ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

IDENTIFICATION OF OBJECT MANIPULATION ANOMALIES FOR SERVICE ROBOTS

Ph.D. THESIS

Doğan ALTAN

Department of Computer Engineering

Computer Engineering Programme

OCTOBER 2021

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

IDENTIFICATION OF OBJECT MANIPULATION ANOMALIES FOR SERVICE ROBOTS

Ph.D. THESIS

Doğan ALTAN
(504142504)

Department of Computer Engineering

Computer Engineering Programme

Thesis Advisor: Assoc. Prof. Dr. Sanem SARIEL UZER

OCTOBER 2021

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

SERVİS ROBOTLARI İÇİN NESNE ETKİLEŞİM ANOMALİLERİNİN TANISI

DOKTORA TEZİ

Doğan ALTAN
(504142504)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Sanem SARIEL UZER

EKİM 2021

Doğan ALTAN, a Ph.D. student of ITU Graduate School student ID 504142504 successfully defended the thesis entitled "IDENTIFICATION OF OBJECT MANIPULATION ANOMALIES FOR SERVICE ROBOTS", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Sanem SARIEL UZER** Istanbul Technical University	....
Jury Members :	**Assoc. Prof. Dr. Gökhan İNCE** Istanbul Technical University	....
	Assoc. Prof. Dr. Sinan KALKAN Middle East Technical University	....
	Prof. Dr. Nadia ERDOĞAN Istanbul Technical University	....
	Prof. Dr. Işıl BOZMA Boğaziçi University	....

Date of Submission : **4 October 2021**
Date of Defense : **27 October 2021**

__Anneme ve babama,__

FOREWORD

First of all, I would like to express my deep appreciation to Dr. Sanem SARIEL UZER for not only being such an inspiring advisor but also a great mentor. During my graduate studies, she has always been there with her enlightening ideas and suggestions on my research. She has also provided moral support when I felt lost. It is a great honor for me to be her first Ph.D. graduate.

I would like to thank my thesis committee members, Dr. Gökhan İNCE and Dr. Sinan KALKAN for providing me the opportunity to improve this research with their constructive comments and suggestions. I would also like to especially express my thankfulness to Dr. H. Levent AKIN, who had to leave the membership of my thesis committee due to circumstances beyond control. His invaluable suggestions and comments on this research helped me improve this study. Finally, I would like to thank the defense exam members, Dr. Nadia ERDOĞAN and Dr. Işıl BOZMA, for forming the final version of this thesis booklet with their helpful comments.

I have conducted this research in a friendly environment, the Artificial Intelligence and Robotics Laboratory (AIR Lab) of Istanbul Technical University (ITU). I had the privilege to collaborate with very bright researchers, Arda İNCEOĞLU, Cihan AK, Çağatay KOÇ, Mustafa ERSEN and Türker ÜNLÜ. I want to thank them all for their support and collaboration. I will always be proud of being a member of the AIR Lab.

During my graduate studies, I worked as a research assistant in the Faculty of Computer and Informatics Engineering of ITU. I want to thank all faculty members for providing a friendly working environment. Especially, I would like to thank Dr. Sema OKTUĞ for her support as the dean of the faculty. I would also like to mention Dr. Tolga OVATMAN and Dr. Berk CANBERK as they have always been good mentors.

This thesis was supported by ITU Scientific Research Projects (ITU-BAP) under Grant 40240 and by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant 115E-368. I want to thank ITU-BAP and TUBITAK.

During this tough journey, I have been surrounded by amazing people who have always motivated me. I am very fortunate to have Atakan ARAL, Nilay ARAL and Melodi Deniz ÖZTÜRK as intimate friends. Although Atakan and Nilay live in a different country than I live in, they never made me feel the distance as they were always ready for an audio or video conference when I needed their support. I cannot express my gratitude to Deniz for always providing mental support, especially at the pessimistic and hopeless moments. Deniz and Atakan also helped me a lot with the proofread of my publications and they provided suggestions for further improvements. I want to thank them for easing this journey with their support.

My parents have always provided me unconditional support with every decision of mine, including being a researcher. My mother, Nevze, always motivated me with her endless love and patience. My father, Ethem, always encouraged me to achieve my ambitions. Thus, I dedicate this thesis to them.

October 2021 Doğan ALTAN

TABLE OF CONTENTS

Page

FOREWORD ix
TABLE OF CONTENTS xi
ABBREVIATIONS xv
SYMBOLS xvii
LIST OF TABLES xix
LIST OF FIGURES xxi
SUMMARY xxiii
ÖZET xxv
1. INTRODUCTION 1
1.1 Terminology 3
1.1.1 Anomaly vs. Failure 3
1.1.2 Detection, identification and recovery 4
1.2 Proposed Solution 6
1.3 Contribution 7
1.3.1 Symbol-based anomaly identification framework 7
1.3.2 Convolutional three-stream anomaly identification (CLUE-AI) framework 8
1.3.3 Multimodal analysis 8
1.3.4 Symbolic-level features vs. convolutional features 8
1.4 Organization of the Thesis 8
2. LITERATURE REVIEW 11
2.1 Anomaly Detection 11
2.2 Anomaly Identification 14
2.2.1 Logic-based methods 14
2.2.2 Clustering-based methods 14
2.2.3 Probabilistic methods 15
2.2.4 Deep learning methods 16
2.2.5 Other methods 16
2.3 Recovery 17
3. PROBLEM DEFINITION 21
3.1 Problem Formulation 21
3.2 Scenarios and Potential Anomaly Cases 23
3.2.1 Investigated manipulation actions 24
3.2.2 Scenarios and potential anomaly cases 25
3.2.2.1 Location change anomaly (LOC) 26
3.2.2.2 Object disappearance anomaly (DIS) 27
3.2.2.3 Earlier unstable action anomaly (EUA) 27

3.2.2.4 Overturning anomaly (OTA) 28
3.2.2.5 Spilled object anomaly (SPC) 28
3.2.2.6 Full container anomaly (FCA) 29
4. BACKGROUND 33
4.1 Artificial Neural Networks (ANNs) 33
4.1.1 Recurrent neural networks (RNNs) 34
4.1.2 Long short-term memories (LSTMs) 34
4.1.3 Gated recurrent units (GRUs) 35
4.1.4 Attention 35
4.1.4.1 Self-attention 35
4.2 Graphical Models 36
4.2.1 Hidden Markov models (HMMs) 36
4.2.2 Conditional random fields (CRFs) 37
5. SYMBOL-BASED ANOMALY IDENTIFICATION FRAMEWORK 41
5.1 Low-level Processes to Detect Anomalies 41
5.1.1 Audition 41
5.1.2 Vision 42
5.1.3 Proprioception 45
5.2 Symbols as Features for Anomaly Identification 46
5.3 Methods for Anomaly Identification 48
5.3.1 The LSTM-based anomaly identification algorithm 48
5.3.1.1 Early multimodal fusion 52
5.3.1.2 Majority voting 54
5.3.2 The HMM-based anomaly identification algorithm 54
5.3.3 The CRF-based anomaly identification algorithm 55
5.4 Experiments 56
5.4.1 Experimental setup 57
5.4.2 Data collection and annotation 57
5.4.3 A temporal analysis of an anomaly case 58
5.4.4 Performance evaluation 60
5.4.4.1 State classification analysis of the observation sequences 61
5.4.4.2 Overall identification analysis of the observation sequences 66
5.4.4.3 Sensory modality analysis 67
5.5 Concluding Remarks 69
6. CONVOLUTIONAL THREE-STREAM ANOMALY IDENTIFICATION FRAMEWORK 71
6.1 Convolutional Three-stream Anomaly Identification (CLUE-AI) Framework 71
6.1.1 Visual stream 71
6.1.2 Auditory stream 73
6.1.3 Proprioceptive stream 75
6.1.4 Late fusion 75
6.2 Experiments 76
6.2.1 Experimental setup 77
6.2.2 Experimental evaluation 77

6.2.2.1 Training analysis 78
6.2.2.2 Feature extraction analysis 78
6.2.2.3 Overall performance analysis 82
6.2.2.4 Multimodality analysis 84
6.2.2.5 Performance analysis on noisy data 88
6.2.2.6 Transformer analysis 89
6.2.2.7 Analysis of audio processing 90
6.3 Concluding Remarks 90
7. CONCLUSIONS AND DISCUSSIONS 93
7.1 Discussion 94
7.2 Limitations and Future Work 95
7.3 Impact 96
REFERENCES 97
CURRICULUM VITAE 108

ABBREVIATIONS

AP	: Average pooling
ANN	: Artificial neural network
AR	: Autoregressive
AROW	: Adaptive regularization of weight vector
CAM	: Class activation map
CLUE-AI	: Convolutional three-stream anomaly identification
CNN	: Convolutional neural network
CRF	: Conditional random field
DIS	: Object disappearance anomaly
EUA	: Earlier unstable action anomaly
ECS	: Entity-Component-System
FCA	: Full container anomaly
GRU	: Gated recurrent unit
Grad-CAM	: Gradient-weighted class activation mapping
HMM	: Hidden Markov model
HDP	: Hierarchical Dirichlet process
KB	: Knowledge base
L-BFGS	: Limited-memory Broyden-Fletcher-Goldfarb-Shanno
LCD	: Liquid crystal display
LLM	: Locally linear models
LOC	: Location change anomaly
LSTM	: Long short-term memory
MFCC	: Mel-frequency cepstral coefficients
MTL	: Metric temporal logic
MEM	: Model error modeling
MLP	: Multilayer perceptron
OWA	: Open-world assumption
OTA	: Overturning anomaly
ReLU	: Rectified linear unit
RGB-D	: Red green blue-depth
RNN	: Recurrent neural network
SOM	: Self organizing maps
SPC	: Spilled object anomaly
SVM	: Support vector machine
TAL	: Temporal action logic
UAV	: Unmanned aircraft systems

SYMBOLS

$\boldsymbol{A}$	: Anomaly set
$\boldsymbol{a}_{z_i z_j}$	: Transition model
$\boldsymbol{anomaly_i}$	: Anomaly type i
$\boldsymbol{b}$	: Bias
$\boldsymbol{b}_{z_i}(\boldsymbol{x})$	: Observation model
$\boldsymbol{C}$	: Selected input variables for LSTMs
$\boldsymbol{d_k}$	: Size of the key
$\boldsymbol{\eta}$	: Learning rate
$\boldsymbol{H}$	: Observation history
$\boldsymbol{f}$	: Feature function
$\boldsymbol{f^i}$	: Features for a specific modality i
$\boldsymbol{F}$	: Activation function
$\boldsymbol{o}_t^{s_m}$	: Observation gathered through a specific sensor s_m
$\boldsymbol{r^i}$	: Output vector for a specific modality i
$\boldsymbol{s_i}$	: Sensor i
$\boldsymbol{S}$	: Sensor set
$\boldsymbol{M}$	: Anomaly model
$\boldsymbol{\sigma}$	: Sigmoid function
$\boldsymbol{x}$	: Observation
$\odot$	: Element-wise multiplication
$\boldsymbol{z}$	: Hidden state
$\boldsymbol{Z}$	: Hidden state sequence
$\boldsymbol{w}$	: Weights for neural networks
$\boldsymbol{W}$	: Weights for LSTM hidden states
$\boldsymbol{K}$	: Key
$\boldsymbol{Q}$	: Query
$\boldsymbol{\Pi_i}$	: Initial state distribution
$\boldsymbol{\lambda}$	: Feature function weight
$\boldsymbol{V}$	: Value
$\boldsymbol{U}$	: Weights for LSTM inputs
$\boldsymbol{ws}$	: Window size

LIST OF TABLES

Page

Table 3.1 : Dictionary of the anomalies that are handled [1]. **31**
Table 5.1 : Overall performance evaluation and comparison of the anomaly identification algorithms. **63**
Table 5.2 : Performance analysis of the LSTM-based identification algorithm with different sensor modalities. **69**
Table 6.1 : Number of scenarios for each anomaly type. **77**
Table 6.2 : Parameter dictionary of the LSTM structure. **77**
Table 6.3 : Number of epochs that each model is trained and the number of parameters (in millions) that each vision model includes. **79**
Table 6.4 : Performance evaluation of different feature extraction methods. **82**
Table 6.5 : Time elapsed (sec) during training and testing with different feature extraction techniques. **83**
Table 6.6 : Comparative performance analysis of different methods. **83**
Table 6.7 : Time elapsed (sec) during training and testing of different deep learning methods. **84**
Table 6.8 : Sensory modality and attention analysis. **85**
Table 6.9 : Analysis of the CLUE-AI framework with transformer layers. **90**

LIST OF FIGURES

Page

Figure 1.1 : An anomaly case where a 3-block tower collapses due to an earlier unstable action execution [1]. **3**
Figure 1.2 : Illustration of the anomaly and failure definitions.... **4**
Figure 1.3 : An example scenario illustrating the workflow of corresponding modules. **5**
Figure 3.1 : Baxter robot with its selected sensors and actuators. **24**
Figure 3.2 : The objects that are used in the anomaly scenarios.... **25**
Figure 3.3 : Snapshots of the anomaly scenario that includes location change anomaly (LOC). **26**
Figure 3.4 : Snapshots of the anomaly scenario that includes object disappearance anomaly (DIS). **27**
Figure 3.5 : Snapshots of the anomaly scenario that includes a fall-down of a tower due to an earlier unstable action (EUA).... **27**
Figure 3.6 : Snapshots of the anomaly scenario that includes cases where an object falls while it is pushed by the robot along a given axis (OTA). **28**
Figure 3.7 : Snapshots of the anomaly scenario that includes cases where a container overflows (SPC). **29**
Figure 3.8 : Snapshots of the anomaly scenario that includes cases where an object is placed into an already full container (FCA). **29**
Figure 4.1 : A single neuron with its inputs and output. **34**
Figure 4.2 : A binary-state HMM with three observations [2].... **37**
Figure 4.3 : An example sequence for an HMM for the block stacking scenario. **37**
Figure 4.4 : An example sequence with feature functions for the block-stacking scenario.... **38**
Figure 5.1 : Spectrogram (top) and waveform (bottom) analyses of three drop events of distinct objects. **43**
Figure 5.2 : (a) Illustration of the KB of the robot taken from rviz on an intermediate step of a tower construction scenario. (b) Corresponding image of the rviz illustration taken from the RGB-D camera.... **45**
Figure 5.3 : The processed data by the symbol-based anomaly identification framework [1]. **47**
Figure 5.4 : An illustration of the proposed symbol-based anomaly identification framework [1]. **50**
Figure 5.5 : The early fusion scheme for auditory data ($ws = 1$). **54**
Figure 5.6 : A sample LOC anomaly case scenario. A human changes the location of the object be manipulated. [1]. **58**

Figure 5.7 : Sensor values, their interpretation, and the illustration of the robot's belief on each anomaly type during the change in the object's location. **59**
Figure 5.8 : Analysis of the change in loss (blue) and f-score (green) during the training phase of the LSTM-based method. A learning rate (η) of 10^{-3} is set. **62**
Figure 5.9 : Normalized confusion matrices of the methods for the state classification task. **65**
Figure 5.10 : Normalized confusion matrices of the methods for the overall case classification task. **68**
Figure 6.1 : The architecture for processing visual stream. **74**
Figure 6.2 : The architecture for processing auditory stream. **75**
Figure 6.3 : The architecture for processing proprioceptive stream. **76**
Figure 6.4 : Proposed late fusion scheme of the CLUE-AI framework. **76**
Figure 6.5 : Change on the train and test loss for $\eta = 10^{-4}$ for the VGG16-based setting. **78**
Figure 6.6 : ResNet18 as a feature extractor. **80**
Figure 6.7 : ResNet101 as a feature extractor. **80**
Figure 6.8 : AlexNet as a feature extractor. **81**
Figure 6.9 : VGG16 as a feature extractor. **81**
Figure 6.10 : CAM of a SAFE case. **86**
Figure 6.11 : CAM of an FCA case. **86**
Figure 6.12 : CAM of an OTA case. **87**
Figure 6.13 : CAM of a EUA case. **87**
Figure 6.14 : CAM of an SPC case. **88**
Figure 6.15 : CAM of a DIS case. **88**
Figure 6.16 : CAM of a LOC case. **89**
Figure 6.17 : Performance analysis of the CLUE-AI framework on noisy data. **89**
Figure 6.18 : Performance analysis of the CLUE-AI framework with different shaped kernels. **91**

IDENTIFICATION OF OBJECT MANIPULATION ANOMALIES FOR SERVICE ROBOTS

SUMMARY

Recent advancements in artificial intelligence have resulted in an increase in the use of service robots in many domains. These domains include households, schools and factories to facilitate daily life in domestic tasks. Characteristics of such domains necessitate the intense interaction of robots with humans. These interactions necessitate extending the abilities of service robots to deal with safety and ethical issues. Since service robots are usually assigned to complex tasks, unexpected deviations of task state are highly probable. These deviations are called anomalies, and they need to be continually monitored and handled for robust execution. After an anomaly case is detected, it should be identified for effective recovery. For the identification task, a time series analysis of onboard sensor readings is needed since some anomaly indicators are observed long before the detection of the anomaly. These sensor readings need to be fused effectively for correct interpretations as they are generally taken asynchronously.

In this thesis, the anomaly identification problem of everyday object manipulation scenarios is addressed. The problem is handled from two perspectives by considering the feature types that are processed. Two frameworks are investigated: the first one takes into account domain symbols as features while the second framework considers convolutional features.

Chapter 5 presents the first framework to address this problem by analyzing symbols as features. It combines and fuses auditory, visual and proprioceptive sensory modalities with an early fusion method. Before they are fused, a visual modeling system generates visual predicates and provides them as inputs to the framework. Auditory data are fed into a support vector machine (SVM) based classifier to obtain distinct sound classes. Then, these data are fused and processed within a deep learning architecture. The architecture consists of an early fusion scheme, a long short-term memory (LSTM) block, a dense layer and a majority voting scheme. After the extracted features are fed into the designed architecture, the occurred anomaly is classified.

Chapter 6 presents a convolutional three-stream anomaly identification (CLUE-AI) architecture that fuses visual, auditory and proprioceptive sensory modalities. Visual convolutional features are extracted with convolutional neural networks (CNNs) from raw 2D images gathered through an RGB-D camera. These visual features are then fed into an LSTM block with a self-attention mechanism. After attention values for each image in the gathered sequence are calculated, a dense layer outputs the attention-enabled results for the corresponding sequence. Mel frequency cepstral coefficients (MFCC) features are extracted from the auditory data gathered through a microphone in the auditory stage. This is followed by feeding these auditory features into a CNN block. The position of the gripper and the force applied by it are also fed

into a designed CNN block. These resulting sensory modalities are then concatenated with a late fusion mechanism. Afterward, the resulting feature vector is fed into fully connected layers. Finally, the anomaly type is revealed.

The experiments are conducted on real-world everyday object manipulation scenarios performed by a Baxter robot equipped with an RGB-D head camera on top and a microphone placed on the torso. Various investigations including comparative performance evaluations, parameter and multimodality analyses are studied to show the validity of the frameworks. The results indicate that the presented frameworks have the ability to identify anomalies with f-scores of 92% and 94%, respectively. As these results indicate, the CLUE-AI framework outperforms the other in classifying occurred anomaly types. Due to the requirements that the frameworks necessitate, the CLUE-AI framework does not require additional external modules such as a scene interpreter or a sound classifier as the other one does and provides better results compared to the symbol-based solution.

SERVİS ROBOTLARI İÇİN
NESNE ETKİLEŞİM ANOMALİLERİNİN TANISI

ÖZET

Yapay zeka alanındaki son gelişmelerle, servis (hizmet) robotları gündelik hayatın içinde yer almaya başlamışlardır. Özellikle ev, okul ve fabrika gibi ortamlarda hizmet sunmaktadırlar. Bu ortamlardaki işlerin doğası gereği, robotların ortamda bulunan insanlarla ve diğer başka robotlarla etkileşimde olmaları gerekmektedir. Bu etkileşimler nedeniyle, etik konuların gündeme alınması ve robotların güvenli çalışmaya yönelik gerekli olan bilişsel yeteneklere sahip olmaları gerekmektedir. Bu tez kapsamında bu amaca yönelik çalışmalar yürütülmüştür.

Servis robotları çoğunlukla birden fazla eylemi içeren karmaşık görevler yürüttüğünden, görev yürütmesi sırasında beklenmeyen durumlar ile karşılaşılması kaçınılmazdır. Bu tip beklenmeyen durumlar anomali olarak isimlendirilir. Anomaliler robotun kendi yürüttüğü eylemlerin bir sonucu olarak ortaya çıkabileceği gibi, içerisinde bulunduğu ortamda bulunan başka robotlar veya insanlar nedeniyle de ortaya çıkabilir. Anomali durumlarının ortamda bulunan insanlar, diğer robotlar veya robotun kendisi için tehdit oluşturacak herhangi bir duruma yol açmaması güvenli görev yürütme açısından önemli bir kriterdir. Bu nedenle, bilişsel bir robotun öncelikle ortamını devamlı bir şekilde gözlemleyerek potansiyel anomali durumlarını sezmesi gerekmektedir. Anomalinin sezilmesinden sonra, robotun etkin bir şekilde anomali kotarımı yapabilmesi için meydana gelen bu anomalinin tipini belirleyebilmesi oldukça önem taşımaktadır. Bu sebeple, bilişsel bir robotun bu tip anomali durumlarını tespit edip çözümlemesi için bir anomali tanısı yordamına ihtiyacı vardır. Bu tezde, robotun hedeflerine ulaşabilmesi için görev yürütmesi sırasında karşılaşabileceği potansiyel anomali durumlarının sınıflandırması problemi ele alınmıştır. Bu problemde, robotun üzerinde taşıdığı sensörlerinden (görüntü, ses, kuvvet vb.) gelen verileri yorumlayarak anomali tanımını yapabilmesi ele alınır. Bu sensör verilerinin her biri, doğaları gereği farklı karakteristiklere sahip olabilirler. Bu nedenle robot, ortamını mevcut sensörleri ile algıladıktan sonra, mevcut gözlemlerini bir arada değerlendirerek etkin bir şekilde işleyebilmeli ve anomaliler üzerine çeşitli çıkarım algoritmaları yürütebilmelidir. Fakat, robotun anomali durumlarını tam olarak analiz edebilmesi için yürütmeye ilişkin önceki gözlemlerini de göz önüne alması gerekir. Örneğin, bazen anomaliler önceki adımlarda yürütülen eylemler nedeniyle oluşabilir. Tüm bu zorluklarla başa çıkabilmek için anomali tanısı probleminin zamansal bir boyutta analiz edilmesi gerekir. Bu analiz ile, potansiyel anomali durumları ve sınıfları arasında zamansal ilişkilendirmelerin yapılması gereklidir.

Bu tezde, servis robotları için anomali tanısı problemi, özniteliklerin işlenmesi açısından farklı seviyelerde ele alınmış ve iki farklı derin öğrenme tabanlı çok kipli (multimodal) anomali tanısı mimarisi önerilmiştir. İlk olarak sunulan mimaride, öznitelikler sembolik seviyede oluşturulmuş ve işlenmiştir. Bu mimaride robot, sahip olduğu kamerasından gelen iki boyutlu görüntü verisini, mikrofonundan gelen ses

verisini ve tutucusundan gelen uygulanan kuvvet bilgisini işlemektedir. Bu bilgiler bir araya getirilmeden önce, yüksek seviyeli sembollerin çıkartılması gerekmektedir. İki boyutlu görüntülerden görsel sembollerin çıkartılması işini yerine getirmek amacıyla bir sahne modelleme sistemi kullanılmaktadır. Bu sayede robot, sahneye ilişkin bilgilere (nesne bilgileri, uzamsal ilişkiler vb.) sahip olmaktadır. Daha sonra, mikrofondan alınan ses verisi, bir ön işleme aşamasından geçmekte ve veriye ait öznitelikler çıkartılmaktadır. Bu aşamayı, ilgili özniteliklerden yüksek seviyeli sese ilişkin sembolik yüklemlerin destek vektör makineleri (Support vector machines, SVM) kullanılarak çıkartılması izler. Tutucudan alınan işlenmemiş bilginin de (kuvvet, tutucu açıklı bilgisi vb.) sembollere dönüştürülmesi ile işlenmemiş sensör verisinden yüksek seviyeli sembolik verinin elde edilmesi işlemi tamamlanmış olur. Robotun sensörlerinden gelen veriler her zaman senkron olmayabilir ve bu durum elde edilen gözlemlerin zamansal olarak bir araya getirilmesinde zorluklara yol açabilir. Bu nedenle önerilen yöntem, sensor bilgisinin bir araya getirilmesinde zamansal bir erken füzyon yöntemini benimser. Bu aşamadan sonra, farklı sensör kiplerine (modality) ait verilerin elde edilme zamanları tutarlı olacak şekilde bir araya getirilmiş olur. Daha sonra, anomali semptomları bir derin öğrenme mimarisinde öğrenilir. Bu çalışmada, uzun kısa süreli bellekler (Long short-term memories, LSTM) kullanılarak anomali semptomlarının bir modeli öğrenilmiştir. Yürütme anında elde edilen ardışık gözlemler, bu model kullanılarak etiketlendikten sonra bir çoğunluk oylaması yöntemi ile anomalinin nedenine karar verilir.

