics of the human arm. In the last step, this cost function is used to generate optimal trajectories for a humanoid robot doing the same task as the human did before.

The main contribution of our approach is to use physically inspired cost functions, which optimally describe observed human arm motions, to control a robot not by copying the observed trajectories of the human, but by minimizing the same cost function for the robot while maintaining the ability to change the initial position and the final goal.

Due to the fact that the robot control is based on optimizing a cost function that is likely to be meaningful and important for the execution of the current activity from the intention perspective, this system has the potential to be connected in a convenient way to high level reasoning and planning. This high level system could select the appropriate cost function parameters to fit the current desired intention of the action.
Bibliography


Bandouch, J., Engstler, F., Beetz, M., 2008a. Accurate Human Motion Capture Using an Ergonomics-Based Anthropometric Human Model, in: Fifth International Conference on Articulated Motion and Deformable Objects.


209


Inamura, T., Tan, J.T.C., Sugiura, K., Nagai, T., Okada, H., 2013. Development of RoboCup@Home Simulation towards Long-term Large Scale HRI.


217


Bibliography