Tez kapsamında ele alınan ikinci mimari, evrişimli (convolutional) özniteliklerin analiz edilerek anomali tanısının yapılması fikrini baz alır. Geliştirilen mimari, görsel algı (vision), işitsel algı (audition) ve iç algı (proprioception) özniteliklerin işlendiği üç aşamaya sahiptir. Her bir aşamada, farklı sensör kiplerine ait verilerin işlenmesi görevi yerine getirilir. Bu mimaride, bir önceki mimariden farklı olarak, görsel özniteliklerin çıkartılması için herhangi bir sahne yorumlama sistemine gerek duyulmaz. Aksine, ilgili öznitelikler doğrudan iki boyutlu görüntüler üzerinden evrişimli sinir ağları (Convolutional neural networks, CNN) kullanılarak çıkartılır. Çıkartılan görsel öznitelikler, uzun-kısa süreli bellekler ve bir odak (attention) mekanizması aracılığı ile anomali semptomlarının öğrenilmesinde kullanılır. Bu aşamanın tamamlanması ile, görsel verinin işlenmesi aşaması sona ermiş olur. Diğer aşamada ise işitsel verinin işlenmesi görevi yürütülür. Robotun yürütme anında mikrofonu aracılığı ile aldığı işitsel veriden, ilk olarak Mel frekansı kepstrum katsayıları (Mel frequency cepstral coefficients, MFCC) öznitelikleri çıkartılır. Daha sonra çıkartılan bu öznitelikler, bir evrişimli sinir ağları yapısından geçirilerek, işitsel veri işleme görevinin sonuna gelinir. Üçüncü sensör verisi olarak tutucuya ilişkin tutucunun açıklık bilgisi ve tutucu tarafından uygulanan kuvvet bilgisi bu veriyi işlemek üzere tasarlanan bir evrişimli sinir ağı yardımıyla işlenir. Mimaride son aşama olarak bir geç füzyon tekniği kullanılarak görsel algı, işitsel algı ve iç algı bileşenleri bir araya getirilir. Füzyon ile birleştirilmiş bu bilgi, bir sinir ağı yardımıyla işlenerek oluşan anomalinin sınıfı belirlenir.

Önerilen mimariler, Baxter robotu üzerinde çeşitli anomali durumlarını içeren gündelik nesne etkileşim senaryolarında test edilmiştir. Deneyler kapsamında, karşılaştırmalı performans analizleri, parametre analizleri ve kullanılan sensör kiplerinin anomali tanısına etkilerinin analizleri irdelenmiştir. Deney sonuçlarında, iki mimarinin de anomali durumlarını başarıyla (sırasıyla %92 ve %94 f-skorları ile) belirleyebildikleri görülmüştür. Elde edilen sonuçlara göre, gereksinimler göz önüne alındığında ilk mimarinin sahne yorumlayıcı ve sembolik seviyede ses sınıflandırıcı gibi, bir robot mimarisinde her zaman mevcut olamayabilecek modüllere gereksinim duyduğu durumu ortaya çıkmıştır. Aynı zamanda, özellikle bir anomali durumunda görsel olarak sahnenin karmaşık bir hal almasının, sahnenin tam ve doğru bir şekilde yorumlanmasında sorunlara yol açtığı ve bu nedenle anomali tanıma performansını etkilediği görülmüştür. Bu sonuçlar doğrultusunda, evrişimli özniteliklerin göz önüne alınarak ayrı aşamalarda incelendiği mimarinin daha genel bir çözüm sunmasından ve daha iyi performans sağlamasından dolayı, daha tercih edilir olduğu gösterilmiştir.

Tez kapsamında ele alınan problem ve sunulan çözüm, robotların yeteneklerinin arttırılması doğrultusunda önemli bir adımdır. Aynı zamanda elde edilen sonuçlar, güvenli görev yürütmesi ve etik açısından önemli olup, umut vericidir.

1. INTRODUCTION

The rapid spread of robot use in daily life indicates that service robots will help ease many domestic tasks in the near future. Even as of today, many successful applications of service robots in everyday tasks, including cooking, cleaning and shopping, have been reported in the literature. Several open problems, however, still remain and are under investigation for these tasks [3, 4]. The remaining open problems mainly surround the cases where close interactions with objects and humans are required, which tends to add an additional layer of challenge. Such close interactions, and robots being in the same environment with humans, have revealed the importance of safety and ethical concerns to be addressed by the researchers as well [5, 6].

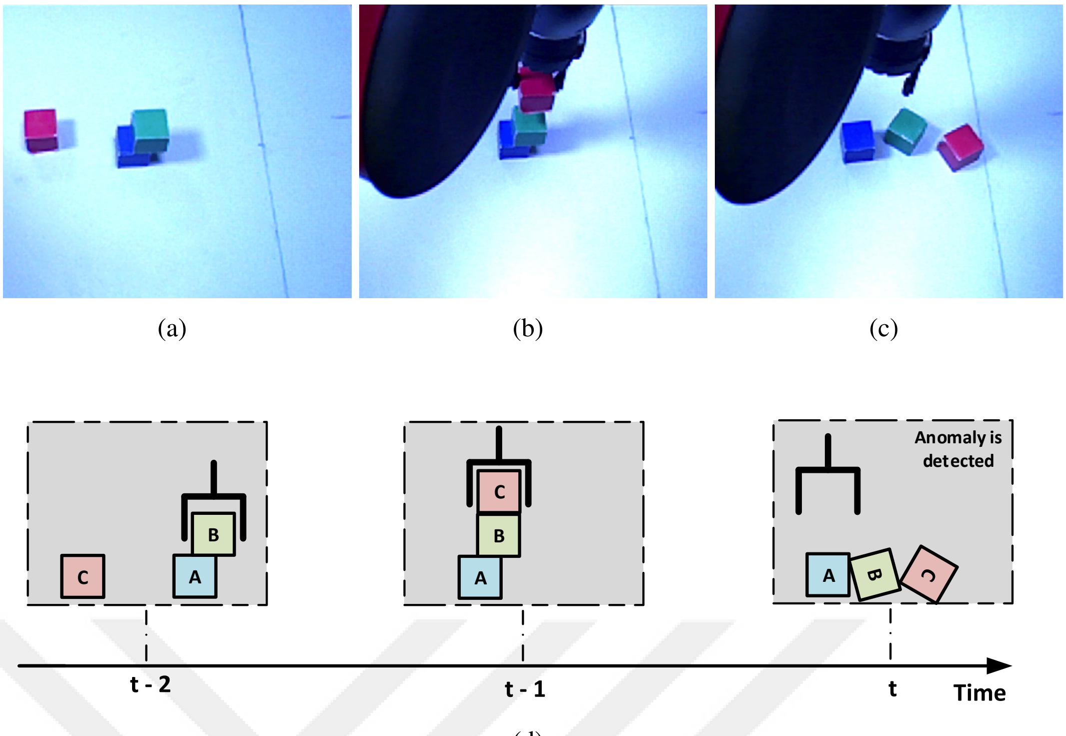
Robots are prone to encounter unexpected situations (i.e., anomalies) while achieving objectives in their environments, especially when a human factor is present. Such anomaly situations may threaten the safety of humans, the environment, or even the robots themselves. Therefore, safety in execution is considered one of the primary concerns for cognitive robots [7]. This concern needs to be taken into account while designing both hardware and software of the robots. Complete avoidance of such anomalies may be impossible, and as such, it is essential for the robots to be capable of dealing with these deviations to ensure autonomy, in addition to robustness for any upcoming tasks. It is also crucial for them to recognize their limits related to what they cannot fulfill. As a consequence, diagnostic procedures that contribute to the aforementioned skills are required for service robots in domestic environments. Such automated procedures are essential components of robot software design since they provide the opportunity for the robots to handle anomalies autonomously without any human intervention. Consequently, enhancing the abilities of service robots to cope with such safety and ethics problems is crucial for expanding their capabilities in achieving everyday manipulation tasks. Learning is one possible way of achieving such enhancements. Recent advancements in learning (i.e., from demonstration) have made it possible for robots to augment their abilities [8].

Diagnostic procedures are possible by sensing the environment. Onboard diagnosis is more appreciated for autonomy independent from external sensors. A combination of different onboard sensing modalities is beneficial to make accurate conclusions about task execution. Hence, it needs to be mounted with sensors that provide different modalities, and the readings gathered from these sensors need to be interpreted to come up with accurate conclusions about the task execution. In order to make inference on the state of the world, a fusion procedure is essential to combine such readings gathered through distinct sensors. The variety of the sensory modalities is also crucial for the diagnostic tasks. Most of the time, common indicators appear in case of an anomaly. Therefore, it may be more challenging for the robots to perceive and infer with these indicators about the anomalies through limited sensory modalities. The data gathered through each sensor should be combined and fused with early or late fusion techniques to identify what went wrong during task execution. In this thesis, onboard anomaly identification methods are investigated to classify unexpected circumstances in tabletop object manipulation scenarios.

Associations between potential anomaly classes and anomaly cases must be established in an identification algorithm. This task must be achieved by the robot to reveal the anomaly type by applying a time series analysis of gathered observations during run time. Such a time series analysis makes it possible to incorporate relations between anomaly indicators that may be gathered at distinct time steps.

Figure 1.1 illustrates snapshots of an anomaly case along with the symbolic representation of the corresponding scenario on the bottom. The robot is tasked to construct a structure by stacking the cubical objects on top of each other. In this particular scenario, a 3-block tower from the given blocks A, B and C needs to be constructed.

The robot stacks block B on top of block A with an offset which causes their centers of mass to be not on the same vertical axis at time $t-2$. In the next time step $(t-1)$, the robot aligns itself to grasp block C to replace it on block B. After the robot places the block at hand at time step t, the tower collapses due to the violation of the physical balance of the tower. The unstable placement at time step $t-1$ does not yield the robot to immediate damage. However, this instability causes an anomaly in upcoming steps due to a previously executed action that resulted in block B not being well-aligned with



(a) (b) (c)

(d)

Figure 1.1 : An anomaly case where a 3-block tower collapses due to an earlier unstable action execution [1].

block A. In order to determine what went wrong in which time step, the robots need to execute an identification procedure that processes the gathered observation sequence. Furthermore, a temporal analysis is required to incorporate anomaly indicators at time step $t-2$ for the task that is executed at time step $t-1$. As a consequence, a temporal analysis that takes into account the observation sequence as a time series is required.

1.1 Terminology

In this subsection, the terminology related to the focused anomalies is presented by introducing necessary definitions. Then, active research topics related to anomalies are explained.

1.1.1 Anomaly vs. Failure

In this thesis, *anomaly* and *failure* terms are defined taking into account how they occur in environments as described analogously in [9]. These descriptions are adapted to the robot manipulation domain. Violations of expectations, specifications or rules are treated as *anomalies*. One should note that these violations may also arise due to external sources (i.e., beyond the control/operation of the robot). On the other hand, an anomalous execution that yields unexpected operational outcomes is treated as *failures*. Figure 1.2 illustrates the cases given in these definitions. Note that, an anomalous execution does not always necessarily lead to a failure.

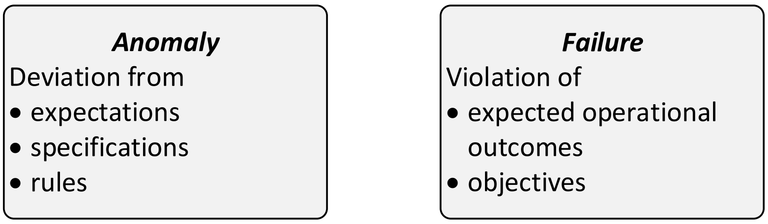


Figure 1.2 : Illustration of the anomaly and failure definitions.

To exemplify, consider a manipulation episode where a robot interacts with a set of objects (i.e., picks, puts down, or pushes) in the environment. An unexpected location change of the target object by an external agent/event is treated as an anomaly. Whereas, when the robot fails in picking the object by itself, this case is treated as a failure since it is related to a violation in the expected operational outcome. In this thesis, failures are considered anomalies since they are special types of anomalies according to the given definitions. Therefore, all deviations are entitled as anomalies.

1.1.2 Detection, identification and recovery

Recognizing and dealing with failure situations requires a diagnostic procedure including generally three main steps: *detection*, *identification*, and *recovery* [10]. In the anomaly detection phase, the robot becomes aware of an unexpected outcome. This is achieved by monitoring the execution continually to keep track of the execution state to check if outcomes are as expected. Whenever an anomaly is detected, the robot needs to execute procedures to recognize or identify the case to explain what went wrong. Such methods are called anomaly identification procedures. In the recovery phase, the robot takes necessary actions to return to a nominal state to recover from the anomaly. This phase is named recovery.

Figure 1.3 presents a flow of an example scenario that is presented in Figure 1.1. In this particular scenario, the case where a tower of objects collapses due to an earlier unstable action is presented. First, the robot perceives the environment in which it exists through the sensors that it possesses, and it maintains a knowledge base (KB) to keep track of its belief in the world state. The KB is updated (i.e., by a visual world

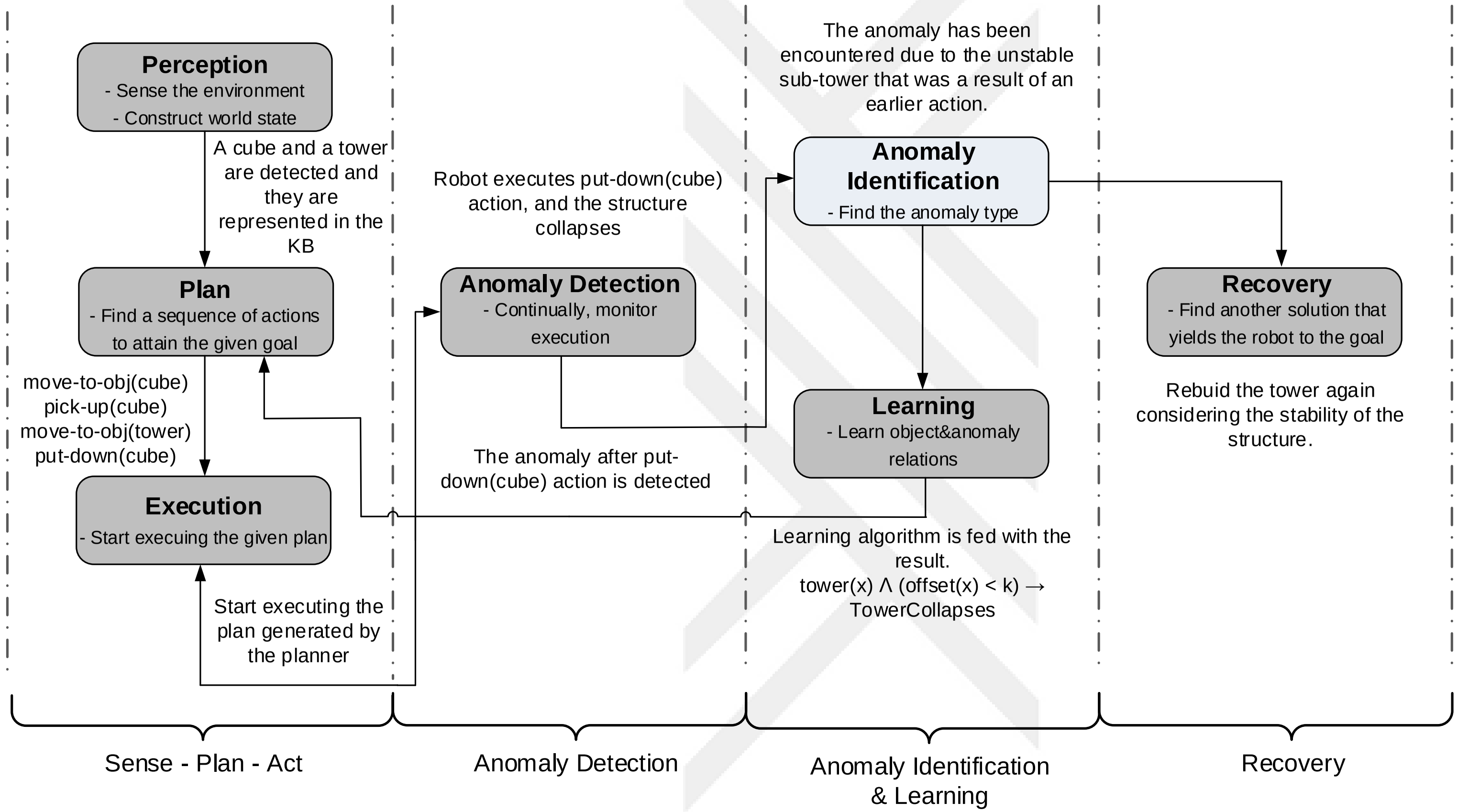


Figure 1.3 : An example scenario illustrating the workflow of corresponding modules.

modeling system) after perceiving the objects (a cube and a tower) in the environment with their specifications (color, location, shape, etc.).

A planner provides the sequence of actions to achieve the objectives that might lead the robot to its desired goal. It may generate this sequence automatically by using the domain knowledge and the built experience on action outcomes that contains knowledge about the available actions, or a domain expert directly gives it. The robot first picks up the object and places it on top of the stack in this particular case. Afterward, the robot starts to execute these consecutive actions in the plan. Note that some low-level phases, such as generating a path plan, are not illustrated in the figure for brevity.

While the robot executes the consecutive actions provided by the planner to attain its goal, an anomaly detection procedure runs concurrently to check whether everything related to the execution goes as planned according to the expectations. However, the tower collapses due to an earlier unstable placement action in this particular case. The anomaly detection process notes this deviation, and it triggers the anomaly identification process, and the type of the anomaly is identified as an earlier unstable placement action. This interaction may be recorded to update the knowledge base for the failure context [11]. Finally, the recovery module comes up with another plan to rebuild the tower stably.

After the robot reveals the anomaly type, necessary recovery actions are suggested by a recovery procedure. In order to relate and incorporate anomaly types with objects or tasks, the robot may construct hypotheses for the upcoming tasks by a learning procedure. These findings may be fed into the domain knowledge to be used in future tasks.

1.2 Proposed Solution

In this thesis, the anomaly identification problem is addressed by taking into account different sensory modalities and by applying a time series analysis on the observation sequences of the robot. The described problem is addressed by processing sequential observations at different granularity. Consequently, a distinct multi-modal anomaly identification framework is proposed at two different granularities. The following levels are considered for processing features:

1. High-level symbols are extracted from data gathered through multi-modal sensory system. Visual predicates provided by a visual world modeling system and auditory symbols provided by a classifier are used as input to the presented framework. An early multimodal fusion algorithm combines these data, and the fused multimodal data are used to capture anomaly indicators within a deep learning-based framework.

2. Convolutional features are extracted from raw 2D images gathered through an RGB-D camera. Auditory data gathered from a microphone are processed within convolutional neural networks (CNNs) with extracted auditory features from the data. Proprioceptive data is also collected and processed with a CNN. These data are processed separately, and a late fusion technique is applied to incorporate the results of each sensory modality stream to identify anomaly cases.

1.3 Contribution

In this thesis, two anomaly identification frameworks are presented. Each framework incorporates various sensory modalities at different granularities to reveal anomaly types. The contribution of the thesis is fourfold and is summarized within the following subsections.

1.3.1 Symbol-based anomaly identification framework

In order to address the anomaly identification problem of everyday object manipulation tasks in unstructured environments, a symbol-based identification framework is presented first. This proposed framework adopts an early fusion mechanism to incorporate different sensory modalities: vision, audition, and proprioception. A visual world modeling system is used and the domain-specific knowledge may be embedded easily in the solution. To the best of the authors' knowledge, this is the first time that the auditory modality is considered to classify object-related perceptions (i.e., fall) for the anomaly identification problem of everyday object manipulation tasks. The framework is evaluated on real-world cases, including anomalies with a Baxter robot.

1.3.2 Convolutional three-stream anomaly identification (CLUE-AI) framework

Since a visual world modeling system may not always be available in a robotic framework, a convolutional three-stream anomaly identification framework (CLUE-AI) is proposed. Unlike the symbol-based anomaly identification framework, CLUE-AI processes 2D images gathered from an RGB-D camera without the need for any preprocessing and domain-specific knowledge. The novel framework processes different sensory modalities and adopts a self-attention mechanism to capture anomaly indicators in an observation sequence. Different sensor modalities are then incorporated with a late fusion mechanism to combine the results from different CNN architectures designed to process each sensory stream. The experimental evaluation of the framework is done on a Baxter robot in real-world anomaly cases.

1.3.3 Multimodal analysis

In this thesis, multimodal anomaly identification methods are proposed where conclusions about the types of anomalies are drawn from the fused sensory data gathered from different sensors. Both early and late fusion mechanisms are implemented for the presented solutions. In order to show the contribution of each sensory modality on the anomaly identification performance, multimodality analyses are presented for each presented framework.

1.3.4 Symbolic-level features vs. convolutional features

The comparison of the proposed anomaly identification frameworks is presented considering the processed feature types. Advantages and drawbacks of processing and fusing symbolic-level features and convolutional features are analyzed for the investigated anomaly cases.

1.4 Organization of the Thesis

This thesis is organized as follows. Chapter 2 presents the state of the art of the anomaly identification literature. This chapter is followed by the formal definition of the anomaly identification problem along with the investigated anomaly types (Chapter 3). Chapter 4 presents fundamentals of the underlying structures that are employed in the proposed framework. Chapters 5 and 6 present the anomaly identification frameworks that process symbolic-level features and convolutional features, respectively. Chapter 7 concludes the thesis with discussions and possible future directions.

2. LITERATURE REVIEW

In this chapter, literature on the recent research studies on different components for anomaly diagnosis is summarized. Since the diagnosis consists of detection, identification, and recovery sub-fields and they are highly interrelated, the literature of each sub-field is investigated under distinct headings.

There exist three different taxonomies on anomalies [12–14]. The first taxonomy distinguishes internal and external anomalies. In this taxonomy, deviations related to environmental problems correspond to external anomalies [13]. On the other hand, the anomalies related to the beliefs of the robot itself are categorized as internal anomalies in this taxonomy. Another taxonomy classifies anomalies considering their sources: physical and human [12]. Yet another taxonomy presents anomalies of smart manufacturing environments [14].

2.1 Anomaly Detection

Anomaly detection is intensively investigated in the literature [15–19]. In [15], the execution monitoring task is investigated under three categories: external monitoring, expectation-based monitoring, and model-based monitoring. A common strategy to detect anomalies is to compare the expected outcomes of actions and actual observations [16]. Model-based methods are also widely investigated for anomaly detection [17]. In another comprehensive survey, anomaly detection methods are presented and categorized based on the underlying ideas of the solutions [18]. Yet another survey on anomaly detection presents related studies considering different types of robotic system characteristics [19]. Failure detection can be achieved without considering a predefined model by checking sensory data [20].

In a model-based method [21], normal working conditions are modeled instead of failure cases, and deviations from the models are used to detect motion failures. A coBot service robot is used to test the method in a dynamic environment. The study focuses on detecting displacement errors by using the outputs of the wheel encoders and the localization algorithm. In [22], a model-based actuator failure detection method is presented for non-linear systems. The proposed method employs parity equations for detecting deviations between expectations and observations. Pioneer 3-AT robot is used to evaluate the performance of the proposed method.

An analytical redundancy approach is used to detect failures on an iRobot Roomba [23]. The study consists of an observer that is responsible for tracking the error between the actual sensor values and nominal robot dynamics (odometer). Furthermore, uncertainties in model parameters (i.e. temperature, battery charge, etc.) and non-linear dynamics of the robot are also considered during this failure detection phase. Adaptive bounds are defined for these situations. In case of measurements out of these boundaries, a failure case is assumed. The following motor failures are considered during the experiments: the motor does not response, the motor stops working, and the performance of the motor degrades. Since each wheel has its own observer, the failure is simply located considering the wheel that raised the alarm. After detecting the failure, the magnitude of the failure is also identified by the model.

An Entity-Component-System (ECS) is maintained to detect unexpected deviations [24]. The system stores dynamic information about object entities. Therefore, actions that manipulate these entities are stored separately. Deviations are also classified by considering resources. The proposed method is tested on a KUKA Lightweight robot on object manipulation scenarios. In a data-driven method, anomalies are detected with a three-phased method: data filtering, attribute grouping, and outliers detection [25].

A monitoring algorithm is implemented for the RoboCup soccer environment to modify and feed action models in a planner [26]. In order to detect failures, expected outcomes of the actions are compared with the actual observations during the execution. A focused anomalous region optimization detector [27] which uses a cross-entropy method [28] is used to enlist the set of significantly deviated states. Then, this set of states (regions of anomaly) are used to update the planning model.

Probabilistic methods are also analysed in the literature for the failure detection task. In [29], self organizing maps (SOPs) are analysed spatio-temporally to detect failures. In [30], HMMs (hidden Markov models) are employed for failure detection by maintaining system feedback with torque and force signals. In another work [31], uncertainties in the environment are detected by maintaining a probability distribution on the robot's expectations. Variants of HMMs are also considered to address the anomaly detection problem in robotic assistance tasks [32, 33]. Another study investigates HMMs with Hellinger distance analysis to detect anomalies in aquatic drones and socially assistive robots [34]. A hierarchical Dirichlet process (HDP) based method is also investigated to detect anomalies during robotic assembly by processing force-torque signals [35]. In [36], neural networks and Kalman filters are used in a mobile robot system to detect sensor and mechanical failures.

In a deep learning based method [37], anomalous behaviours that arise due to cyber-security attacks in robotic systems are detected with various autoencoder settings. To do so, logs are transformed into images to train these autoencoders. Another work addresses navigation anomalies by processing multimodal images in natural outdoor environments [38]. LSTM-based variational autoencoders are also investigated to detect human-robot interaction anomalies [39]. In another study [40], CNNs extract spatio-temporal features and they are fed into bi-directional LSTMs to detect anomalies in surveillance stream data.

Abnormal situations that arise during execution can also be detected by maintaining control rules. In [41], control rules are represented in a domain-specific language to analyze hardware failures. In [42], metric temporal logic (MTL) is used to represent action control formulas for monitoring the action in execution. Hard-coded formulas are checked during execution with the progression algorithm [43] to detect failures. The proposed algorithm is tested on a Pioneer 3-AT robot with real word experiments. In another work [44], execution monitoring for unmanned aircraft systems (UAVs) is achieved by using a progression algorithm that operates on action control formulas represented in temporal action logic (TAL).

Planning is defined as finding a sequence of actions to achieve a goal. Methods including the planning phase for execution monitoring are also investigated in the literature. In [45], actions are represented in an extended way by representing action states in consecutive time steps. Furthermore, this extension also includes degraded versions of these actions. Failure detection is achieved by using these extended models.

2.2 Anomaly Identification

In the literature, logic-based, sensor fusion-based, expert systems, and probabilistic methods are investigated to reveal the types of anomalies. The following subsections explain these studies under distinct headings.

2.2.1 Logic-based methods

Logic Programming with situation calculus [46] is studied to explain unexpected deviations [47]. Cost-assigned hypotheses are maintained in a pool, and actions and their effects are reasoned through situation calculus. Sensor failures, misbeliefs and external events are investigated in the scope of the study. A hypothesis pool is maintained to explain failures, and hypotheses in the pool are updated during execution. In case of an inconsistency, cost values of these hypotheses are updated and the minimum-cost hypothesis is selected until conflicts are resolved among hypotheses.

In a logical inference based method [48], the causes of the failures are categorized under three headings: failure on producing the desired effects of an action, deviations from expectations in the world and exogenous events. The main idea of the proposed method is detecting inconsistencies between the theory and the model of the world. The method is tested on a delivery robot in an environment where the robot is tasked to deliver some objects, such as envelopes, to their destinations. Logical inferences are achieved about failures. For instance, when the robot detects an object in a place which it believes that object is in another room, it tries to pick it up to infer whether it is a ghost object. If the action is successful, the robot updates its belief on that object. The temporal dimension of the problem is not considered in such solutions.

2.2.2 Clustering-based methods

Clustering based identification methods have also been studied [49]. Global fuzzy c-means clustering algorithm is used to construct clusters to find the causes of failures. During the execution, new clusters are constructed for abnormal situations due to the outlier data gathered by the sensors. After detecting the failure with clustering, identification is provided by considering the sensor values during the failure and closest clusters. The performance of the method is evaluated on an 18-degrees-of-freedom hexapod robot with five different cases: normal conditions, uphill inclination is changed, tied leg, deactivated position sensor and changeable light conditions.

Multi-level sensor fusion is used to detect and classify failure cases [50]. Sensors are clustered as groups, and their outputs are interpreted to detect and isolate abnormal situations. Four phases are invoked in the proposed method: conflict monitoring to detect inconsistencies among the sensor groups, confidence level computation for calculating the confidence of the sensor measurement, fault isolation to isolate faults by using the decision making rules and information fusion to fuse different sensor sets' information for confidence level. The method is tested on the navigation domain with obstacle avoidance scenarios, and camera and infrared sensor faults are investigated on simulation scenarios.

2.2.3 Probabilistic methods

Particle filters (PFs) are proposed in the FDI literature [51]. In [52], failure identification is achieved by using a hierarchical representation. Failures are represented in a tree structure. In this representation, leaf nodes correspond to faults where internal nodes correspond to aggregation of similar nodes to group them. Dynamic Bayesian networks (DBNs) and particle filters (PFs) are proposed to identify failures in [53]. In this two-phased method, DBNs are implemented to find the candidate electric machines that cause a failure. Then PFs are used to keep track of these candidate machines to find the cause of the failure. In [54], particle filters and Kalman filters are compared for failure detection and identification for DC motors.

HMMs are also used to identify anomaly cases. In [55], an HDP-HMM is employed as a Bayesian time series model to learn sensor signatures related to failures. The proposed method is evaluated in industrial robotics tasks. In another HMM-based method, gradient analysis is studied to identify anomaly cases [56]. In the particular study, pick and place actions on a humanoid robot are analyzed. In a non-parametric HMM-based solution [57], force-torque signals, velocity and tactile data are processed to detect and identify anomalies that arise during cooperation tasks. In another non-parametric method, autoregressive HMMs (AR-HMMs) perform introspection to identify nominal and anomalous activity by processing force-torque data [58]. Considering a hierarchy between the hidden states and the actions, anomalies are identified with Hierarchical Hidden Markov Models (HHMMs) and Particle Filters (PFs) in a mobile robot performing object manipulation tasks [59, 60].

2.2.4 Deep learning methods

Visual features of visual inputs can be extracted with convolutional neural networks (CNNs). In [61], CNNs are used to train a model for task execution in a simulation environment to investigate visual failure modes with a single modality. An autoencoder-based method that uses stacked denoising autoencoder (SDA) is proposed [62] to identify the health state of rotary machines. The identification of health states is achieved with a three-step method: division of the health states into training and test groups, construction of a deep hierarchical structure with greedy training and validation of the network. A deep transfer learning (DTL) method is proposed to handle motor bearing failures [63]. In this work, a three-layer sparse autoencoder structure is presented to extract features from the raw data. In another work [64], visual features extracted with CNNs are combined with temporal features gathered from HMMs and these features are used to classify anomalies with multilayer perceptrons (MLPs) in activities of daily living (i.e., feeding) scenarios.

2.2.5 Other methods

In a mobile robot domain, locally linear models (LLMs) with adaptive threshold bands and model error modeling (MEM) are proposed to address anomaly detection and identification problems [65] There may also be situations where the planner is not able to generate a plan given a goal state. Causal graphs and domain transition graphs are used to analyse such cases [66]. Causes of why the planner is not able to come up with a plan, excuses, are classified under three categories: acceptable excuses, good excuses and perfect excuses. Acceptable excuses are the explanations with lower facts given two good excuses. A minimum costly excuse is denoted as a perfect excuse. Two heuristics, LM-cut and enhanced-additive, are used with A* algorithm to find excuses.

Anomaly classification problem is also studied in other domains. In a medical problem [67], multiple-instance learning algorithms are investigated to classify anomalies of mammography images. Unsupervised learning techniques are also studied to classify anomalies of powder bed images [68]. In another work [69], SVMs are used to analyze device logs to classify anomalies in a device network domain. Another work classifies anomalies of hyperspectral images with a linearly constrained minimum-variance classifier.

Different from earlier methods, in this thesis, anomaly classification problem is addressed for everyday object manipulation tasks. Multimodal data (visual, auditory and proprioceptive) are taken into account within a designed deep learning framework for the identification of object manipulation anomalies for service robots. Moreover, auditory modality is used to classify object interactions (i.e., fall).

2.3 Recovery

In general, a plan is generated either manually or by using a planner to recover from a deviation/anomaly/failure. In [70], an algorithm for recovery of action execution failures is proposed for multi agent systems. The method is tested on a simulation environment with Khepera II robots. When a failure occurs, the action schema is updated by locking the failed action. By doing so, alternative actions are selected for the resolution of the failure by the TLPlan planner [71]. In another work [72], a method that includes function-checking and diagnostic-plan strategies is proposed to gain diagnostic information for failure diagnosis. The method is tested on robots using JSHOP2 planner.

Since replanning is a costly operation compared to repairing, plan repair is investigated in the literature to maintain robust plan execution in case of failures. In [73], the advantage of repairing the plan is shown against replanning. Plan stability is introduced to find new plans instead of replanning when deviation occurs. Plan stability is associated with the difference of two plans. A local-search-based planner, LPG [74] is used with a heuristic considering the plan cost, end time, increase in the number of search steps, estimation of the distance between the new plan and the current to find stable plans in case of a deviation. In [75], refinement (adding actions to the plan) and unrefinement techniques (removing constraints using planning heuristics) are used to repair a plan. The proposed method is implemented on VHPOP planner [76] resulting with the POPR planner and compared with replanning from scratch and another plan repair method GPG [77].

A hybrid planning approach using both partial order planning (POP) and hierarchical planning is presented in [78]. Abstract and elementary actions are considered. Abstract actions correspond to a set of elementary actions where elementary actions correspond to atomic actions. The proposed system is implemented on HiPOP [79] planner. Repair procedure consists of the following steps: removing the steps that are in the failed plan but not in the domain, removing violated temporal steps, removing actions preserving the hierarchy, interacting with deadlines and organizing forbidden actions. Experiments take place in the survivors [79] and parking domain.

Replanning is also investigated to achieve recovery [45]. In case of a failure, the robot generates a plan from the current faulty state to a repaired state. In order to define the repaired state, three rules are considered: Fixing all the functionalities in the faulty state, satisfying the preconditions of the upcoming action and maintaining the state in such a way that there are no open preconditions. If there is not a plan in this case, the robot plans to a safe state where it does not affect the other robots.

Representing failures within an ontology is also a method to recover from failures. A failure ontology is represented as a graph with indication nodes [80]. The indication nodes connect to diagnostic nodes. Then the proposed meta-cognitive loop (MCL) moves into the guide process. Potential responses are listed considering their utilities, then one of the recovery methods from the list is implemented. If a failure is encountered before, the previous response is marked and indications are updated with the new failure.

Model-based methods are also presented in the recovery literature. In [81], a model-based failure diagnosis method is presented where three failure scenarios on a hard drive are considered with three different types of severity. In the first scenario, failure is not serious, and the robot is able to return its nominal state. In the second one, the robot returns its nominal state with a degraded performance. Final failure is the case where it is too serious that the robot is not able to go on executing. These failure modes are modelled within a probabilistic hybrid automata (PHA). Finally, reconfiguration is achieved by using the kinematics.

Petri nets are studied to recover from erroneous states in the blocks world domain [82]. Failures are recovered considering the failure type: known failures and unknown failures. In case of a known failure, the Petri net finds a sequence of actions for execution. Otherwise, in case of an unknown failure, the robot learns the recovery from a human by demonstration, and extends the current Petri net.

Probabilistic structures, such as hidden Markov models (HMMs), are also analysed in the literature to find the most appropriate execution mode for a mobile robot [83]. *Plan, stop* and *react* are defined as modes for the robot, and the robot transfers among these states and responds to situations during execution.

An agent programming language is also proposed to handle anomalies that arise in multi-robot domains [84]. In such a method, a robot that encounters an anomaly case, ~~it~~ can execute a diagnostic procedure locally and ask another robot for help. Natural language instructions can be processed when the robot has missing information about its domain [85]. For example, when the robot does not have a push action to open a door, a human intervenes with a verbal statement. The robot parses this statement and inserts the new action to its domain as a result of parsing. SapaReplan is used to construct plans which is an extended version of the SapaPlan planner.

A fuzzy logic based expert system is proposed to diagnose faults in [86]. In this method, a three-phased data-driven procedure is invoked that includes the following steps: acquiring information, making inferences and taking actions. In the first phase, information is extracted. Second phase includes if-then rules that are gathered from a domain expert of that area. Afterward, the system is returned to its nominal state by taking some actions in case of a failure. Drawback of the proposed system lies in the high dependence of the quality of the expert knowledge.

3. PROBLEM DEFINITION

In this chapter, the anomaly identification problem for service robots is formally presented. Then, the investigated scenarios and the focused anomaly types that occur during the execution of these scenarios are explained.

3.1 Problem Formulation

A service robot needs to interact with the environment in which it operates during task execution. The environment may contain objects or other robots. Humans may also exist in robotic environments to collaborate with robots to fulfill a task. As everything may not go as planned, deviations from expectations, called anomalies, can occur during task execution. Therefore, a continuous monitoring process is a necessity to detect such situations. After detecting anomalies, explanations related to these anomalies need to be presented for efficient recovery. This Ph.D. research aims at implementing a framework that has the ability to address the question "What went wrong during execution?".

A service robot needs to perceive its environment to become aware of the real world with the readings gathered through its sensors. Various types of sensors may be mounted on the robot, and the readings from these sensors must be processed appropriately for reasoning on the environment and acting accordingly. These readings correspond to observations related to the environment. All these readings and the robot's internal state (i.e., the positions of its actuators) make up the execution state. The execution state is directly observable by the robot. On the other hand, the robot cannot directly observe the hidden state that contains whether there exists an anomaly. Therefore, an inference algorithm is required to decide on the hidden state. Let each sensor is represented and denoted with s_m where m corresponds to the index of the sensor that provides a distinct sensing type. The whole set of these sensors is denoted with S and is represented as in equation 3.1.

$$S = \{s_0, s_1, ..., s_m\} \tag{3.1}$$

The robot gathers readings through each sensor corresponding to an observation at a distinct time step t, and this observation taken from a distinct sensor (s_m) is represented with $o_t^{s_m}$ for that time step t. In this study, three sensory modalities are considered: vision, audition and proprioception. Images are processed as visual readings, audio data are considered as auditory inputs, and gripper-related data are processed as proprioceptive readings. Since each sensor provides readings asynchronously, these readings eventually need to be fused together to come up with a consistent interpretation. As a result, these readings need to be aligned to end up with fused observations. Therefore, a fusion procedure is needed to construct a fused observation at a time step. A fused observation tuple at time step t can be denoted with x_t as in equation 3.2

$$x_t = (o_t^{s_0}, o_t^{s_1}, .., o_t^{s_m}) \tag{3.2}$$

where $o_t^{s_m}$ represents the observation gathered through the sensor s_m at time step t. The robot needs to maintain an observation history (H) that includes these sequentially gathered observations during execution, and equation 3.3 formulates this concatenated observations as follows:

$$H = (x_0, x_1, ..., x_t) \tag{3.3}$$

where each element (x_i) corresponds to a gathered observation. Observations can directly be perceived from the environment through the sensors. A hidden state (z_i) corresponds to the execution state's anomaly type (ground truth). It gives information about the execution state, denoting whether there is an anomaly (if so, it indicates the anomaly type). Since the hidden state is not directly observable, a diagnostic procedure is needed to infer the anomaly type. Equation 3.4 presents the hidden state sequence (Z) that corresponds to an observation history.

$$Z = (z_0, z_1, ..., z_t) \tag{3.4}$$

The domain of this hidden state is represented with the set given in equation 3.5.

$$A = \{safe, anomaly_1, anomaly_2, ..., anomaly_n\} \tag{3.5}$$

The first element of this set (*safe*) corresponds to the cases where there is not any deviation from the expectations in that corresponding state. Other elements of this set

($anomaly_n$) correspond to distinct anomaly cases where n is the cardinality of this set. The robot should continuously monitor the execution to infer the ground truth. In case of an anomaly, the robot needs to have the ability to investigate the observation history and come up with a conclusion about the type of anomaly to explain it. This task may be achieved by assigning probabilities to each ground truth candidate in the observation history. Considering these likelihoods, it can come up with related anomaly classes for each hidden state. This task can be expressed as in equation 3.6.

$$argmax_i(P(z_{0:t} = anomaly_i|H)) \tag{3.6}$$

In this equation, $z_{0:t}$ is the hidden state label sequence where H is the observation history. The robot needs to find the most appropriate anomaly type corresponding to the ground truth by investigating the anomaly type $anomaly_i$ that maximizes the given posterior in the equation.

3.2 Scenarios and Potential Anomaly Cases

In order to analyze case scenarios, Baxter robot is selected. Baxter is a manufacturing humanoid robot produced by Rethink Robotics [87], and it is used to execute selected everyday object manipulation tasks. Baxter is equipped with an LCD head which makes it possible to interact with the environment. It possesses two arms, one of its arms has a five-fingered hand, and the other one has a force-sensing gripper which is used during selected scenarios. An RGB-D camera is mounted on its head which makes it possible to perceive images. A microphone is also mounted on its body to keep track of the occurred sound information. Figure 3.1 illustrates the Baxter robot market with the presented sensors and actuators. Robot Operating System (ROS) [88] is used to integrate and construct the proposed framework. It provides a wide range of libraries and tools for robotic applications. To design low-level behaviors, MoveIt [89] that provides motion planning and manipulation libraries is used.

Since the main focus in this thesis is the identification of anomalies that occur during object manipulation episodes, table-top object manipulation scenarios are examined. Various scenarios including several objects with different shapes and sizes are investigated for manipulation. A set of boxes (cereal, pasta, powder), a set of cans (wafer and tomato paste), a milk bottle, a set of plastic fruits and vegetables (corn,

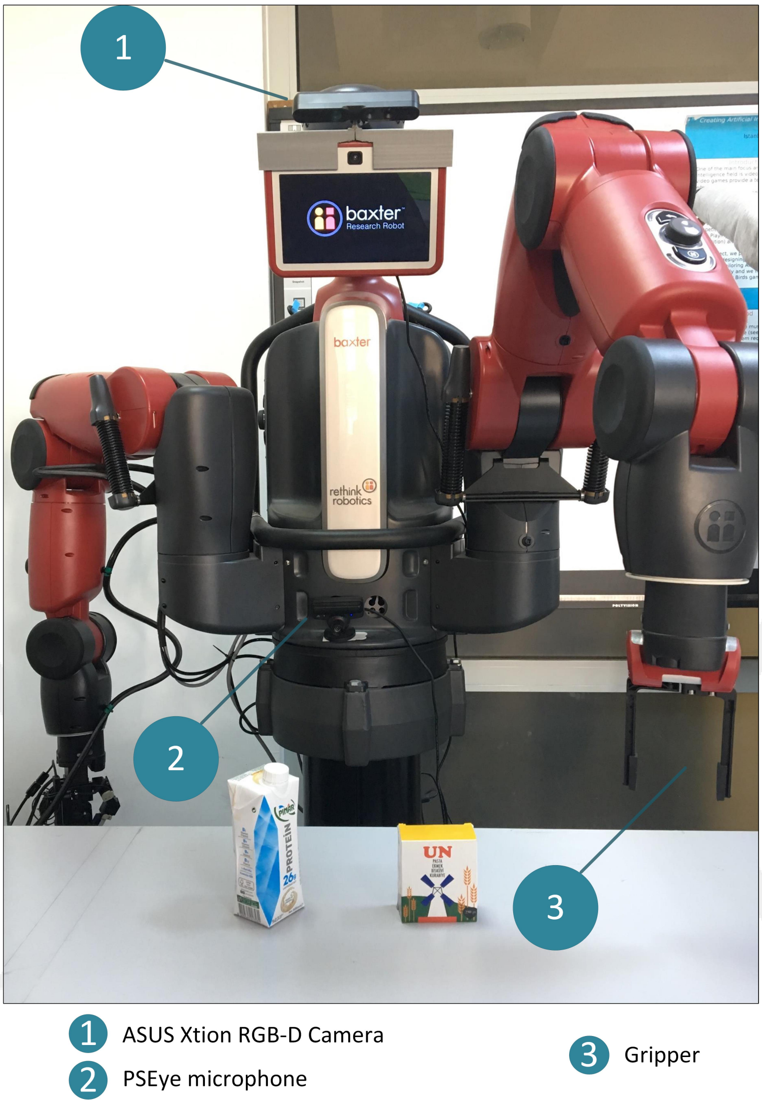


Figure 3.1 : Baxter robot with its selected sensors and actuators.

grape, banana), a set of cubical objects and container objects with various sizes are included in the object set. Figure 3.2 illustrates the objects that Baxter interacts with. As seen from the figure, Baxter is located behind a table on which the selected objects exist.

3.2.1 Investigated manipulation actions

The anomaly identification problem is investigated under the assumption that the robot is capable of executing the following listed actions to interact with the environment and objects: *move-towards-object, move-to-location, pick-up*, *put-down*, *put-down-on*, and *push*. The following items describe these actions.

Figure 3.2 : The objects that are used in the anomaly scenarios.

- *move-towards-object(objectX)*: The robot moves its arm to locate and align its gripper to objectX's location.
- *move-to-location(destination)*: The robot's arm is located in a given destination coordinate.
- *pick-up(objectX)*: The robot closes its gripper to grasp objectX.
- *put-down(objectX)*: The robot releases currently-gripped objectX from the gripper to the table.
- *put-down-on(objectX, objectY)*: The robot releases currently-gripped objectX from the gripper on top of objectY.
- *push(objectX, axis, distance)*: The robot pushes objectX with its arm along a specified axis considering a given distance.
- *pour(objectX, container)*: The robot pours objectX into the given container object.

3.2.2 Scenarios and potential anomaly cases

The robot may need to execute actions described in Section 3.2.1 consecutively in order to accomplish a given task. In order to analyze anomalies, the following scenarios are considered in this thesis.

- The robot is tasked to grasp an object with its gripper.
- The robot is tasked to release an object from its gripper to the desired location.
- The robot is tasked to push an object along an axis for a specified amount.

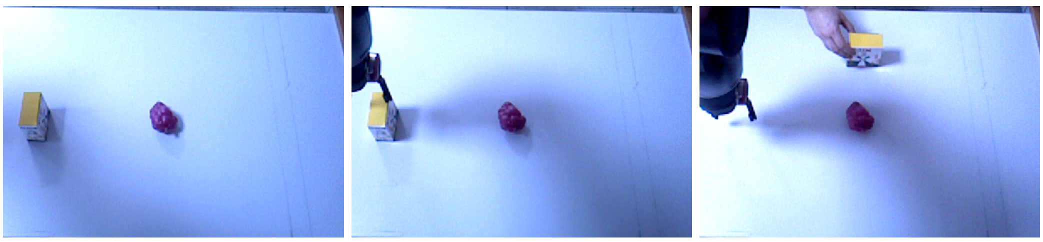
- The robot is tasked to construct a tower from a set of blocks that have cubical shapes.
- The robot is tasked to pour the content of one container which includes objects into another container. The second container may be empty, partially-full, or full.

During the execution of these particular case scenarios, miscellaneous anomalies may arise. The following subsections justify the potential anomaly cases.

3.2.2.1 Location change anomaly (LOC)

The robot's interaction with the objects during manipulation necessitates keeping track of the states of objects for efficient execution. Furthermore, there may be cases where there are other humans or robots in the environment of the robot. In such cases, the robot also needs to be in communication with other's actions. When the robot is not aware of the changes in the environment that arise due to the actions of other agents, the robot may encounter anomaly cases related to these changes. In this thesis, the cases where an object's location is changed beyond the knowledge of the robot itself are investigated. The abbreviation *LOC* is used for such cases.

Figure 3.3 illustrates the snapshots of an anomaly case that includes LOC. Each illustration is captured from the head-mounted RGB-D camera of the robot. First, the robot gathers the initial scene (Figure 3.3(a)) which includes a powder box and plastic grape toy on the table. The task of the robot is grasping the powder box. Therefore, the robot moves its arm to the location of the object (Figure 3.3(b)). While the robot approaches the powder box, the object's location is changed by a human in the environment (Figure 3.3(c)). This anomaly type corresponds to such deviations.

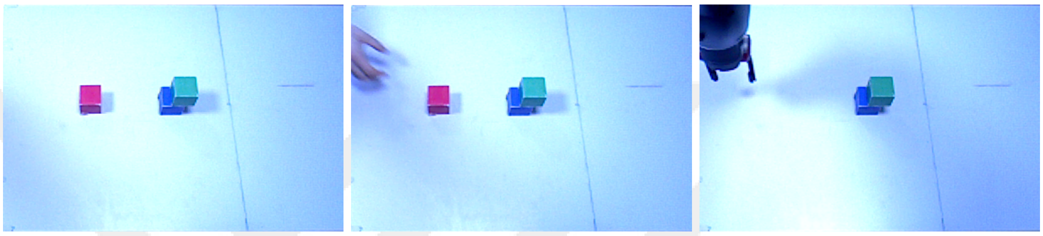


(a) (b) (c)

Figure 3.3 : Snapshots of the anomaly scenario that includes location change anomaly (LOC).

3.2.2.2 Object disappearance anomaly (DIS)

While the robot interacts with an object during a manipulation episode, the object may be taken out of the point of view (i.e. environment) of the robot. Such a situation may lead the robot to encounter an anomaly case. Figure 3.4 presents a sequence of snapshots while the robot performs a tower construction scenario of cubical objects. After the robot perceives the scene (Figure 3.4(b)), it moves its arm through the object. However, while the robot moves its arm, the object is taken out of the environment by a human without the knowledge of the robot (Figure 3.4(c)). This anomaly type explains such cases. The abbreviation *DIS* is used for such cases.

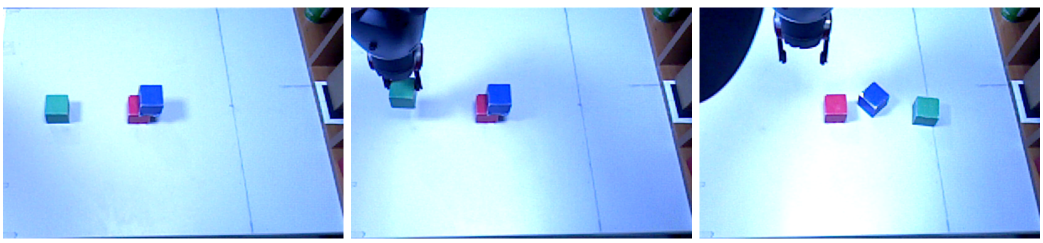


(a) (b) (c)

Figure 3.4 : Snapshots of the anomaly scenario that includes object disappearance anomaly (DIS).

3.2.2.3 Earlier unstable action anomaly (EUA)

While the robot constructs a tower of objects, the tower may collapse due to a previously built unstable sub-tower. This instability arises due to misaligned centers of mass of the tower components. Figure 3.5 illustrates an example scenario where an anomaly occurs due to an unbalanced sub-structure. After the robot observes the environment with the objects to be interacted (Figure 3.5(a)), it grasps the green block with its gripper (Figure 3.5(b)). After the robot places the currently grasped object on top of the unbalanced sub-structure, the tower falls down (Figure 3.5(c)). This anomaly type corresponds to such deviations. The abbreviation *EUA* is used for such cases.

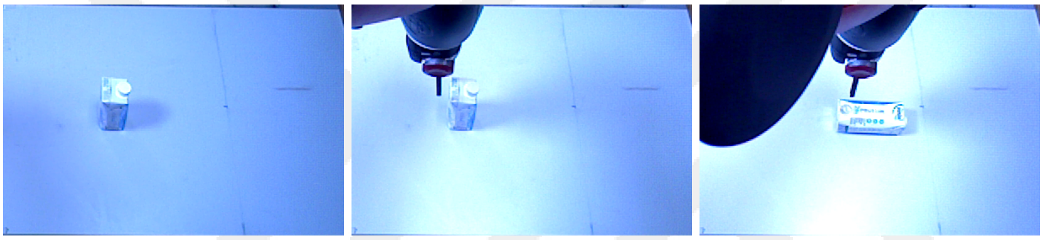


(a) (b) (c)

Figure 3.5 : Snapshots of the anomaly scenario that includes a fall-down of a tower due to an earlier unstable action (EUA).

3.2.2.4 Overturning anomaly (OTA)

When the robot is tasked to manipulate an object by pushing it, the point of push should be selected properly. Otherwise, the object may overturn and change its orientation. Such cases may lead the robot to a deviation from the expected outcomes. Figure 3.6 shows an example sequence of this kind of anomaly cases. In Figure 3.6(a), the robot is tasked to push a milk bottle along the horizontal axis for a given amount. To do so, the robot moves and aligns its hand for manipulating the bottle (Figure 3.6(b)). However, due to an unstable pushing point, the object overturns and changes its orientation (Figure 3.6(c)). Such deviations are explained as an overturning anomaly, and *OTA* is used as an abbreviation for this anomaly type.



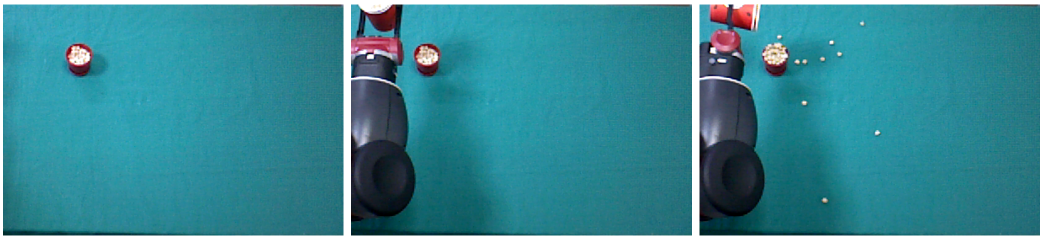
(a) (b) (c)

Figure 3.6 : Snapshots of the anomaly scenario that includes cases where an object falls while it is pushed by the robot along a given axis (OTA).

3.2.2.5 Spilled object anomaly (SPC)

In kitchen environments, the robot usually needs to pour objects from one container into a destination container. The destination container may be empty, partially full, or full. However, the robot may not perceive the fullness of the destination container. Therefore, while pouring one container's content into another one, the robot may not infer when to stop pouring. Consequently, the destination container may overflow, and the content of the first container may be spilled on the table. This anomaly type corresponds to such deviations and the abbreviation *SPC* is used.

Figure 3.7 illustrates an anomaly case as snapshots where an SPC occurs. The robot is tasked to pour a glass of chickpeas into a container that is placed on the table. Note that the container on the table is already partially full of chickpeas (Figure 3.7(a)). The robot aligns its arm to the container to start pouring (Figure 3.7(b)). However, since the robot cannot detect the fullness of the destination container, the poured objects (i.e. chickpeas) are spilled on the table 3.7(c)).



(a) (b) (c)

Figure 3.7 : Snapshots of the anomaly scenario that includes cases where a container overflows (SPC).

3.2.2.6 Full container anomaly (FCA)

A robot may need to interact with a container object in kitchen domains to place objects into it. However, the robot may not have the ability to detect the fullness of the container. Different from the previously explained anomaly type (SPC), this anomaly type corresponds to the cases where an object is placed or stacked into a container that is already full. As a result, the robot's placement of the object into the container results in the fall of the object.

Figure 3.8 presents a sequence of snapshots that includes an example case of the presented anomaly. In this case, the robot is tasked to place a plastic pear into a container. The container already includes several plastic fruits inside. First, a path plan is constructed to locate the robot's arm to the location of the pear. Then, the robot starts executing the path plan (Figure 3.8(a)). After the robot localizes its arm to the pear's location, it grasps the object. Then, the pear is placed into the container (Figure 3.8(b)). However, the plastic pear falls onto the table since the container is almost full of objects (Figure 3.8(c)).

(a) (b) (c)

Figure 3.8 : Snapshots of the anomaly scenario that includes cases where an object is placed into an already full container (FCA).

Table 3.1 summarizes the selected anomaly types in the scope of this study. Each row in the table corresponds to a distinct anomaly type where each column corresponds to the anomaly name, its explanation, the action set that the corresponding anomaly type may occur, and some example scenarios during which the particular anomaly type may occur, respectively.

Table 3.1 : Dictionary of the anomalies that are handled [1].

Anomaly	*Explanation*	*Action*	*Example Scenarios*
The target object is not in its previously detected location but in a different location (LOC)	An object's location is changed by an agent without the knowledge of the robot.	move-towards-object pick-up put-down-on push	- While the robot approaches an object to manipulate (grasp, push, etc.) it, the location of the object is changed. - While the robot constructs a tower from a set of objects, object locations are changed.
The target object that was previously in the field of view disappears (DIS)	An object, that was in the environment, is taken out of the environment by an agent.	move-towards-object pick-up put-down-on push	- While the robot approaches an object to grasp or push it, the object is taken out of the environment. - While the robot constructs a tower from a set of objects, one or more objects are taken out of the environment.
The built structure is collapsed due to an unbalanced substructure (EUA)	While the robot constructs a structure from a set of objects, due to a misaligned sub-structure in a previous time step, the current structure collapses.	put-down-on	The robot tries to construct a tower from a number of objects. Due to an unstably placed object in a previous action, the structure collapses.
An object falls while it is pushed by the robot along an axis (OTA)	An object falls and changes its orientation while it is pushed.	push	- While the robot pushes an object (a box, a tower, etc.) to a given location along a given axis, it falls on the table and changes orientation.
Objects are spilled while they are put into a container. (SPC)	When the robot is tasked to pour objects or liquids into a container, it may over-pour and the object may fall outside of the container.	pour	- While the robot pours objects into an empty or already full container, it may over-pour.
An object falls out of the container since the container is already full. (FCA)	While the robot puts down an object into a container, the object falls out of it since it is already full.	put-down-on	The robot tries to place an object into an already full container. The object falls outside of the container.

4. BACKGROUND

In this chapter, the background to the focused study in this thesis is provided. A brief summary of artificial neural networks, graphical models and deep learning models is presented.

4.1 Artificial Neural Networks (ANNs)

Artificial neural networks (ANNs) are structures inspired by the anatomy of the human brain. The human brain consists of a huge number of neurons connected via links, namely synapses. The communication between these neurons is achieved through electrochemical activity. Inspired from the structure of the human brain, ANNs adopts a similar idea in addressing machine learning problems.

ANNs include neurons connected to each other to store information [90]. Each neuron receives information and processes and transmits it to other neurons. Based on the type of the neuron and its connections, an incoming signal is transmitted through the neuron. Signals are received through the synapses which occur on dendrites. Each connection has a value, namely weight. The weighted summation of these input signals is calculated and fed into an activation function to define the output of that corresponding neuron. Considering the calculated result, this neuron transmits the information to its connected neurons analogously. Figure 4.1 illustrates a single neuron presenting its input signals and their corresponding connections. In the figure, x_n stands for the inputs of the neuron where b_0 is the bias for this neuron. w_n stands for the weights in the network for the corresponding neuron. The neuron simply evaluates the weighted summation of its inputs and provides the result to an activation function F. The output of this function may be an input of another neuron in a more complex network resulting in an ANN. An ANN consists of three layers: input, hidden and output. The input layer includes nodes that collect observations from the environment. Intermediate neurons form the hidden layer, and this layer is followed by the output layer that includes neurons that produce the final result. The weights of the network are adjusted during training with the backpropagation algorithm that computes gradients considering weights so that the desired outputs are observed.

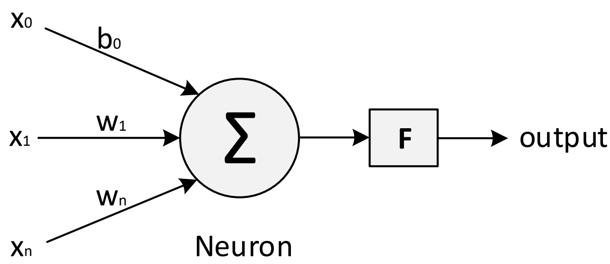


Figure 4.1 : A single neuron with its inputs and output.

Upcoming subsections explain commonly used variants of neural networks in AI and machine learning literature: Recurrent neural networks (RNNs), Long short-term memories (LSTMs), and Gated Recurrent Units (GRUs).

4.1.1 Recurrent neural networks (RNNs)

Recurrent neural networks (RNNs) are structures that include at least one feedback connection from an output as an input [91]. This feedback is organized and applied considering a set of weights that can be set with backpropagation. Multilayer perceptrons (MLP), which may contain one or more hidden layers, generally suffer from incorporating temporal relations. Therefore, RNNs have the ability to store hidden states in previous outputs to be used as inputs for upcoming neurons. RNNs are shown to be successful in several application domains such as natural language processing, speech recognition, etc. The main drawback of RNNs is their lack of handling long-term dependencies.

4.1.2 Long short-term memories (LSTMs)

A long short-term memory (LSTM) [92] is a derived structure of a recurrent neural network (RNN) that has the ability to model long dependencies effectively. Recurrent neural networks (RNNs) can propagate historical information from the past; however, long-term dependencies cannot be resolved well because of the vanishing gradient problem where the gradients that are very small vanish. In such a case, the information that needs to be transferred vanishes: this problem is called the vanishing gradients problem. LSTMs alleviate this problem by constructing a more complex version of the repeating module that exists in RNNs.

LSTMs have a wide variety of applications in several domains. Some application domains, but not limited to, include text classification [93], activity recegnition [94], financial industry applications [95], packet routing [96], etc.

4.1.3 Gated recurrent units (GRUs)

Gated recurrent units (GRUs) [97] are introduced in the literature to handle the vanishing gradient problem that arises in RNNs in longer sequences, and they share similar characteristics with LSTMs. In order to overcome the vanishing gradient problem, update and reset gates are employed in the network, and the degree of information to be transferred is decided by using these gates. Thanks to using such gates, relevant information belonging to the past can be passed or irrelevant information may be eliminated.

4.1.4 Attention

Sequence-to-sequence (seq-to-seq) models (i.e., machine translation) consist of encoder and decoder layers which are generally stacked deep sequence models (i.e., GRU, LSTM). An encoder transforms a given input to a context which can be considered another representation as a summary of the input. Then, this context is processed in the decoder layers to produce the desired output. This context vector may lack remembering very early information related to the given sequence. This problem is addressed by attention. Attention is a mechanism of focusing on the most relevant parts of a given input sequence [98]. It mainly deals with revealing the importance of the observations in a given history for making a decision. There are several types of mechanisms for calculating attention such as additive [98], general [99], location-based [99], dot-product [99] and scaled dot-product [100]. The following subsection summarizes the idea of self-attention with a scaled dot-product mechanism that is employed in this study.

4.1.4.1 Self-attention

Self-attention is a variation of attention that considers only a single sequence to calculate the outputs taking into account observations in the sequence itself. The scaled dot-product attention simply produces the output by applying matrix multiplication. The scaled dot-product attention is calculated as follows [100]:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

where Q is the query, K is the key, V is the value and d_k is the hidden size.

4.2 Graphical Models

In this section, the selected graphical structures used in the solution of the addressed problem are explained. Graphical models are structures whose underlying conditional dependencies are expressed with graphs.

4.2.1 Hidden Markov models (HMMs)

Hidden Markov models (HMMs) [101] are generative probabilistic models, and their underlying structure is a Markov process which is a temporal model. An HMM consists of five components: hidden states (z_i), observations (x), a transition model ($A = a_{z_i z_j}$ where z_i and z_j are hidden states), an observation model ($B = b_{z_i}(x)$ where z_i is a hidden state and x is an observation) and an initial state distribution (Π_i). States that can not be directly observed are called *hidden states* in the structure. The model that defines the transition likelihood between the states is denoted as a transition model, while the probability of observing an observation in a distinct state is called an observation model. The belief in the initial state of the model is represented with the initial state distribution. Figure 4.2 illustrates a binary-state HMM with some observations at the bottom. In the figure, z_0 and z_1 are hidden variables and x_0, x_1 and x_2 are the observations that the model may perceive. a_{ij} corresponds to the probabilities in the transition model, and b_{ij} corresponds to the probabilities in the observation model (the names of the hidden states are omitted in the subscripts in the figure for clarity).

Figure 4.3 illustrates a general HMM model in a block-stacking domain that is presented earlier in Figure 1.1. In the figure, the upper sequence corresponds to the hidden states, where the lower part illustrates the observations that the model emits at each time step. As time proceeds, the model takes perceptions from the environment that corresponds to the states of the blocks. Considering these observations (x_i), hidden states (z_i) are inferred.

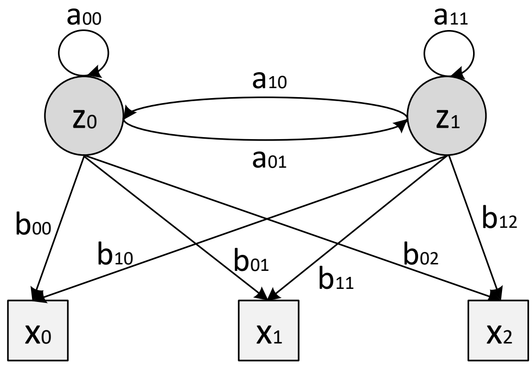


Figure 4.2 : A binary-state HMM with three observations [2].

4.2.2 Conditional random fields (CRFs)

Conditional Random Fields (CRFs) are [102] undirected graphical models to label data sequences. CRFs are discriminative models as Support Vector Machines (SVMs) where posterior probabilities are modeled instead of conditional probability distribution functions and prior probabilities (unlike Hidden Markov Models (HMMs) or Naive Bayes Classifiers). Therefore, they don't model underlying probability distributions.

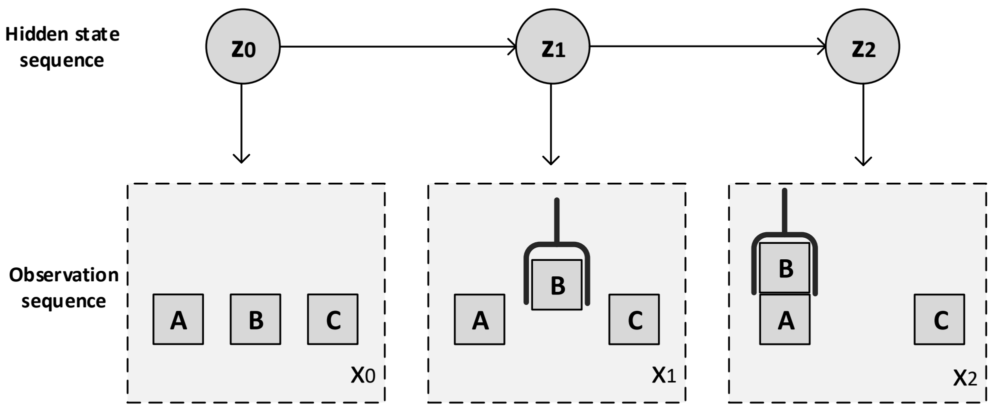


Figure 4.3 : An example sequence for an HMM for the block stacking scenario.

A CRF is represented by feature functions and their corresponding weights, and in order to label a sequence, the normalized product of these feature functions is used. Due to being an indicator function, each feature function can take a value of one or zero. Each feature function also has a weight denoted with λ. Considering other generative models, CRFs can have more relaxed assumptions between the hidden states and the observations and can have an arbitrary number of feature functions in the model. A feature function can be defined as in equation 4.1.

$$\mathrm{f_j}(z_{i-1}, z_i, x_i, S, i) = \begin{cases} 1, & \textit{if}\; z_{i-1} = z_n \quad \wedge \\ & z_i = z_k \quad \wedge \quad x_i = x_m \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

In the formulation, j stands for the index of the feature function, z_i denotes the hidden state that is not directly observable, S is for the whole observation sequence, and i is the index of the state in the sequence. The observations that the sequence gathers for each state are denoted with x_i. Therefore, this feature function represents the case where the current state $z_i = z_k$, the previous state's label is $z_{i-1} = z_n$ (where z_k and z_n are elements of the hidden state set), and x_m is an element of the observation set. The aim is to label each hidden variable with a class. An example sequence for the block-stacking scenario with corresponding feature functions is illustrated in Figure 4.4. Note that each feature function is given to be associated with a single observation and state for simplicity in in Figure 4.4.

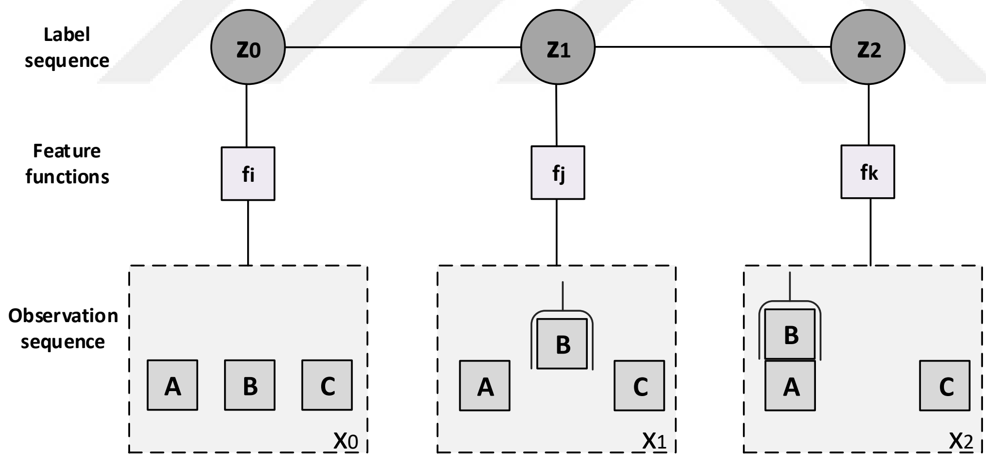


Figure 4.4 : An example sequence with feature functions for the block-stacking scenario.

The feature functions are used in the model together with their weights to maintain a probability distribution; thus, each feature function has its own weight, denoted by λ_j. Given a sequence, the probability of a state is defined as given in equation 4.2.

$$P(z|x) = \frac{exp[\sum_{j=1}^{m} \lambda_j \mathrm{f_j}(z_{i-1}, z_i, x_i, S, i)]}{\sum_{z'} exp[\sum_{j=1}^{m} \lambda_j \mathrm{f_j}(z'_{i-1}, z'_i, x_i, S, i)]} \tag{4.2}$$

In equation 4.2, the nominator part takes the score for the specific state into account, considering the feature functions and their weights. On the other hand, the denominator is used for normalization where z' stands for each hidden state.

5. SYMBOL-BASED ANOMALY IDENTIFICATION FRAMEWORK

This chapter presents a multimodal symbol-based anomaly identification framework that takes into account visual, auditory and proprioceptive features as symbols to address the anomaly identification problem. The proposed framework combines and fuses multimodal sensory data to learn anomaly indicators. Before explaining the proposed framework, the underlying procedures for detecting anomalies are presented since it is essential to present the low-level procedures to clarify the anomaly identification algorithms that contribute to the solution of the corresponding problem. Then, a detailed analysis of the proposed framework is presented including comparisons with baselines and sensor modality analyses. Finally, the chapter is concluded with discussions and remarks about the proposed framework.

5.1 Low-level Processes to Detect Anomalies

Anomalies first need to be detected before they are identified. The anomaly detection procedure necessitates a continuous monitoring process that runs onboard. A classical anomaly detection procedure takes into account online readings that are gathered through multimodal sensory processing. These readings are needed to be fused and interpreted to decide on an anomaly. One should note that the detection of anomalies is out of the scope of this thesis, and an existing HMM-based anomaly detection procedure [103, 104] is employed. This method runs on the fused observation data to detect anomalies. In this study, visual (s_v), auditory (s_a), and proprioceptive (s_p) sensors are considered each of which provides a distinct modality to the robot. The following subsections explain these modalities that contribute to both detection and identification algorithms.

5.1.1 Audition

The auditory data provided by the microphone placed on the lower torso of the Baxter robot are used to infer about the audition-based events that occur in the environment.

The auditory data are perceived through the microphone during execution with 16 KHz sampling rate. In order to construct an anomaly model, sound data are first converted to the frequency domain after dividing the received signal into 32 ms frames. After then, the Mel-frequency Cepstral Coefficients (MFCC) algorithm [105] is applied to detect sound information and its duration. Finally, fragments in the data are detected. The auditory event is classified as a distinct symbolic class by using with a Support Vector Machine (SVM) considering these fragments [104, 106]. In this study, three classes are considered: *no-sound, drop*, and *ego-noise*. The cases where a sound information is not perceived is denoted with the *no-sound* class label. The sound of the fall of an object on the table is represented with the *drop* class. The *ego-noise* class represents the motion of the robot's arm and the sound of the robot itself.

Figure 5.1 presents spectrogram and waveform plots of drop events of three different objects as sub-plots. Each subplot on top illustrates the spectrogram of a drop event where the bottom sub-plots present waveforms for these corresponding objects. The first sub-plots on each line belong to the fall of a pasta box while being pushed, the second ones present the fall of a plastic bunch of grapes while being placed into a container, and the last ones present the fall of a cube while being stacked on top of another cube. As can be observed from the plots, the fall event of the plastic object ends up with a rolling motion of the corresponding object which yields to a broad signal. On the other hand, the fall events of cubes and boxes seem to be more instant resulting in such characteristics as in the first and third 4-second intervals of the plots.

5.1.2 Vision

The RGB-D camera that is placed on the head of the Baxter robot provides visual data. Violet [107], which is a scene interpretation system, is used to continuously extract/maintain an up-to-date world model. Conclusions maintained from different vision sources are combined with fusion to construct a consistent world model. World symbols produced by Violet are used.

The first visual source that provides data to Violet is the LINE-MOD [108] algorithm. The algorithm has the ability to handle recognizing textureless objects, and it can deal with small image transformations. The method adopts a template matching technique by considering a small subset of possible pixel regions during the parsing phase. Since

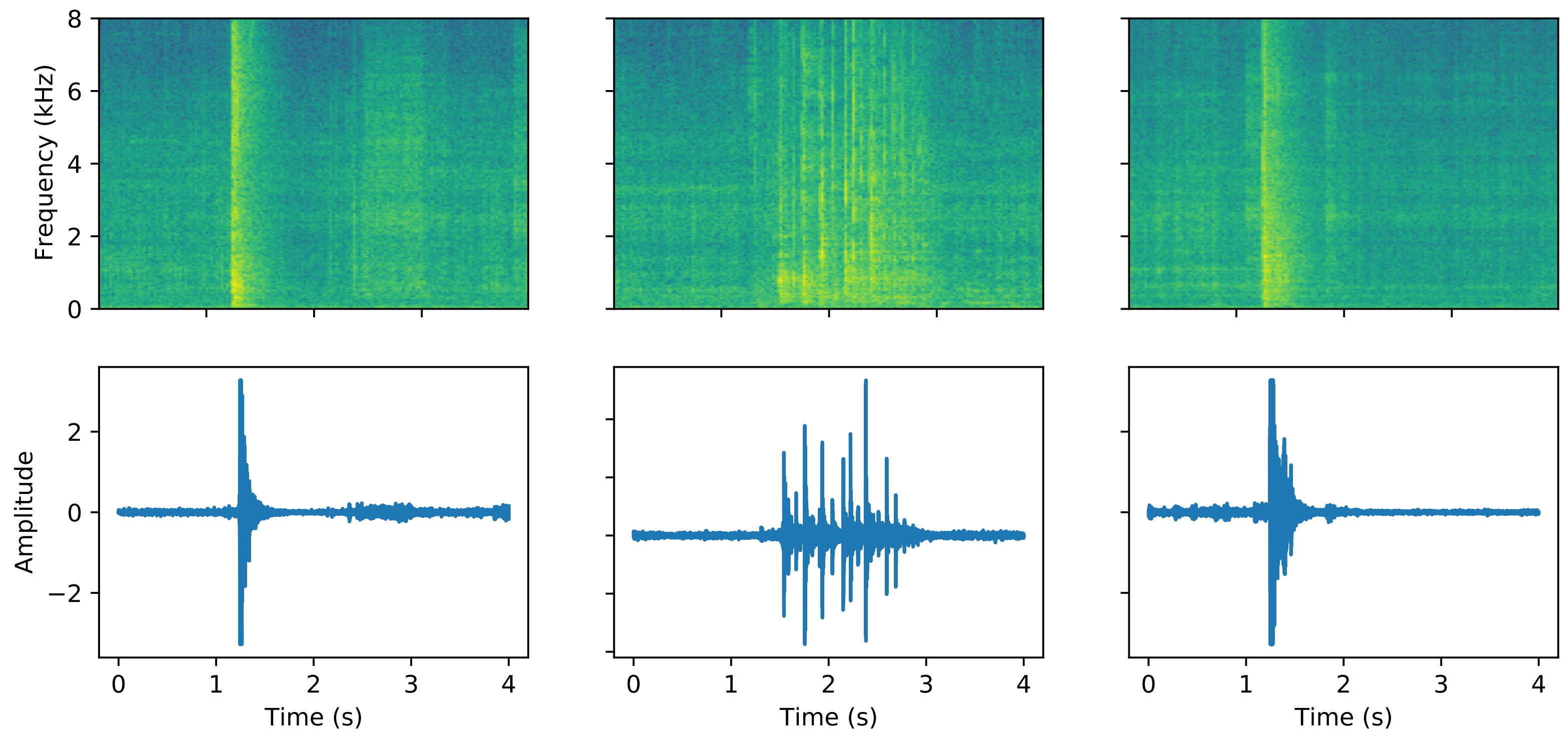


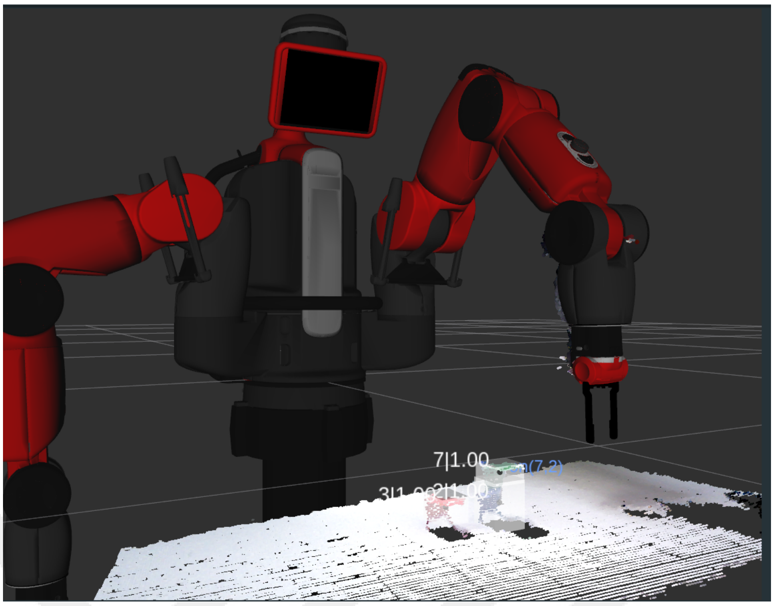
Figure 5.1 : Spectrogram (top) and waveform (bottom) analyses of three drop events of distinct objects.

the number of registered templates does not necessarily be high, the method provides low complexity while recognizing textureless objects. Depth information is also taken into account in the object recognition task.

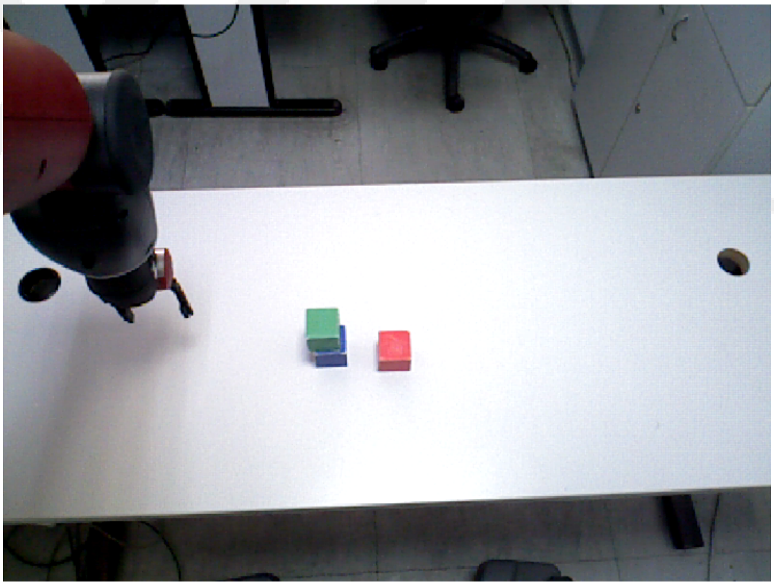
Violet processes LINE-MOD&HS [109], an extension of the LINE-MOD algorithm, as its second source. LINE-MOD-HS takes into account color histograms to recognize textureless objects. Object recognition task in Violet is achieved by template matching on the data with the data provided by the RGB-D camera that provides depth information along with color information for the objects. Object templates are first registered to the KB of the robot before execution. During the execution, point clouds gathered online from the scene are compared with the templates previously registered in the KB. Point clouds that are matched with a template in the KB are recognized with a similarity ratio.

A depth-based segmentation algorithm as the third source of Violet also extracts segments (point cloud clusters) in the scene. Organized Point Cloud Segmentation algorithm [110] is used to detect clusters in the scene by processing the RGB-D data. Violet discards larger or smaller segments that exceed a predefined threshold. One should note that the segmentation algorithm does not provide object type information as LINE-MOD or LINE-MOD&HS, but point cloud clusters in the scene. All this information provided by different sources is maintained and fused to end up with a consistent world model by Violet. Violet applies Bayesian fusion to provide such conclusions, and it continuously updates the KB of the robot to create object models and deal with object permanence issues in case of a change in the scene.

Figure 5.2 presents images taken from rviz which is a 3D visualizer implemented in ROS. It can be used for illustrating the robot's model and sensory data gathered through the sensors. The figure shows an intermediate step of a block-stacking scenario where the robot approaches a red block to pick and place it on top of a green block that is already stacked on top of a blue block. Figure 5.2(a) presents an illustration of the robot's belief in the corresponding world state (KB) constructed by Violet. The KB is constructed by considering the outputs of the explained vision algorithms. The numbers on each object in the figure denote the object IDs and beliefs on the existence of the corresponding objects, respectively. Figure 5.2(b) presents the image taken from the RGB-D camera placed on the head of the robot for this intermediate step of the scenario.



(a)



(b)

Figure 5.2 : (a) Illustration of the KB of the robot taken from rviz on an intermediate step of a tower construction scenario. (b) Corresponding image of the rviz illustration taken from the RGB-D camera.

5.1.3 Proprioception

The robot possesses a gripper attached to its left hand that provides proprioceptive information during execution. The distance between the gripper fingers is processed and used in this study as a proprioceptive modality.

5.2 Symbols as Features for Anomaly Identification

An anomaly identification framework that processes symbols as features to capture anomaly indicators is presented in this chapter. This symbol-based anomaly identification framework takes into account various features provided by the sensory processing system presented earlier. In this study, three types of sensory modalities gathered through distinct sensors (visual, auditory, and proprioceptive) and auxiliary data are considered to address the described problem. The outputs of Violet for the high-level visual predicates, the auditory classes provided by the method explained in Section 5.1.1, gripper-related data, and auxiliary data (force on the gripper, distance gathered from the laser mounted on the gripper, and action execution information) are used to construct a feature vector. Figure 5.3 illustrates the features that are processed by the identification process. The following items explain each of these features in detail.

- *Laser Distance:* This feature presents the distance information between the gripper and the object to be manipulated. The laser mounted on the hand of the Baxter robot provides this information continuously during execution.

- *Gripper Pose:* This feature presents the state of the gripper of the robot. The information gathered from the gripper pose is used after discretization of the raw data. Two symbolic classes are maintained: *open* or *closed*. Considering the range of the gripper as 8 cm, *open* is used for the cases where the distance between the gripper pedals is greater than 5 cm where *closed* is used for the opposite case (distance is less than 5 cm). The robot also needs to adjust its gripper to grasp an object according to the size of the object at hand. When there is an object is grasped by the robot, the state of the gripper is considered *closed*.

- *Gripper Force:* This feature presents the force value at the gripper. The scale for the measured force is 0 to 35 N. Whenever force is detected on the gripper, its corresponding value is processed.

- *Sound:* This feature presents the auditory data gathered during execution. The auditory data gathered through the microphone of the robot are classified considering three classes (*no-sound, drop*, and *ego-noise*) with the method explained in Section 5.1.1.

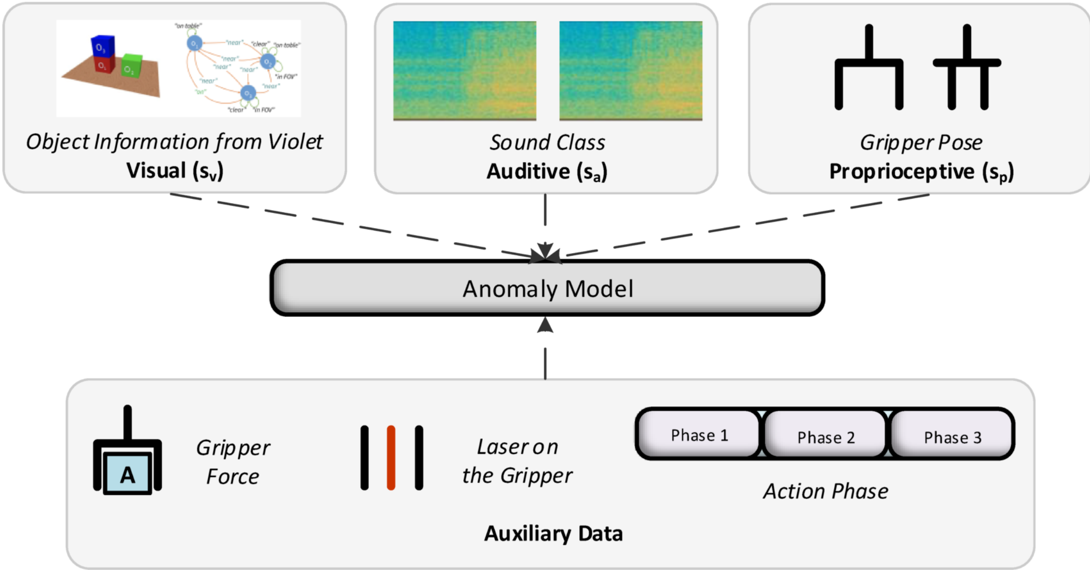


Figure 5.3 : The processed data by the symbol-based anomaly identification framework [1].

- *Object Information:* This feature presents the object-related information in the KB provided by Violet. Location-related information (i.e., changes in the object locations), object existence information, and spatial relations among objects (i.e., the offset of the objects when there is an *on* relation) are presented by this feature. There are three symbolic classes for the existence of an object: *yes, no*, and *unknown*. The class *yes* corresponds to the cases where an object is recognized by the vision algorithms described in Section 5.1.2. On the other hand, *no* stands for the cases where the object is not recognized anymore. The *unknown* class for existence stands for the cases where the visual updates for the corresponding object are suspended. Such cases may arise due to an occlusion of the visual input by the robot's arm.

- *Action Phase:* Each action has a number of different phases. This feature presents the phase of an action that is being executed by the robot. For example, the *pick-up* action is composed of five consecutive phases: *pick-started, pick-approaching, gripper-open, gripper-close* and *pick-end*. The *push* has four phases: *push-started, push-approaching, push-executing* and *push-ended*. The *put-down* action has six phases: *place-started, place-moving, gripper-open, arm-retract, gripper-close* and *place-ended*. The actions *move-to-object* and *move-to-location* include similar four discrete phases: *move-started, move-planning, move-executing* and *move-ended*.

5.3 Methods for Anomaly Identification

In this section, first, the proposed anomaly identification method is explained. Then, the selected baseline methods are presented in detail.

5.3.1 The LSTM-based anomaly identification algorithm

The idea behind an LSTM is straightforward. An LSTM cell is a unit that makes the information flow from the entrance through the exit by applying necessary modifications to it. The LSTM structure modifies this cell to add or remove information. In order to do so, first, unnecessary information is removed. This is followed by adding new information and updating the previous memory cell. Finally, the output is produced. The equations that summarize these operations for an LSTM are given below [92].

$$f_t = \sigma(W_f z_{t-1} + U_f x_t + b_f) \tag{5.1}$$

$$i_t = \sigma(W_i z_{t-1} + U_i x_t + b_i) \tag{5.2}$$

$$\widetilde{c_t} = tanh(W_c z_{t-1} + U_c x_t + b_c) \tag{5.3}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \widetilde{c_t} \tag{5.4}$$

$$o_t = \sigma(W_o z_{t-1} + U_o x_t + b_o) \tag{5.5}$$

$$z_t = o_t \odot tanh(c_t) \tag{5.6}$$

In these equations, z_t corresponds to the hidden variables at time t, x is for input variables (observations in the described problem), b is for bias for the corresponding variables, W and U denote weights for the hidden states and inputs, respectively. c denotes the selected input variables. $\odot$ is element-wise multiplication. In order to decide what is stored and forgotten in the cell, a sigmoid function (σ) that gives the degree of the forget operation is implemented. In case of a higher value that the function returns, more information is stored in the cell from the previous state (Equation 5.1). After deciding the degree of removal for the cell, the information that is going to be stored in it should be decided. This is achieved by selecting the variables (Equation 5.2) and constructing new candidates for the update (Equation 5.3) with sigmoid and tangent functions, respectively. Afterward, the cell state is updated (Equation 5.4), and the output is produced (Equations 5.5 - 5.6).

The presented anomaly identification algorithm uses LSTMs as underlying models to learn anomaly indicators. The observation sequences as observation histories are taken into account to construct a model to classify anomaly types whenever such a case is detected. The offline collected data are used to train the model. After distinct sensor modalities are combined and fused, they are sequentially fed to LSTM cells. Then, the gathered result from the LSTM layers is transferred to a fully connected layer. This step is followed by the calculation of the likelihoods of having the indicators of an anomaly type for each observation. As a final step, majority voting is applied to the labeled sequence to decide on the failure type. This step concludes the training phase by calculating loss values and updating weights accordingly.

The offline trained anomaly model is used during run-time to classify anomaly types. When the robot is tasked with a goal, it starts to execute its actions sequentially to fulfill the goal. During the execution, observations are gathered through the sensors that the robot possesses and these observations are fused and stored in an observation history. Meanwhile, task execution is monitored with the anomaly detection algorithm explained in Section 5.1. Unless an anomaly occurs, the robot continues the task execution and collects sensor readings. In case of an anomaly, the fused observation history is sent to the trained anomaly model to end up with a labeled sequence. Finally, majority voting is applied to this labeled sequence to find out the anomaly type. Figure 5.4 illustrates these sequential phases.

Algorithm 1 presents the overview of the symbol-based anomaly identification algorithm. It accepts the trained anomaly model (M), the task ($task$), the anomaly set A and the window size ws for fusion as inputs. As an output, it produces the anomaly class ($type$). First, the observation sequence (H) is initialized as an empty list. Then, the robot starts to execute a sequence of actions. During the execution, the robot observes the environment through its sensors (Line 3, *senseEnvironment()* procedure). An observation gathered from the sensor that provide visual modality s_v (the subscript corresponds to the abbreviation for the type of the particular modality) at time t is denoted with $o_t^{s_v}$. Similarly, $o_t^{s_a}$ and $o_t^{s_p}$ denote the observations gathered from the auditory and proprioceptive sensors, respectively.

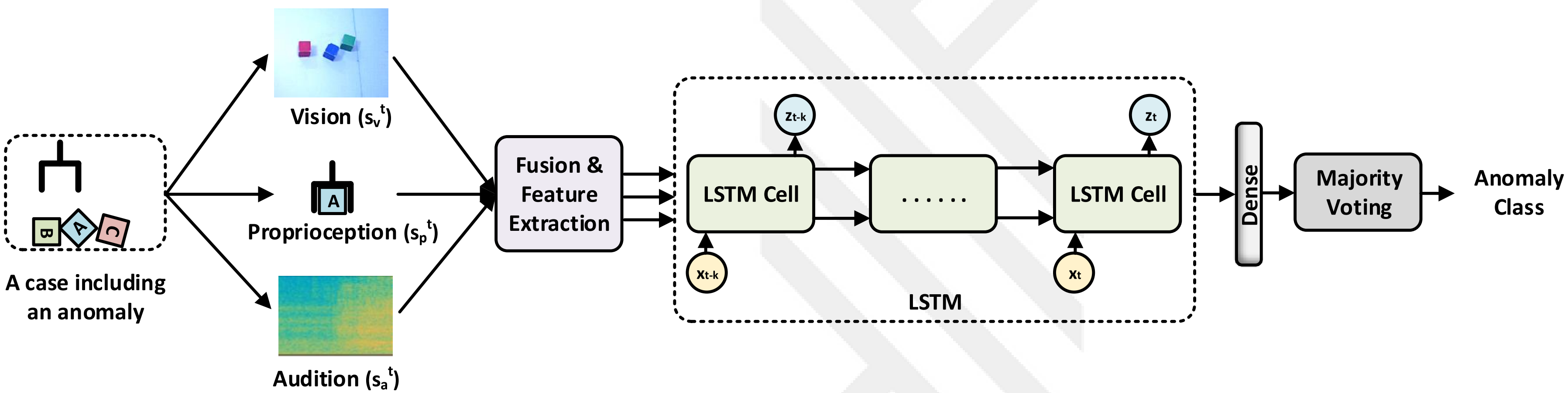


Figure 5.4 : An illustration of the proposed symbol-based anomaly identification framework [1].

Algorithm 1: Symbol-based Anomaly Identification Algorithm

Data: Trained anomaly model M, Task $task$, Anomaly set A, window size ws
Result: Anomaly class $type$

1 $H = []$;
2 **while** *An anomaly is not detected and task is not over* **do**
3     $o_t^{s_v}, o_t^{s_a}, o_t^{s_p}$ = *senseEnvironment()*;
4     $o_t^{s_v}$.*interpretScene()*;
5     $o_t^{s_a}$.*classifySound()*;
6     $o_t^{s_p}$.*gatherGripperData()*;
7     $x_t = (o_t^{s_v}, o_t^{s_a}, o_t^{s_p})$;
8   H.*append*(x_t)
9 H.*earlyMultimodalFusion(ws)*;
10 **if** *An anomaly is detected* **then**
11     $z_{0:t}$ = *identifyAnomaly(M, H)*;
12 **else**
13     $z_{0:t}$ = *safe*;
14   **return** $safe$
15 $type$ = *majorityVoting*($z_{0:t}$, A);
16 **return** $type$

Violet [107] provides symbolic representation of the world model inferred from the fusion of the RGB and RGB-D data. Violet continually processes these data to construct a consistent world model and provides the list of objects in the scene and their locations, colors, and the spatial relations among them (Line 4, *interpretScene()* procedure). The microphone also continually provides data, and the robot classifies gathered sounds during run-time with a trained auditory model explained in Section 5.1.1 (Line 5, *classifySound()* procedure). Perceiving gripper-based data completes the sense cycle of the robot. The force measurements taken from the gripper and infrared laser's data that is mounted on the gripper to measure the distance from an object are also processed as auxiliary data (Line 6, *gatherGripperData()* procedure). After the robot collects these data, there is a need for a fusion procedure (Line 9, *earlyMultimodalFusion()* procedure). In case of an anomaly, the observation sequence is labelled (Line 11, *identifyAnomaly()* procedure). As a result, an overall decision on the occurred anomaly type is required. Consequently, a majority voting procedure is implemented (Line 15, *majorityVoting()* procedure). The upcoming sections present the details of these subroutines that are invoked in the presented algorithm.

5.3.1.1 Early multimodal fusion

A sensor fusion procedure that combines the provided sensory data is essential since sensors may provide asynchronous data. For the symbol-based solution, an early fusion procedure is implemented. Since the main necessity for a fusion procedure arises due to the sensors working asynchronous, the early fusion procedure considers the timestamps of the observations gathered from each sensor. Therefore, all sensory modality information is combined and fused in a temporal manner to come up with a feature vector for classification. The data provided by Violet, the auditory data, and the gripper data are fused to obtain fused observations.

The early fusion algorithm considers visual observations to be complemented with the observations provided by the other sensors. This is achieved by considering the minimum temporal difference between the corresponding visual observation and the other observations provided by other sensors. As a result, a visual observation which is complemented with an auditory class and tactile information is obtained. Temporal distances between the timestamp of visual observation and the observations provided by the other sensors are calculated for a constant window size (ws). In this study, the window size is set to 2 seconds. The proposed early fusion procedure terminates when all visual observations are complemented with auditory and proprioceptive data.

Algorithm 2 presents the overview of the proposed early fusion algorithm. The algorithm accepts the observation history (H) that contains sequential data perceived through different sensors as inputs. Sequences of three sensory modalities are processed by the algorithm: visual (o^{s_v}), auditory (o^{s_a}) and proprioceptive (o^{s_p}), respectively. The window size is also provided as a parameter to fuse the data. The algorithm terminates by returning the fused observation sequence. Since the auditory modality provides the most sparse data, it complements the visual data. For each auditory observation, this observation is diffused to the visual observation with a window size of ws seconds. Then, the most temporally closest proprioceptive observations timestamp is calculated (Line 4). If the visual observation is already complemented with a sound class, then predefined priorities of the sound classes are taken into account. Priorities are assigned manually to the auditory classes to avoid overriding of one class by another class with relatively less priority during fusion. The *drop* class has the highest priority, and the *ego-noise* class immediately follows it. If a sound class has a higher priority, the fusion procedure considers that particular class. As a final step, this visual observation is complemented with corresponding auditory and proprioceptive reading. After this procedure, all fused observation sequence is checked for the observations that include no auditory data (Line 8). In such a case, a *no-sound* label is assigned to these particular observations. This operation guarantees that all observations in the fused sequence include all three sensory modalities, and the fused observation sequence ($x_{0:t}$) is returned. Note that this early fusion algorithm is executed offline after the anomaly detection algorithm reports an anomaly case.

Algorithm 2: Early multimodal fusion

Data: Observation history H, window size ws
Result: Fused observation sequence $x_{0:t}$

1 **for** *each* $o_i^{s_a}$ *in* H **do**
2   **for** $j = (i - ws)$ *to* $(i + ws)$ **do**
3   **if** $(o_j^{s_v}$ *is not matched) or* $(o_i^{s_a}.priority > o_j^{s_a}.priority)$ **then**
4             $k = getClosestProprioceptiveObservation(o_{0:t}^{s_p}, j)$;
5             $x_j = (o_j^{s_v}, o_i^{s_a}, o_k^{s_p})$;

6 **for** *each* x_i *in* H **do**
7   **if** $o_i^{s_v}$ *is not matched* **then**
8         $k = getClosestProprioceptiveObservation(o_{0:t}^{s_p}, i)$;
9         $x_i = (o_i^{s_v}, no\text{-}sound, o_k^{s_p})$;

10 **return** $x_{0:t}$

Figure 5.5 illustrates the explained early fusion scheme for a given auditory and visual observation sequence for a window size of 1. Note that proprioception is omitted in this figure since audition provides sparse data (i.e., distinct auditory classes provided by the classifier) compared to proprioception in this study. Consequently, it is straightforward to match the proprioceptive modality with the visual modality. Non-fused observations are shown on top of the figure, and the resulting observation sequence is shown on the bottom of the figure. The auditory observation sequence includes three observations: *drop*, *ego-noise* and *drop* perceived at time steps 1, 2 and 6, respectively. These sparse observations need to be fused with the visual observation sequence to complement them. In this particular example, window size (ws) is set to 1 for brevity. Therefore, each auditory observation is propagated consistently with the window size. For instance, *drop* is received at time step 1, and it is spread in the fused observation between time steps 0 and 2 since the window size is 1. The same procedure is applied to the other auditory observations similarly. Note that *ego-noise* is overwritten by *drop* at time steps 1 and 2 as the priority of *drop* is higher than that of *ego-noise*. As the last step, the visual observations without an auditory observation is complemented with *no-sound*. The fused observation sequence includes a visual-auditory observation pair for each time step.

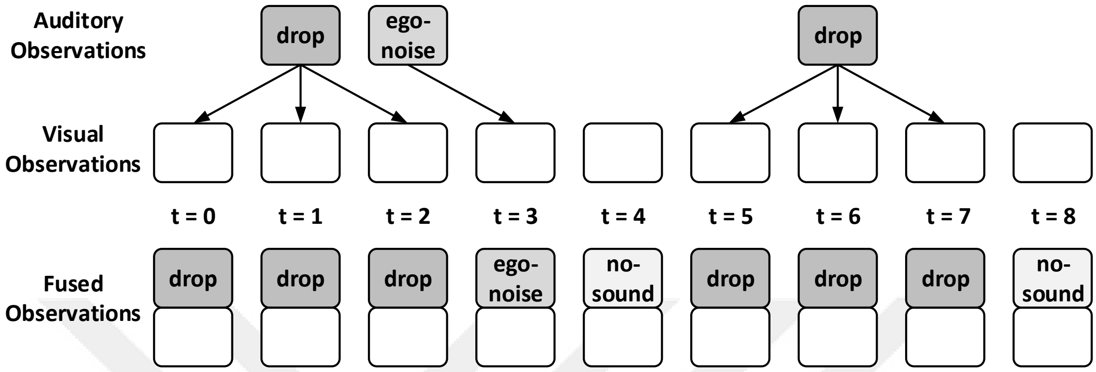


Figure 5.5 : The early fusion scheme for auditory data (*ws* = 1).

5.3.1.2 Majority voting

The task execution stops either in case of an anomaly or the task is fulfilled. If an anomaly is detected, the observation history H and the anomaly model M are taken into account to identify the anomaly. The fused observation sequence that includes the extracted symbolic features is fed into the trained model. This results in each observation in the sequence being labeled with one of the anomaly classes or with the label *safe*. At the end of this procedure, a labeled sequence is formed.

To come up with a consensus on the decision among the labeled observations, a majority voting approach is adopted. Each observation's vote is uniformly weighted, and the overall score for each anomaly type is evaluated by considering the summation of the votes of the observations (Line 3). The anomaly type with the highest vote is considered to be of the anomaly class (Line 4). Algorithm 3 summarizes the used majority voting technique.

5.3.2 The HMM-based anomaly identification algorithm

The anomaly identification problem is also addressed by modeling anomaly types with HMMs. The problem representation includes binary hidden states for each anomaly type: *safe* and *anomaly*. The *safe* hidden state denotes the cases where no anomaly

Algorithm 3: Majority voting

Data: Label sequence $z_{0:t}$, Anomaly set A

Result: Anomaly class $type$

$v_{0:n} = 0$;   `// n is the cardinality of the anomaly set A`

for *each* z_i *in* $z_{0:t}$ **do**

  $v.vote(z_i)$;

$j = argmax(v_{0:n})$;

$type = A_j$;

return $type$

exists. On the other hand, the *anomaly* hidden state corresponds to the cases where the state includes an anomaly. Since each anomaly type is represented by a distinct model, this *anomaly* state depends on the represented model's type. Before the execution, a distinct anomaly model is trained for each anomaly type with the corresponding execution sequences. The fused sensory data are the observations of these hidden states. Gaussian HMMs are used in the HMM implementation.

Algorithm 4 presents a summary of the implemented solution to the problem. The algorithm takes the trained HMMs ($M_{0:n}$), the anomaly set A, the task to be executed by the robot and the window size (ws) for fusion as inputs. The robot starts to execute the assigned task with an empty observation history list. While an anomaly is not detected and the task is not over, gathered and fused observations are pushed to the observation history. If the anomaly detection algorithm reports an anomaly, likelihoods are evaluated to find out the model that fits the gathered observation sequence the most. In other words, each model calculates its likelihood to generate the given observation sequence. Afterward, the anomaly model (M_j) with the highest likelihood is assumed to be the anomaly type of the corresponding deviation. Then, the Viterbi algorithm labels the sequence with the selected model. This is followed by the majority voting phase to find out the anomaly type. Since the identification mechanism is activated in case of an anomaly, the algorithm returns the *safe* label only if no anomaly is detected (Line 17). Therefore, the anomaly type is returned at the end of the execution.

5.3.3 The CRF-based anomaly identification algorithm

Anomaly types are modeled with CRFs with the anomaly types as hidden states similar to HMM modeling of anomalies. Before execution, an anomaly model is trained with the collected data including anomaly samples of extracted features. The training phase

Algorithm 4: HMM-based Anomaly Identification Algorithm

Data: Trained anomaly models $M_{0:n}$, Task *task*, Anomaly set A, window size *ws*
Result: Anomaly class *type*

$H = []$;
while *An anomaly is not detected and task is not over* **do**
  $o_t^{s_v}, o_t^{s_a}, o_t^{s_p}$ = *senseEnvironment()*;
  $o_t^{s_v}$*.interpretScene()*;
  $o_t^{s_a}$*.classifySound()*;
  $o_t^{s_p}$*.gatherGripperData()*;
  $x_t = (o_t^{s_v}, o_t^{s_a}, o_t^{s_p})$;
  H*.append*(x_t)
H*.earlyMultimodalFusion(ws)*;
if *An anomaly occurred* **then**
  $p_{0:n} = 0$;
  for *each* M_i *in* $M_{0:n}$ **do**
    p_i = *calculateLikelihood*(H, M_i);
  j = *argmax*($p_{0:n}$);
  $z_{0:t}$ = *labelSequence*(H,M_j);
else
  $z_{0:t}$ = *safe*;
  return *safe*
type = *majorityVoting*($z_{0:t}$, A);
return *type*

provides feature functions related to the anomaly indicators. The observation history is processed during the execution. If an anomaly is detected, the observation history is investigated to reveal the anomaly type of the encountered deviation in a similar manner to the HMM-based algorithm. For the CRF-based anomaly identification algorithms, the following variants of parameter learning procedures are considered: Adaptive Regularization of Weight Vector (AROW) [111] and Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [112]. A library that provides various CRF representations, namely CRFsuite [113], is used for the used training algorithms.

5.4 Experiments

In this section, the experimental setup, the data collection phase, and the evaluation of the presented symbol-based anomaly identification framework are presented.

5.4.1 Experimental setup

Experiments are conducted in a laboratory environment where a selected variety of everyday manipulation objects for kitchens are placed on a table. The robot manipulates the following objects in realized anomaly scenarios: a powder box, a milk bottle, a plastic grape toy, a pasta box, and cubical blocks with different colors and sizes. The Baxter robot presented in Chapter 3 is tasked to manipulate these objects in the environment with the available manipulation actions. Anomaly cases that arise related to object manipulation (i.e., stacking, pushing) and external events (i.e., cooperation with humans or other agents) are investigated and analyzed. These scenarios and actions are elaborated in Chapter 3.

5.4.2 Data collection and annotation

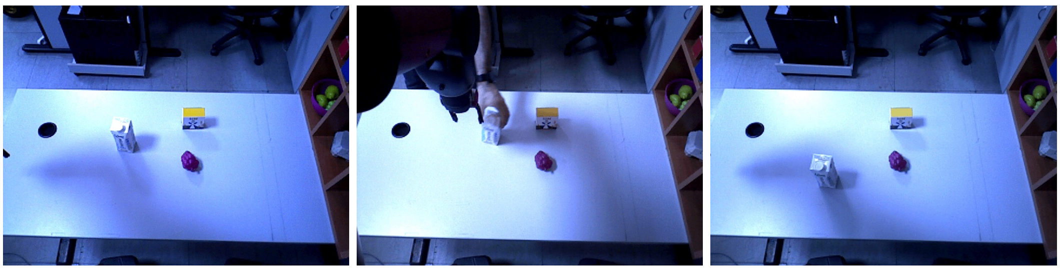
132 object manipulation scenarios that include four distinct anomaly types (LOC, DIS, EUA, and OTA) with the Baxter robot are used to evaluate the symbol-based anomaly identification framework. Objects are initially located at random locations with random configurations at each execution. The tasks are randomly assigned to the robot by a human operator. Some of these tasks include constructing a tower from a set of objects, pushing a target object or tower, etc. A set of actions is also provided to the robot to fulfill the task. While the robot executes the task, anomalies arise either without any intervention (i.e., fall of objects) or with human intervention (i.e., exogenous events such as object disappearance). These anomaly types, actions, and cases are presented in Chapter 3 in detail.

The distribution of anomaly types over the scenarios are given as follows: 49 for DIS (the target object disappears from the environment by the manipulation of a human or another robot), 39 for EUA (a tower collapses due to previously unbalanced stacking of objects), 23 for LOC (an object is displaced without the knowledge of the robot), and 21 for OTA (the object or tower to be manipulated overturns). It should be noted that there exist safe scenarios where the objects are stacked with an offset in the tower construction scenarios, however, the tower does not collapse. In case of an anomaly, the robot locates its arm to the home position, and the execution terminates. The observation sampling frequency is at 10 Hz.

The collected data set is annotated to indicate anomaly types manually before training. Each anomaly scenario is labeled with its corresponding anomaly type and its time of occurrence timestamp. The data set is randomly partitioned into train and test sets. 80% of the data set is considered as a train set, and the remaining 20% is used for testing the performance of the presented framework.

5.4.3 A temporal analysis of an anomaly case

Before presenting the experimental evaluation, a sample anomaly case is illustrated to analyze how the sensory data temporally change during execution before the anomaly is detected. The snapshots of this execution sequence from the robot's RGB-D camera are shown in Figure 5.6. After the robot detects the objects in the environment, it is tasked with the manipulation of the milk bottle. Initially, the location of the object is determined by Violet. Then, the robot plans a trajectory to grasp the object. While the robot approaches the object, a human intervenes in the scene and changes the location of the object before the robot grasps it. This deviation from expectations leads to an anomaly detected by the robot. After detection, the robot retracts its arm to the initial configuration. It then perceives the object at its new location. This event is followed by the identification procedure to reveal the related anomaly class by investigating the observation sequence. This is an example LOC whose details are presented in Chapter 3.



(a) (b) (c)

Figure 5.6 : A sample LOC anomaly case scenario. A human changes the location of the object be manipulated. [1].

Figure 5.7 presents the interpretation of the perceived observation history for an anomaly case of *LOC* that is illustrated in Figure 5.6. The figure has six plots, and each of them illustrates a different sensor data or interpretation. For all plots, x-axes denote the time in seconds, and y-axes correspond to the corresponding feature of that type. The first graph shows the interpretation of the force in the gripper of the

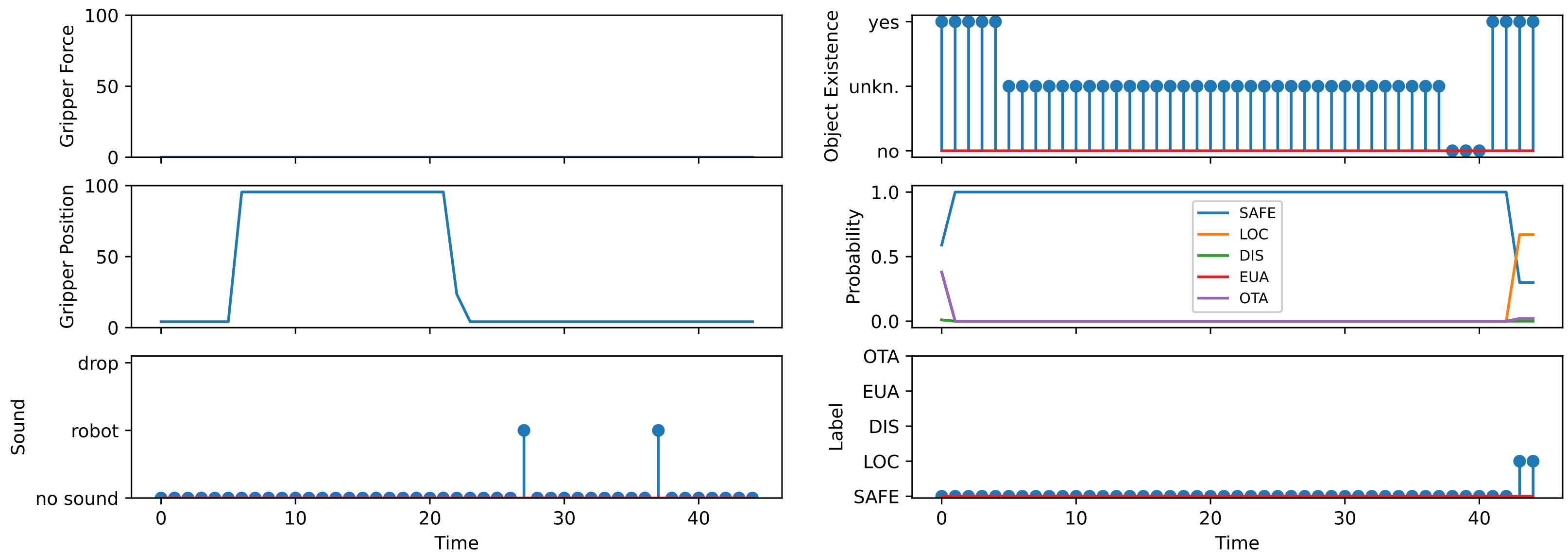


Figure 5.7 : Sensor values, their interpretation, and the illustration of the robot's belief on each anomaly type during the change in the object's location.

robot. When the gripper is empty, in other words, an object is not being held, the value is zero. It provides a value greater than zero when there is an object in the gripper. The position of the gripper pedals is shown in the second graph. When the pedals are closed, the value is approximately zero. When there is an object at hand, the interval of the value changes between 0 and 100 depending on the object's width. The classified sound information is presented in the next plot. As can be observed from this plot that the robot perceives sound related to its own movement at time steps 27 and 37. This plot is followed by the graph illustrating the belief on the existence of an object in the KB. Note that this existence information is gathered from Violet and there are three symbolic classes: *yes*, *no* and *unknown*. For brevity, the updates of only a single object are presented. The value *yes* denotes the situation where Violet recognizes the object, the value *no* corresponds to the non-existence of the object. The visual updates are suspended due to the occlusion caused by the movement of the robot's arm. The value *unknown* denotes such cases. The robot's belief on an anomaly case for each class as normalized probabilities is presented on the next plot. Based on the gathered observations, probabilities of the classes of anomalies are updated at each time step. The latent label of the state is presented on the last graph. This is the result of the presented algorithm for the given test case.

In this scenario, the robot first perceives the objects in the environment. Then, the task execution starts with the first action in the plan. The robot opens its gripper to grasp the object, and a peak in the gripper position is observed in the corresponding plot. As long as the robot's arm occludes the field of view of the robot, the robot suspends the visual updates related to the objects during that period. Consequently, the value *unknown* is set to the belief in the existence of the object due to the open-world assumption (OWA). Since the object is relocated to another location on the table, the robot fails to use its initial trajectory. The robot retracts its arm to its initial location after the anomaly is detected. After Violet considers no more occlusion of the arm, the robot updates the changes on the corresponding object's location between 38-40 seconds after which the anomaly classification takes place.

5.4.4 Performance evaluation

This section presents the experimental evaluation of the presented symbol-based anomaly identification framework. Five distinct methods are evaluated: LSTM-based, GRU-based, HMM-based, and two CRF-based methods with different parameter learning techniques (AROW and L-BFGS). Note that LSTMs are replaced with GRUs to implement the GRU-based method. For the deep learning-based methods, Adam Algorithm [114] is used as an optimizer, and Cross Entropy Loss is used to evaluate the loss during training. The PyTorch framework is used to implement the models. The following parameters are set for the implemented deep learning algorithms: number of epochs is 500, learning rate (η) is 10^{-3}, number of layers is 2, hidden dimension is 128.

The algorithms are investigated in terms of the following criteria: First, a comparative analysis of the methods in terms of precision, recall and f-score is presented. Then, confusion matrices indicating the state classification performance of the methods are analyzed. Afterward, the accuracy of the case classification task is given with confusion matrices. Finally, multimodality analysis of the selected sensors is presented to show the contribution of each sensory modality on the identification task.

Figure 5.8 presents the change in the loss during the training phase for the LSTM-based anomaly identification method. All collected observations as sequences of previous executions are taken into account with their corresponding labels. After the loss calculation at each epoch, weights are updated accordingly. As the number of epochs increases, the loss has a decreasing behavior according to the plot, and it eventually converges. The plot also indicates the f-score values of the related methods at corresponding epochs. Considering these results in the plot, one can infer that the f-score does not vary approximately after the epoch 300.

5.4.4.1 State classification analysis of the observation sequences

In this experiment, the classification analysis of each gathered observation is presented. To do so, each observation in the test cases is labeled by the robot. The performance of the methods is measured in terms of precision, recall, and f-score metrics.

Table 5.1 lays out the comparative statistical analysis of the methods. The average (μ) and standard deviation (σ) values of 10 executions on randomly partitioned data sets (80% training and 20% testing) are presented. Each column presents the results for the four distinct anomaly classes and the safe class. The last column presents the overall average of the corresponding rows. The *safe* class takes part in the table to analyze false alarms of the anomaly detection algorithm. The three horizontally divided parts correspond to distinct metrics: precision, recall and f-score, respectively. Each separate row within these divided parts present the scores for the corresponding method. The table is extended with a final row indicating the average length of the observation sequences for each anomaly type in terms of the number of observation instances.

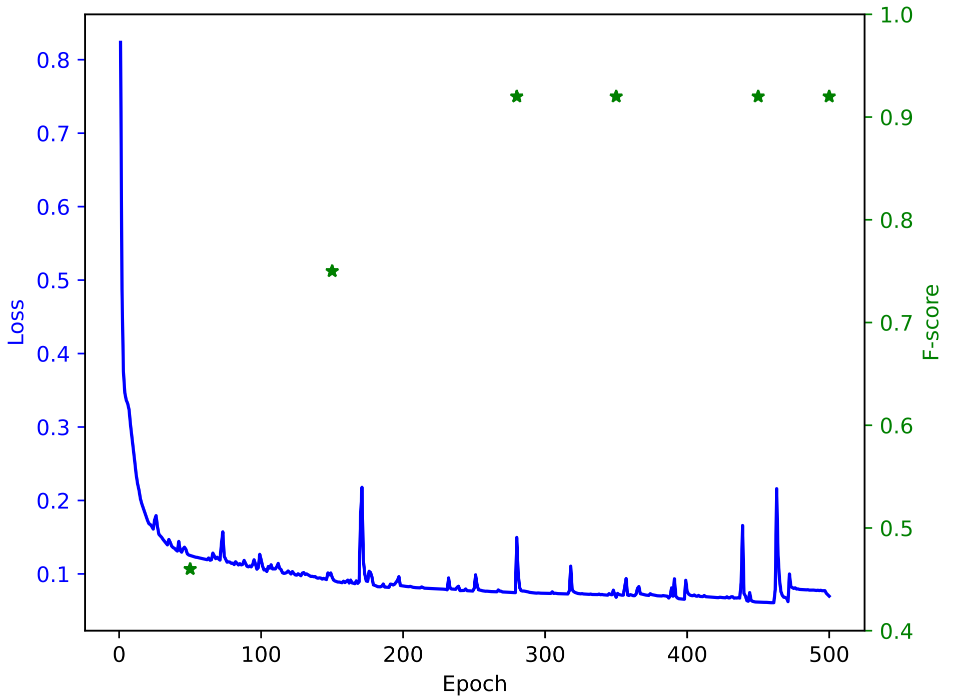


Figure 5.8 : Analysis of the change in loss (blue) and f-score (green) during the training phase of the LSTM-based method. A learning rate (η) of 10^{-3} is set.

The LSTM-based, GRU-based and CRF-based (w/ L-BFGS) methods provide better precision scores compared to the other methods in Table 5.1. The CRF-based (w/ L-BFGS) method outperforms the other CRF-based (w/ AROW) method for all metrics and anomaly types. LOC and EUA cannot be distinguished by the HMM-based method. The LSTM-based algorithm provides better recall and f-score scores compared to the HMM-based and AROW-trained CRF-based methods for all anomaly cases. Although similar results for LOC, DIS and OTA are provided by the LSTM-based, GRU-based and L-BFGS-trained CRF-based methods, the LSTM-based method outperforms the GRU-based and CRF-based (w/ L-BFGS) methods in EUA. This can be explained by the fact that the LSTM-based method has the ability to deal with the propagation of the effects of previously gathered observations. Generally,

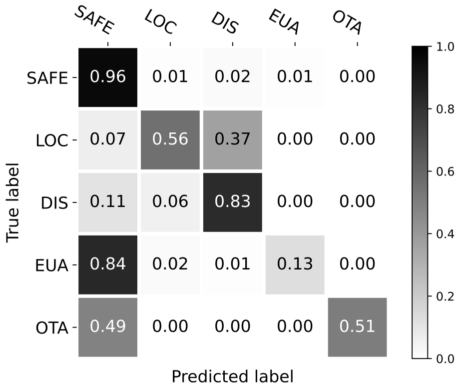
Table 5.1 : Overall performance evaluation and comparison of the anomaly identification algorithms.

		SAFE ($\mu \pm \sigma$)	LOC ($\mu \pm \sigma$)	DIS ($\mu \pm \sigma$)	EUA ($\mu \pm \sigma$)	OTA ($\mu \pm \sigma$)	Overall ($\mu \pm \sigma$)
Precision	HMM	0.83 ± 0.03	0.39 ± 0.27	0.43 ± 0.13	0.92 ± 0.21	1.00 ± 0.00	0.52 ± 0.13
	CRF w/ AROW	0.85 ± 0.04	0.25 ± 0.40	0.89 ± 0.12	0.85 ± 0.19	0.40 ± 0.39	0.65 ± 0.23
	CRF w/ L-BFGS	0.91 ± 0.04	0.99 ± 0.02	0.96 ± 0.07	0.99 ± 0.02	0.99 ± 0.01	$\mathbf{0.97} \pm \mathbf{0.03}$
	GRU	0.91 ± 0.04	0.87 ± 0.17	0.99 ± 0.02	0.98 ± 0.02	0.99 ± 0.02	0.95 ± 0.05
	LSTM	0.94 ± 0.04	0.96 ± 0.07	0.98 ± 0.02	0.93 ± 0.07	0.99 ± 0.01	0.96 ± 0.04
Recall	HMM	0.96 ± 0.02	0.59 ± 0.24	0.83 ± 0.15	0.14 ± 0.05	0.52 ± 0.06	0.61 ± 0.10
	CRF w/ AROW	0.99 ± 0.01	0.08 ± 0.12	0.85 ± 0.14	0.28 ± 0.16	0.49 ± 0.49	0.55 ± 0.18
	CRF w/ L-BFGS	0.99 ± 0.01	0.91 ± 0.19	0.87 ± 0.14	0.58 ± 0.15	0.98 ± 0.02	0.87 ± 0.10
	GRU	0.99 ± 0.01	0.85 ± 0.15	0.87 ± 0.14	0.58 ± 0.16	0.99 ± 0.02	0.86 ± 0.09
	LSTM	0.99 ± 0.02	0.88 ± 0.18	0.86 ± 0.15	0.74 ± 0.16	0.99 ± 0.01	$\mathbf{0.89} \pm \mathbf{0.10}$
F-score	HMM	0.89 ± 0.02	0.40 ± 0.19	0.55 ± 0.11	0.23 ± 0.07	0.68 ± 0.05	0.56 ± 0.23
	CRF w/ AROW	0.92 ± 0.02	0.12 ± 0.18	0.86 ± 0.11	0.40 ± 0.21	0.43 ± 0.43	0.55 ± 0.19
	CRF w/ L-BFGS	0.95 ± 0.02	0.93 ± 0.15	0.91 ± 0.10	0.72 ± 0.13	0.99 ± 0.01	0.90 ± 0.08
	GRU	0.95 ± 0.02	0.84 ± 0.12	0.92 ± 0.09	0.72 ± 0.11	0.99 ± 0.01	0.88 ± 0.07
	LSTM	0.96 ± 0.02	0.91 ± 0.14	0.91 ± 0.09	0.81 ± 0.10	0.99 ± 0.01	$\mathbf{0.92} \pm \mathbf{0.07}$
	avg. length of S	N/A	690	469	1041	674	

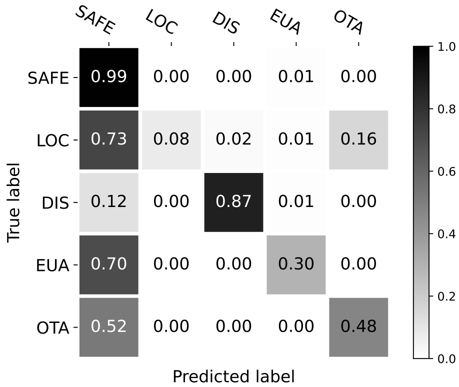
anomaly indicators are not simultaneously perceived but gathered at different time steps in EUA. In other words, the robot first receives a fall down sound, then the visual information of a cluttered environment is perceived. After the collapse, a part of objects may not be recognized. Furthermore, their locations are changed due to the collapse although they may be recognized. In such a scene, indicators of the following anomaly types exist: LOC, DIS, and EUA. Considering the average length of observation sequences for each anomaly type reported in Table 5.1, the anomalies including EUA occur in longer episodes (nearly twice as long) compared to the others. The LSTM-based method provides an average f-score of 0.81 for the classification of EUA cases, and an f-score of 0.72 as the closest score to this result is observed for the GRU-based method and the CRF-based (w/ L-BFGS) method. Therefore, it can be inferred that the LSTM-based model has the ability to relate indicators to anomaly classes in a temporal manner more efficiently compared to the other methods, and the robot can relate symptoms that appear earlier than the corresponding anomaly. This can be explained by the ability of LSTMs in transferring information through inner layers and capturing dependencies between indicators by maintaining forget gates to decide on the amount of information to be remembered and forgotten about the previous steps. Furthermore, lower standard deviation values are also provided compared to the other methods, and an overall f-score of 0.92 is achieved.

According to the results, low scores are gathered for EUA compared to the other anomaly cases for all methods. The reason for this situation can be interpreted as follows. In case of a collapse of a structure, the robot encounters a cluttered environment most of the time. The accuracy of the sensory interpretation decreases (especially for the visual modality) in such a complex scene. Therefore, the methods end up with relatively lower f-score scores for this anomaly compared to the other anomaly types.

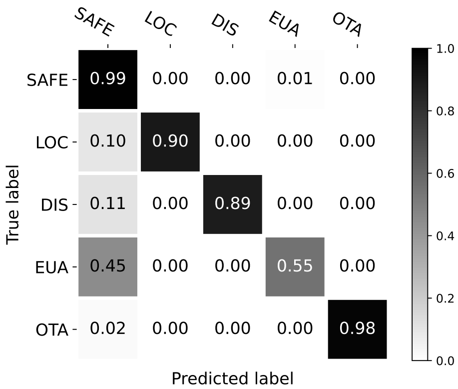
Figure 5.9 presents confusion matrices for the same task shown in Table 5.1. Each confusion matrix lays out the results of a different method: Figure 5.9(a) for the HMM-based method, Figure 5.9(b) for the AROW-trained CRF-based method, Figure 5.9(c) for the L-BFGS-trained CRF-based method and Figure 5.9(d) for the GRU-based method, respectively. According to these results, the CRF-based method (w/ AROW) and HMM-based methods confuse anomaly types most of the time. For



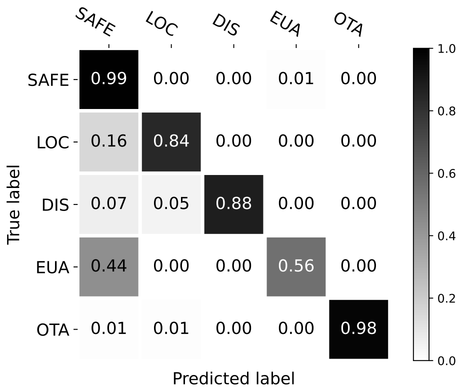
(a) The HMM-based method.



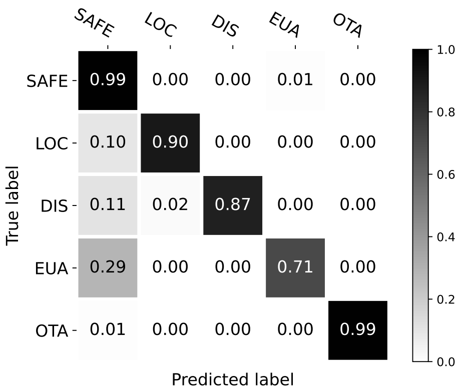
(b) The CRF-based method (w/ AROW).



(c) The CRF-based method (w/ L-BFGS).



(d) The GRU-based method.



(e) The LSTM-based method.

Figure 5.9 : Normalized confusion matrices of the methods for the state classification task.

instance, the CRF-based method (w/ AROW) confuses LOC (i.e., an object's location is changed by an external agent) with SAFE and OTA. The CRF-based method (w/ L-BFGS) substantially confuses EUA with SAFE, and lightly confuses LOC with SAFE (see Figure 5.9(c)). The GRU-based method confuses the same classes similarly (see Figure 5.9(d)). Nonetheless, the LSTM-based anomaly identification algorithm is able to classify anomaly types with a minimum success rate of 71% while minimizing the confusion among the anomaly types (see Figure 5.9(e)).

5.4.4.2 Overall identification analysis of the observation sequences

In this experiment, scenario (case) classification accuracy is analyzed. The robot assigns labels (anomaly classes) to each gathered observation sample in the observation history after an anomaly is detected. At the end of this task, a sequence of labels indicating anomaly classes is constructed. The overall accuracy of this task is presented in the previous analysis in Figure 5.9. A single overall decision on the occurred anomaly type should also be made for a corresponding scenario. To do so, a majority voting procedure is applied to come up with a final decision, and the anomaly class that gets the maximum number of votes is assumed to be the identification of that case of anomaly. The analysis of a single overall decision for the majority voting procedure is presented here.

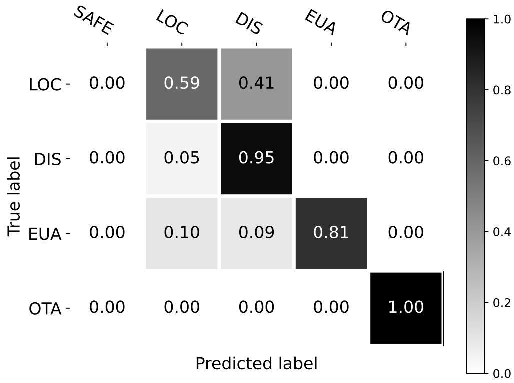
This experiment presents the performance analysis of the overall anomaly identification task. The results of these experiments are presented with distinct confusion matrices for each method in Figure 5.10. As can be seen from Figure 5.10(a), the HMM-based method confuses LOC with DIS most of the time as their indicators are quite similar in being related to location. However, EUA and OTA can be distinguished better. The confusion matrix for the AROW-trained CRF-based method is presented in Figure 5.10(b). Although lower performances are reported in Table 5.1, better scores for the case classification task are achieved for DIS and EUA. This situation can be explained with the following fact: The low state classification performance of the AROW-trained CRF-based method indicates confusion of the labels during the classification process most of the time. In other words, a consensus cannot be reached, and the algorithm switches among labels during the labeling phase of an observation history. Nonetheless, the majority of the assigned labels indicate the true anomaly label. Since the decision is taken with the majority voting procedure, an increase in the performance is achieved for DIS and EUA. On the other hand, LOC and OTA are confused more often for this method which leads to lower performance. The labeling of EUA is also unstable for the CRF-based (w/ L-BFGS) and GRU-based methods as in the previous case, and a higher performance is also reported in Figures 5.10(c) and 5.10(d) as well due to the same fact. The results for the LSTM-based algorithm are presented in Figure 5.10(e). As can be seen from this confusion matrix, the accuracy is 0.87 or above, and the confusion of the anomaly types for the case classification task is minimized. The label *SAFE* does not take part in these results, unlike Figure 5.9. This is because only the scenarios that include anomalies are tested, and a single overall decision about the investigated scenario is taken to explain the occurred anomaly instead of considering all labels in the observation sequence.

5.4.4.3 Sensory modality analysis

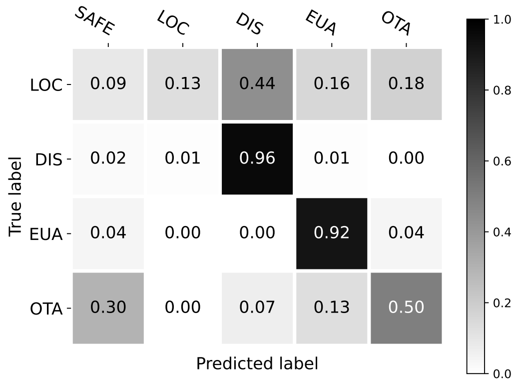
The presented framework combines the gathered observations related to the environment through various sensors. To do so, an early multimodal fusion procedure is adopted to learn these anomaly indicators. This experiment presents an analysis of the contribution of each selected sensory modality on the performance of the proposed symbol-based anomaly identification framework.

Table 5.2 presents the average precision, recall, and f-score scores of the multimodal sensor analysis for ten random executions to emphasize the effect of each sensory modality on the identification task. In the table, each row corresponds to a distinct single/multimodal setting presented with its corresponding symbols (s_v stands for for vision, s_a is for audition and s_p denotes proprioception). The scores of related metrics are presented in each column.

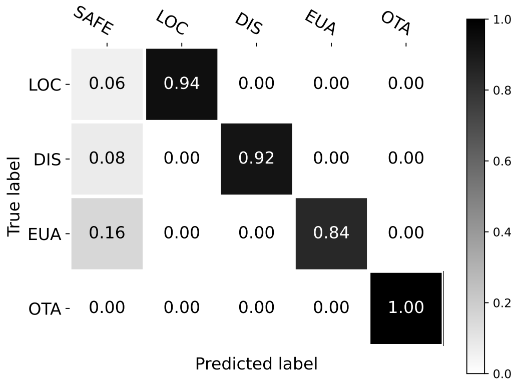
In this experiment, auxiliary data illustrated in Figure 5.3 are combined with proprioception (s_p). Vision (s_v) is considered as the base modality, and the analysis is constructed by incorporating other modalities on top of this modality. As a single modality, visual features perform an f-score of 0.71. It should be noted that LOC and EUA are mostly confused in this setting, and this can be explained as follows. Proprioception (s_p) provides discriminative indicators for LOC (i.e., it can be inferred that the manipulation of an object is not achieved via proprioception more easily), and



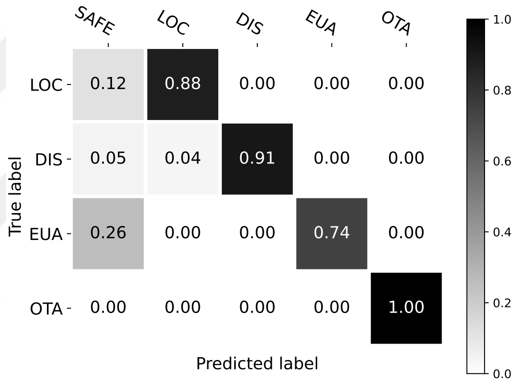
(a) The HMM-based method.



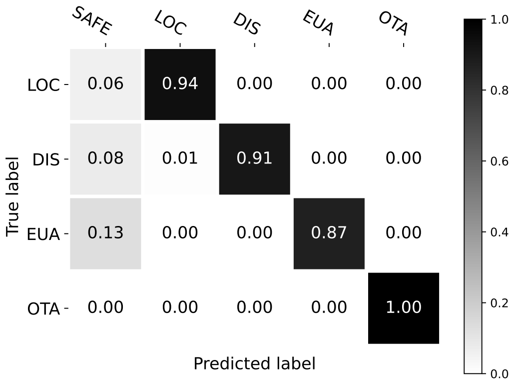
(b) The CRF-based method. (w/ AROW).



(c) The CRF-based method. (w/ L-BFGS).



(d) The GRU-based method.



(e) The LSTM-based method.

Figure 5.10 : Normalized confusion matrices of the methods for the overall case classification task.

audition (s_a) and vision (s_v) provide crucial indicators for collision and fall cases (i.e., EUA). Incorporating audition (s_a) on top of vision (s_v) leads to ending up with a higher recall (0.71) and f-score (0.74). Furthermore, combining proprioception (s_p) provides an f-score of 0.86. The setting that combines all selected modalities outperforms the other settings with an f-score of 0.92.

Table 5.2 : Performance analysis of the LSTM-based identification algorithm with different sensor modalities.

	Precision $\mu \pm \sigma$	Recall $\mu \pm \sigma$	F-score $\mu \pm \sigma$
s_v	0.88 ± 0.14	0.68 ± 0.12	0.71 ± 0.12
$s_v + s_a$	0.88 ± 0.11	0.71 ± 0.13	0.74 ± 0.12
$s_v + s_p$	0.93 ± 0.08	0.83 ± 0.13	0.86 ± 0.11
$s_v + s_a + s_p$	$\mathbf{0.96 \pm 0.04}$	$\mathbf{0.89 \pm 0.10}$	$\mathbf{0.92 \pm 0.07}$

5.5 Concluding Remarks

This chapter presents a symbol-based solution to the anomaly identification problem. The presented solution addresses the described problem by fusing multimodal domain symbols as features to relate anomaly indicators with symptoms of anomalies. The usage of symbolic features necessitates domain expertise and feature engineering which is an extra effort. Furthermore, a world modeling system is also a prerequisite for the proposed framework. Although the provided scores by the LSTM-based anomaly identification algorithm are impressive, the hand crafted features degrades the quality of the solution. The upcoming chapter presents a convolutional-based method for the feature extraction task that does not rely on such preprocessing procedures.

6. CONVOLUTIONAL THREE-STREAM ANOMALY IDENTIFICATION FRAMEWORK

In Chapter 5, a symbol-based anomaly identification framework is presented that processes high-level symbols as features to capture and relate anomaly indicators. In this chapter, the anomaly identification problem is addressed by capturing anomaly indicators with convolutional features. To do so, a ConvoLUtional threE-stream Anomaly Identification (CLUE-AI) framework is presented and analyzed. CLUE-AI processes auditory, visual and proprioceptive convolutional features, and adopts late fusion techniques. This chapter is organized as follows: First, the stages of the proposed framework with the processed sensory modalities are explained. Then, the experimental evaluation of the proposed framework is presented.

6.1 Convolutional Three-stream Anomaly Identification (CLUE-AI) Framework

Convolutional three-stream anomaly identification (CLUE-AI) framework consists of three stages in each of which the data stream of a distinct sensor modality are processed. The first stage processes the visual data (2D images) gathered from an RGB-D camera (ASUS Xtion RDB-D Camera) mounted on the head of the robot. The second stage processes auditory data collected through a microphone (PSEye microphone) which is placed on the lower torso of the Baxter robot. The last stage processes gripper-related data (i.e., the position of the gripper (its openness), the force applied by the gripper to the object at hand) to capture anomaly indicators. One may refer to Figure 3.1 to see the sensor placements on the robot. The following subsections elaborate on the procedures that take place on these aforementioned streams.

6.1.1 Visual stream

In this study, sequential RGB frames are extracted from the videos obtained from the head camera, and these frames are sampled at 0.125 Hz fixed frequency. This sampling process creates a frame sequence sorted in ascending order considering the timestamps of the frames. Convolutional visual features are extracted from each sampled image, and then the sequences of these features are fed into the LSTM layers by preserving the order of the sampled frames to capture temporal dependencies among anomaly indicators. These features are extracted with a pre-trained convolutional neural network (CNN) structure from the sequential 2D images gathered from the RGB-D camera placed on the head of the robot. Before the feature extraction task, images are cropped and resized ($\mathbb{R}^{224x224}$). After this transformation phase, transformed images are fed into a pre-trained CNN structure to detect the relevant parts of them. In order to fulfill this task, the following CNN architectures are used to extract features from sequential 2D images:

- **Residual Net (ResNet)** [115]: ResNets are neural networks that ease the training phase of deeper neural networks. They are not only easier to optimize, but also can achieve high accuracy for deeper networks. The main idea behind this architecture is that the layers are considered as residual functions that reference the layer inputs instead of learning functions without references. Learning with up to 152 layers is possible with ResNets.
- **AlexNet** [116]: AlexNet is a CNN architecture that includes 5 convolution layers followed by 3 fully-connected layers. The first, second and fifth convolution layers are immediately followed by a max pooling operation. The architecture achieves high performance on the ImageNet dataset.
- **VGG** [117]: The main idea behind this structure relies on the evaluation of networks with increasing depths using an architecture with very small (3x3) convolution filters. It generally includes 16 to 19 layers. In this study, the VGG variant with 16 layers (VGG16) is employed. VGG16 consists of 13 convolution layers and 3 fully connected layers. Each convolution layer is followed by ReLU, and a max pooling operation takes place after the second, fourth, seventh, tenth and thirteenth convolution layers.

Pre-trained implementations of these vision models on ImageNet [118] are used in this study. These models are trained with a wide range of images including various objects. Since these pre-trained CNNs are only used for extracting relevant features, the last layers of them that include fully connected layers are not used. This structure is used to extract relevant features of the images. Each feature vector at time t is denoted with f_t^v. After the relevant visual features (f^v) are extracted related to anomaly indicators, a dropout function is applied with a probability of 0.4. The extracted features for each image are considered as a sequence, and these features are fed into LSTMs to capture temporal dependencies among anomaly indicators. Afterward, the gathered result from the LSTM is fed into an attention layer to find out the importance of the images in the sequence contributing to the anomaly decision. In this study, scaled dot-product self-attention [100] is employed to come up with attention values related to the images in the sequence by the LSTM layers. After self-attention values are calculated, they are fed into a dense layer. Then, the output vector including the concatenation of the attention output along with the LSTM outputs ($r^v \in \mathbb{R}^{1024}$) is generated to be fused with that of the auditory and proprioceptive modality.

Figure 6.1 illustrates the phases of the proposed CLUE-AI framework while processing visual modality. The proposed visual processing scheme consists of two phases to process sequential RGB images: the extraction of convolutional features and learning these extracted features, respectively. Sequential 2D RGB images are accepted as inputs. In the feature extraction layers, green layers denote convolution layers where blue ones are for the pooling operations. Convolution layers are marked with corresponding kernel sizes and channels. These sequential layers construct the VGG network without its classification layers that include fully connected layers. This structure is followed by a dropout layer, and it is immediately followed by LSTM layers that learn the visual anomaly indicators. Then, the LSTM outputs are fed into an attention layer to end up with attention values related to the positions of the images in the sequence. Note that rectified linear unit (ReLU) is used as an activation function after each convolution layer, and they are omitted in the figure for brevity.

6.1.2 Auditory stream

Auditory data gathered through the microphone mounted on the torso of the robot are processed with two phases. In the first phase, features of the gathered auditory data are extracted. In order to do so, Mel-frequency Cepstral Coefficients (MFCC) features are extracted. The library Librosa [119] is used for this aim. Extracted auditory features of an observation gathered at time step t is denoted with f_t^a. The second phase consists

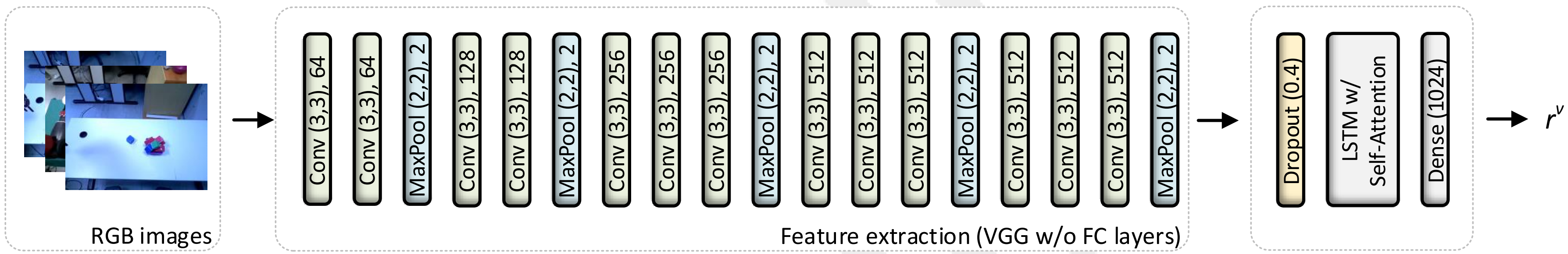


Figure 6.1 : The architecture for processing visual stream.

of feeding these extracted MFCC features (f^a) into a CNN block. The block includes four sequential convolution layers. Note that each convolution layer is followed by a ReLU unit as an activation function. After these operations, max-pooling is applied, and the resulting vector is fed into a dense layer that outputs the resulting auditory feature vector ($r^a \in \mathbb{R}^{64}$).

Figure 6.2 shows the contents of the auditory stage of the proposed framework which processes audio data. As can be seen from the figure, the audio data are processed in two phases. First, MFCC features of the audio data are extracted and this is followed by processing them in convolution layers. The gathered outputs from the convolution layers are fed into a dense layer and a vector that includes processed auditory features is constructed.

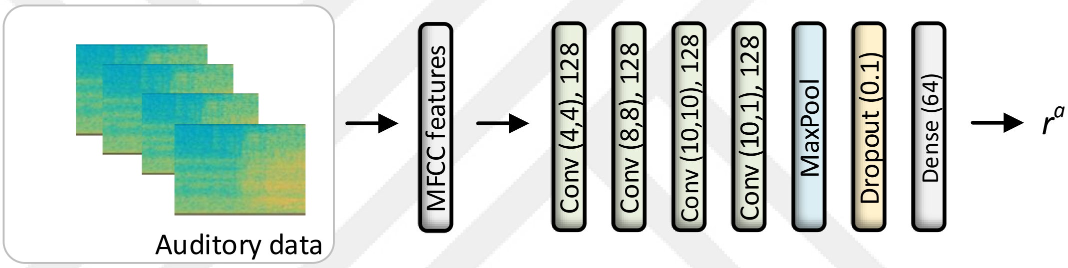


Figure 6.2 : The architecture for processing auditory stream.

6.1.3 Proprioceptive stream

The robot manipulates the objects in the environment with the gripper attached to its arm (Figure 3.1). Proprioceptive observations are continuously collected with the gripper. In this study, the openness of the gripper and the force applied by the gripper to the object at hand are considered as features (f^p). These features are processed within a tiny CNN illustrated in Figure 6.3. The output of this CNN is fed into a max pooling layer that is followed by a dense and dropout layer. As a result, the final vector related to the gripper data is constructed ($r^p \in \mathbb{R}^{64}$).

6.1.4 Late fusion

The explained visual, auditory and proprioceptive streams make of the corresponding feature vectors. Since a late fusion approach is adopted, these obtained visual, auditory and proprioceptive features are concatenated to a single fused vector ($r^{fused} \in \mathbb{R}^{1152}$).

Afterward, the resulting vector of this fusion is fed into a dense layer which is followed by a dropout layer with a probability of 0.4. The resulting vector is then fed into another dense layer and a softmax function. Considering these results, the most probable anomaly type is chosen for the identification of the anomaly. Figure 6.4 illustrates the described late fusion procedure.

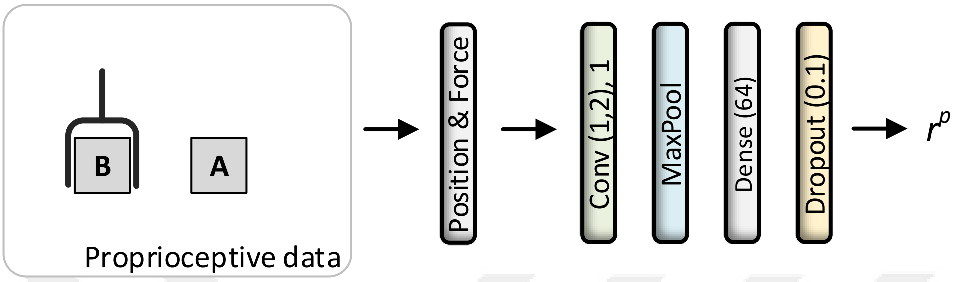


Figure 6.3 : The architecture for processing proprioceptive stream.

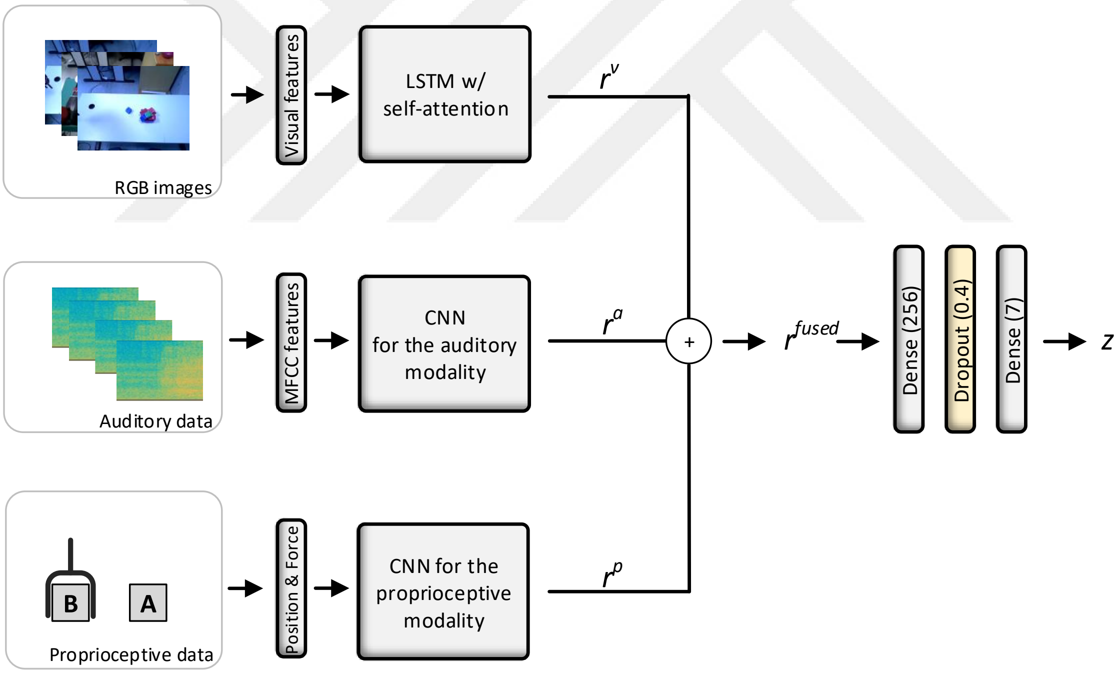


Figure 6.4 : Proposed late fusion scheme of the CLUE-AI framework.

6.2 Experiments

This section presents the evaluation of the CLUE-AI framework in the real-world experiments with the Baxter robot performing everyday object manipulation scenarios. Before presenting the evaluations, the experimental setup is explained.

6.2.1 Experimental setup

To evaluate the presented framework, the experiments are conducted on 7 distinct classes involving 6 anomaly cases and the safe cases. Note that safe scenarios are also included in the evaluation to identify and handle possible false alarms that may arise in the anomaly detection procedure. Table 6.1 presents the distribution of the number of scenarios. In the table, the row indicates the number of the scenarios of the corresponding anomaly type of the column. Note that the same anomaly detection method that is used in Chapter 5 is employed for the explained anomaly identification algorithm. Experiments are run on a server having the following configuration: Intel Core i7-7700K 4.20GHz CPU, 32 GB RAM, NVIDIA Quadro P6000 GPU with 24 GB memory.

Table 6.1 : Number of scenarios for each anomaly type.

	SAFE	LOC	DIS	EUA	OTA	SPC	FCA	Total
# of scenarios	68	22	41	33	18	43	24	249

Several parameters are set empirically and used for the training phase of the models. Table 6.2 lists these parameters. In order to cope with the class imbalance problem of the data set, an adaptive weighting approach where anomaly classes with the less total number of instances are valued more is used while updating the network. The data set is partitioned randomly into the train (80%) and test (20%) sets for each execution for each class. The models are trained for 100 epochs, and averages of best results are reported for ten random seeds.

Table 6.2 : Parameter dictionary of the LSTM structure.

Parameter	Value
Neurons	512
Layers	1
Loss	Cross Entropy
Optimizer	Adam
Learning rate (η)	10^{-4}

6.2.2 Experimental evaluation

CLUE-AI framework is evaluated on various aspects. First, the impact of different feature extraction techniques is presented. This is followed by a set of analyses including comparative analysis, multimodality analysis, noisy data analysis, transformer analysis and auditory analysis.

6.2.2.1 Training analysis

This experimental evaluation presents an analysis of the number of epochs as a training parameter. In order to do so, the data set is split into training and test sets (80% and 20%, respectively). Figure 6.5 presents an analysis of the loss values with respect to the number of iterated epochs. The x-axis indicates the epoch, and the y-axis indicates the loss. As can be seen from the plot, a model can be selected around the fortieth epoch to avoid overfitting the model. This experiment is conducted for ten random seeds

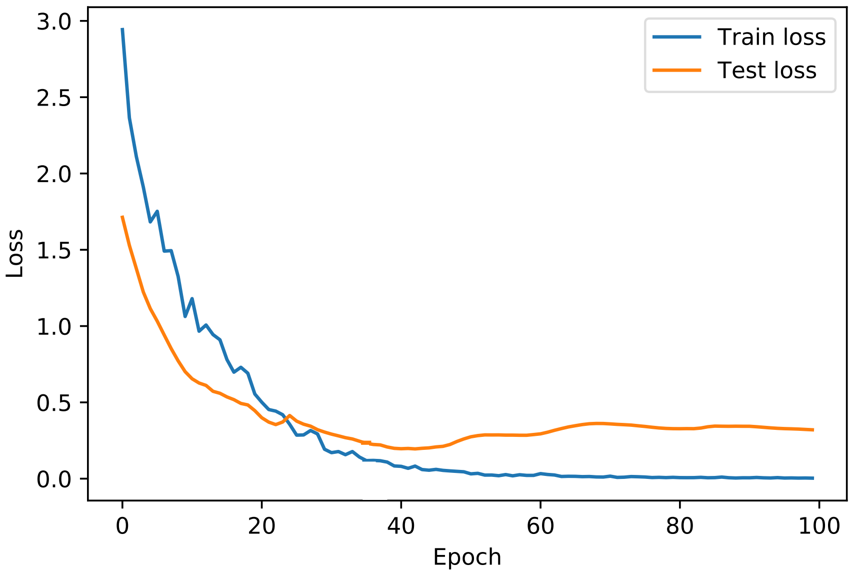


Figure 6.5 : Change on the train and test loss for $\eta = 10^{-4}$ for the VGG16-based setting.

for each visual models that are used for feature extraction, and Table 6.3 presents the average number of epochs that the training is stopped for each visual model considering both loss values to avoid overfitting. Considering these results, the epoch numbers presented in Table 6.3 are used to stop training for each model. This table also presents the number of parameters of the corresponding models excluding the fully connected layers.

6.2.2.2 Feature extraction analysis

The CLUE-AI framework is evaluated with four different feature extraction settings employing the following CNNs: ResNet18, ResNet101, AlexNet, and VGG16. Note

Table 6.3 : Number of epochs that each model is trained and the number of parameters (in millions) that each vision model includes.

	VGG16	AlexNet	ResNet18	ResNet101
# of epochs	40	55	55	60
# of parameters	14.7	2.4	11.1	42.5

that ResNet18 and ResNet101 are two variants of ResNets where a different number of layers form the corresponding CNN structure. Figures 6.6-6.9 presents the normalized confusion matrices for different CNNs that extract visual features. Note that the abbreviations corresponding to the names of the anomaly classes are presented in Section 3.2.2. The values in cells represent the normalized average results for ten random seeds.

Considering the results in Figure 6.6 and Figure 6.7, one may infer that ResNets cannot extract visual features effectively for some classes. This situation mainly arises due to the average pooling (AP) layers (with a square kernel size of 7) of both ResNets variations that are placed before the excluded dense layers. This results in a decrease in the size of the extracted features. Note that only the dense layers of these models are excluded in this study. Such a situation results in ambiguity in anomaly classes since anomaly indicators cannot be distinguished. To exemplify, SAFE and LOC are mostly confused classes for ResNet18-based setting. As can be seen from Figure 6.6, LOC (the anomaly cases where an object's location is changed by an external agent) is confused with DIS (the anomaly cases where an object is taken out of the scene by a human). That is, the location change of the object is confused with the disappearance. The ResNet101-based setting also provides similar results with the ResNet18-based setting. On the other hand, AlexNet and VGG16-based settings provide more accurate classification results for the anomaly identification task.

Considering the results for the AlexNet structure as the feature extractor, the minimum classification score belongs to class SAFE among the classes with a value of 0.85. When VGG16 is invoked to extract relevant information related to anomaly symptoms, a minimum score of 0.85 is achieved for the SAFE class where no anomaly occurs. The interpretation of this situation is that a little number of chickpeas may be spilled on the table, being occluded by the container on some occasions.

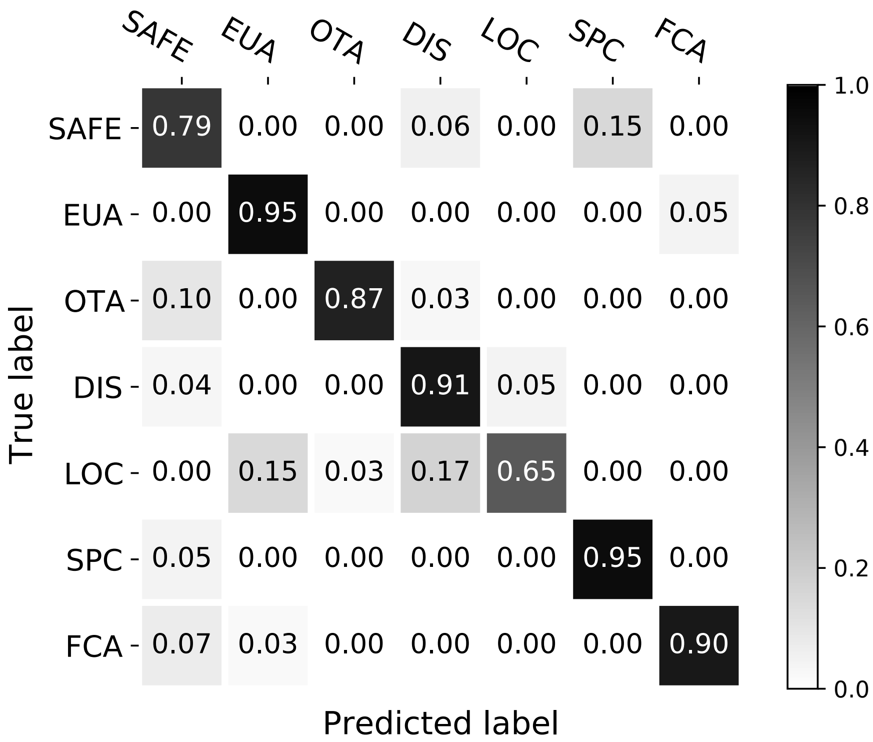


Figure 6.6 : ResNet18 as a feature extractor.

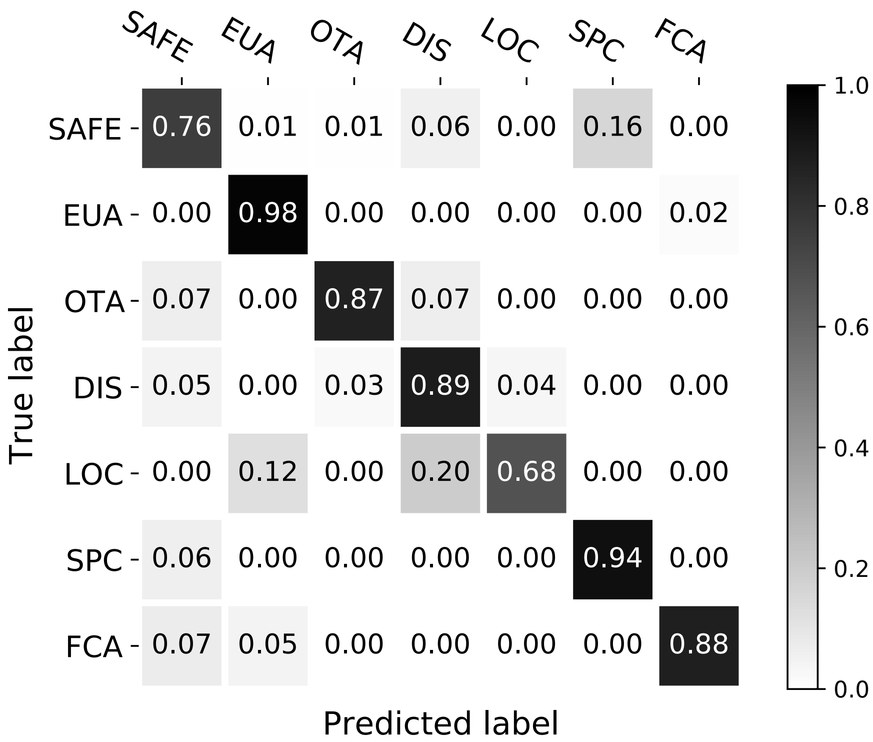


Figure 6.7 : ResNet101 as a feature extractor.

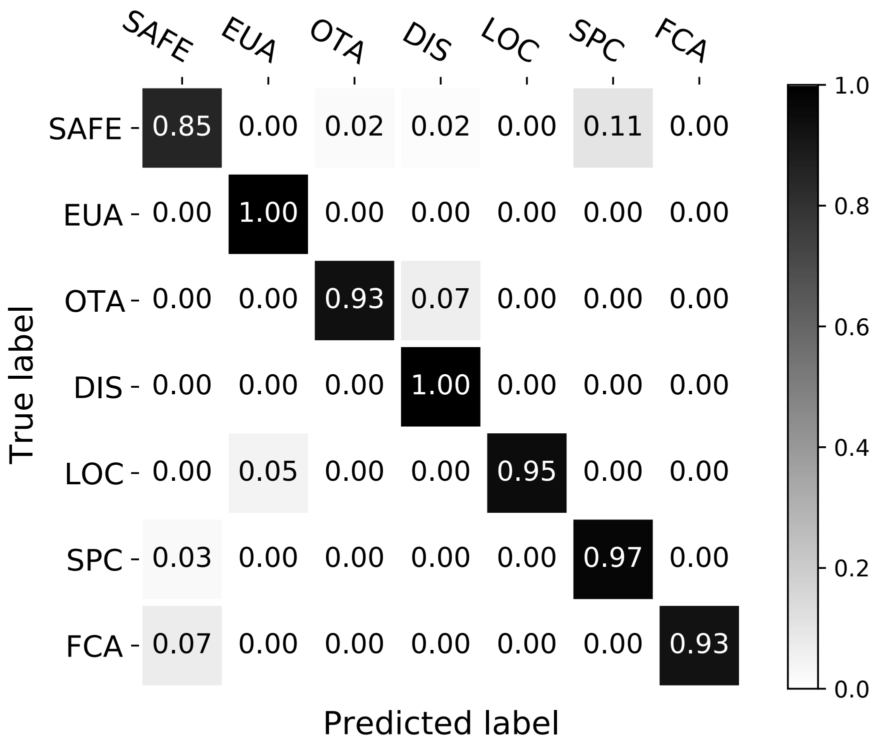


Figure 6.8 : AlexNet as a feature extractor.

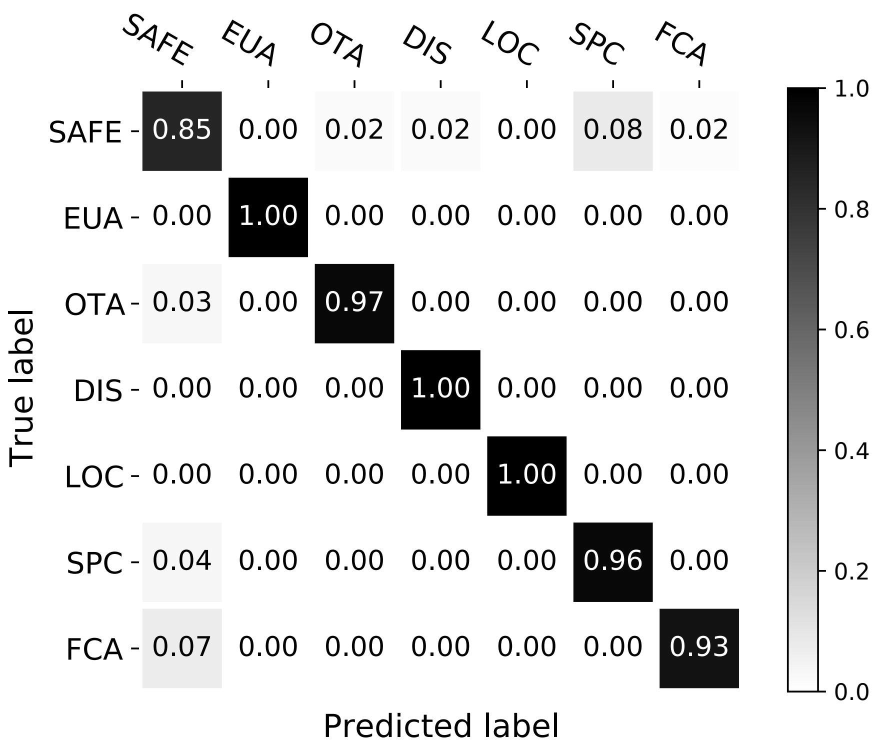


Figure 6.9 : VGG16 as a feature extractor.

Table 6.4 summarizes the performance analysis of the presented feature extraction methods in terms of precision, recall and f-score metrics. In this analysis, both variations of ResNets with and without average pooling (AP) layers placed last are considered. Although ResNet-based feature extraction methods provide better performances without AP layers, they cannot provide better scores in terms of all presented metrics, the AlexNet-based feature extraction technique provides an f-score of 93.88%. Note that the usage of the AP layers decreases the number of extracted features, and that results in degraded performances for ResNet-based settings. As can be seen from the table, VGG16 has the ability to extract relevant visual features, and it outperforms the other ResNet-based settings and slightly gives better results than the AlexNet-based setting with an f-score of 94.34%.

Table 6.4 : Performance evaluation of different feature extraction methods.

	Precision ($\mu \pm \sigma$)	Recall ($\mu \pm \sigma$)	F-score ($\mu \pm \sigma$)
CLUE-AI w/ ResNet18 w/ AP	89.32 ± 2.17	87.60 ± 2.39	87.16 ± 2.75
CLUE-AI w/ ResNet18 w/o AP	92.65 ± 2.54	91.52 ± 2.82	91.43 ± 2.85
CLUE-AI w/ ResNet101 w/ AP	89.63 ± 4.50	88.04 ± 4.48	87.76 ± 4.49
CLUE-AI w/ ResNet101 w/o AP	91.86 ± 3.80	90.43 ± 4.03	90.33 ± 4.15
CLUE-AI w/ AlexNet	94.63 ± 2.57	93.91 ± 2.71	93.88 ± 2.76
CLUE-AI w/ VGG16	94.90 ± 2.23	94.34 ± 2.21	$\mathbf{94.34 \pm 2.26}$

Table 6.5 presents the elapsed time for training and testing the anomaly models with the features extracted by different CNNs. Each row includes the results of a distinct feature extraction method where each column corresponds to the elapsed times for training and testing in terms of seconds (sec). Average and standard deviation of the execution times is reported for ten executions. As can be seen from the results, CLUE-AI with AlexNet provides the lowest model training and testing time for extracting and iterating the extracted features in LSTMs for a single epoch.

6.2.2.3 Overall performance analysis

This experimental analysis presents the comparison of the proposed CLUE-AI framework with different methods for the anomaly identification task. Hidden Markov models (HMMs) and recurrent neural networks (RNNs) are selected as other methods. These methods are selected for comparison since these models are state-of-the-art machine learning algorithms in the literature for the anomaly

Table 6.5 : Time elapsed (sec) during training and testing with different feature extraction techniques.

	Train ($\mu \pm \sigma$)	Test ($\mu \pm \sigma$)
CLUE-AI w/ ResNet18 w/ AP	7.43 ± 0.05	0.33 ± 0.01
CLUE-AI w/ ResNet18 w/o AP	10.39 ± 0.04	1.12 ± 0.01
CLUE-AI w/ ResNet101 w/ AP	11.35 ± 0.07	1.22 ± 0.03
CLUE-AI w/ ResNet101 w/o AP	18.89 ± 0.08	2.38 ± 0.04
CLUE-AI w/ AlexNet	$\mathbf{7.12 \pm 0.05}$	$\mathbf{0.21 \pm 0.01}$
CLUE-AI w/ VGG16	11.77 ± 0.07	1.20 ± 0.03

identification problem. For the settings where RNNs are employed, LSTMs are replaced with the corresponding structures in the CLUE-AI framework. For the HMM-based method, incremental principal components analysis (IPCA) is used to extract relevant features from the input images. Afterward, these features are fed into HMMs to classify the anomaly type. An HMM model with binary latent states (safe and the corresponding anomaly type) is trained for each anomaly type. In case of an anomaly, the observation history, including images, is fed into each trained model. This results in the scores from each anomaly model corresponding to the likelihood of being generated by that anomaly model. Then, these likelihoods are processed along with the results of the auditory stage, and the anomaly type is revealed. Each method is run ten times with different seeds, and the weighted average scores are reported in Table 6.6.

Table 6.6 : Comparative performance analysis of different methods.

	Precision ($\mu \pm \sigma$)	Recall ($\mu \pm \sigma$)	F-score ($\mu \pm \sigma$)
HMM	74.05 ± 4.31	71.63 ± 5.73	70.75 ± 6.08
Symbol-based framework	96.02 ± 4.01	89.22 ± 10.02	92.05 ± 7.23
CLUE-AI w/ VGG16-RNN	93.51 ± 3.83	92.39 ± 4.48	92.34 ± 4.51
CLUE-AI w/ VGG16-LSTM	94.90 ± 2.23	94.34 ± 2.21	$\mathbf{94.34 \pm 2.26}$

Considering the results reported in Table 6.6, it can be seen that VGG16 as the feature extractor with LSTMs (CLUE-AI w/ VGG16-LSTM) provides better results than the RNN-based deep learning-based method, the symbol-based anomaly identification framework that is explained in Chapter 5 and HMMs with IPCA as the feature extractor (IPCA-HMM) in terms of all criterion. When the extracted features by VGG16 are processed with RNNs, it provides an f-score of 92.34%. When VGG16 and LSTMs are combined together in the CLUE-AI framework, this setting outperforms the other methods and provides an f-score of 94.34% in classifying anomaly types.

Compared to the results obtained with the anomaly identification framework that takes into account symbols as visual features it can be inferred that the CLUE-AI framework with VGG16-LSTM outperforms the symbol-based anomaly identification framework (SAIF) with an f-score of 94%. It should also be noted that the CLUE-AI framework is evaluated on an extended set of anomaly scenarios (i.e., a higher number of anomaly classes) compared to the symbol-based anomaly identification algorithm. This extended set of anomaly scenarios also includes challenging scenes for a scene interpretation system (i.e., cluttered environments or containers full of objects).

Table 6.7 presents the elapsed time for training and testing different deep learning models for the anomaly identification problem. Each row indicates includes the results of a different method where each column indicates the elapsed times for training and test sets in terms seconds (sec). Average and standard deviation values of the execution times are reported for ten executions. According to these results, the lowest execution times are achieved by the RNN-based method compared to the LSTM-based method.

Table 6.7 : Time elapsed (sec) during training and testing of different deep learning methods.

	Train ($\mu \pm \sigma$)	Test ($\mu \pm \sigma$)
CLUE-AI w/ VGG16-RNN	**10.87 ± 0.07**	**1.14 ± 0.03**
CLUE-AI w/ VGG16-LSTM	11.77 ± 0.07	1.20 ± 0.03

6.2.2.4 Multimodality analysis

The proposed CLUE-AI framework in this chapter processes the data gathered from distinct sensory modalities. A multimodality analysis is performed to analyze the contribution of each processed sensory modality and the attention mechanism in identifying the anomaly types.

Table 6.8 presents the precision, recall, and f-score metrics for different sensory modality settings of the CLUE-AI framework. Moreover, one column in the table is reserved to indicate whether the attention mechanism is employed in the corresponding setting. In the table, each row corresponds to a different setting where each column represents the scores for the corresponding metric. As can be seen from the table, the gripper-related data as a stand-alone stream provides an f-score of 50.52%, and the single auditory modality setting provides an f-score of 80.52%. When only visual modality with self-attention is considered for classification, this setting provides an f-score of 88.70%. As expected, one can infer that the visual modality provides more information about anomaly indicators compared to the auditory modality. When the gripper-related data are combined with the visual modality, a minor improvement in the f-score (88.72%) is obtained. The fusion of visual and auditory modalities with self-attention provides higher performance with an f-score of 93.51%. Incorporating the auditory modality provides the opportunity to distinguish anomaly cases more efficiently in which sound is a discriminating indicator (i.e., the robot perceives sound as observation when an object falls down as it is pushed; on the other hand, it does not perceive sound when an external agent changes the object's location.). As can be seen from the last two rows of the table, incorporating the attention mechanism with the three modality stream provides an improvement of 2% on the f-score.

Table 6.8 : Sensory modality and attention analysis.

Modality			Attention	Precision	Recall	F-Score
s_v	s_a	s_p				
		✓		52.40 ± 9.72	54.78 ± 8.68	50.52 ± 9.24
	✓			82.48 ± 5.22	81.73 ± 4.14	80.52 ± 4.95
✓			✓	89.83 ± 4.62	88.91 ± 4.50	88.70 ± 4.69
✓		✓	✓	90.24 ± 3.91	88.91 ± 3.82	88.72 ± 4.83
✓	✓		✓	94.23 ± 3.09	93.47 ± 3.36	93.51 ± 3.29
✓	✓	✓		93.55 ± 2.44	92.33 ± 2.88	92.33 ± 2.90
✓	✓	✓	✓	94.90 ± 2.23	94.34 ± 2.21	**94.34** ± **2.26**

In this thesis, audio modality is processed as the complementary data. Nevertheless, for instance, no auditory perception is gathered generally in disappearance cases. Moreover, perceived audio characteristics are similar for SPC (the anomaly cases where objects are spilled in a pouring task due to overflowing) and SAFE scenarios including pouring tasks, and the sounds that arise when an object hits the table and to the container intermingle. Therefore, it may not be an essential indicator for such anomaly types. One should note that Baxter's location does not change in the experiments, its torso is fixed. It only moves its arm to interact with objects in the environment. Therefore, the motors' of its arm make sound during its movements generating ego noise.

Figures 6.10-6.16 illustrate class activation map representations of selected cases. In this study, gradient-weighted class activation mapping-based (Grad-CAM) [120] representations of selected cases are presented. Class activation maps (CAMs) give an insight into which parts of the input contribute to the final decision. Red regions indicate a high contribution to the final decision, where blue regions indicate a lower contribution. Left images represent the images perceived through the camera of the robot, right images represent the corresponding CAMs in both figures. Figure 6.10 presents a SAFE case where the robot successfully places an object into a container. As can be seen from the corresponding CAM, the region where the container exists contributes more to the decision compared to the other regions on the image. In Figure 6.11, an FCA case is analyzed where an object falls out of the container on the table. It can be inferred that the framework focuses on the region where the object that falls out of the container rather than the container itself.

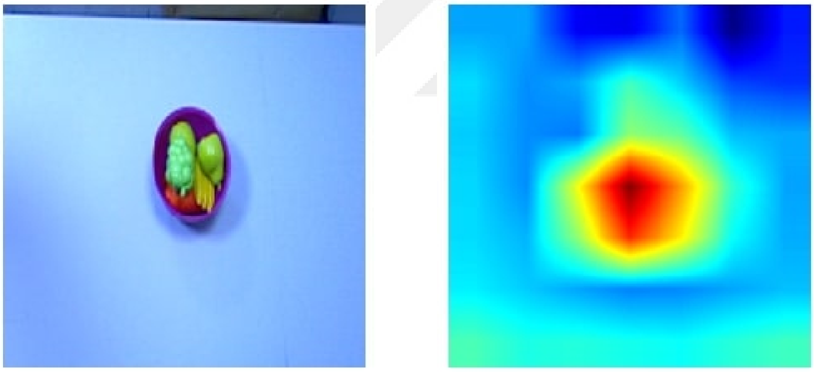


Figure 6.10 : CAM of a SAFE case.

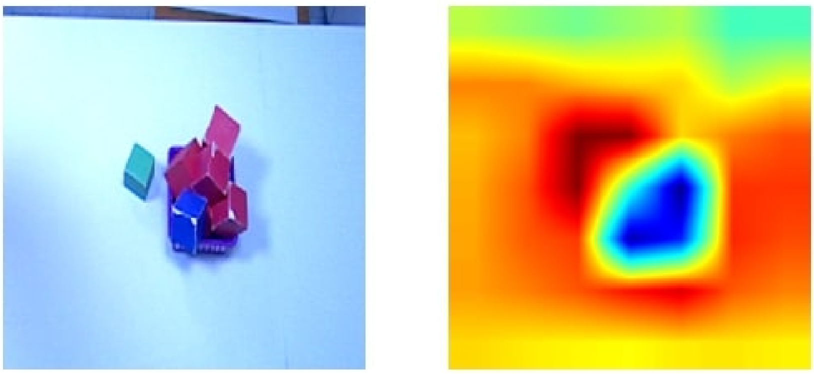


Figure 6.11 : CAM of an FCA case.

Figure 6.12 illustrates the visual observation and the corresponding CAM (right) for an OTA case where an object overturns while it is manipulated. As can be seen from the CAM, the corresponding region where the object falls has a dense red coloring compared to the other parts.

Figure 6.13 illustrates the CAM for an EUA case where a tower structure collapses during construction due to an earlier unstable placement. The CAM has dense red regions where the tower structure collapses, and its surrounding regions are green.

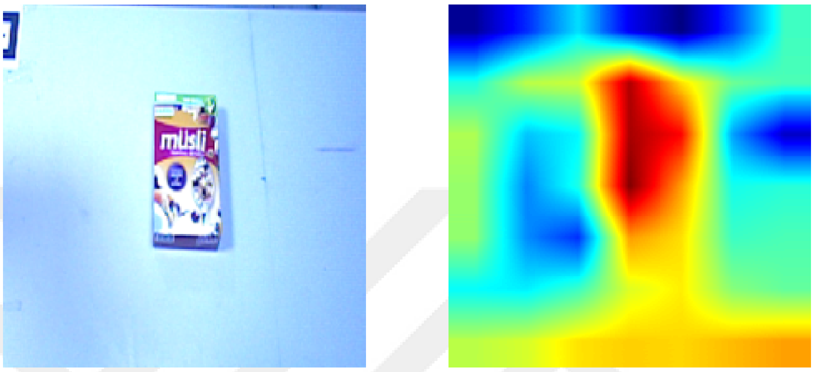


Figure 6.12 : CAM of an OTA case.

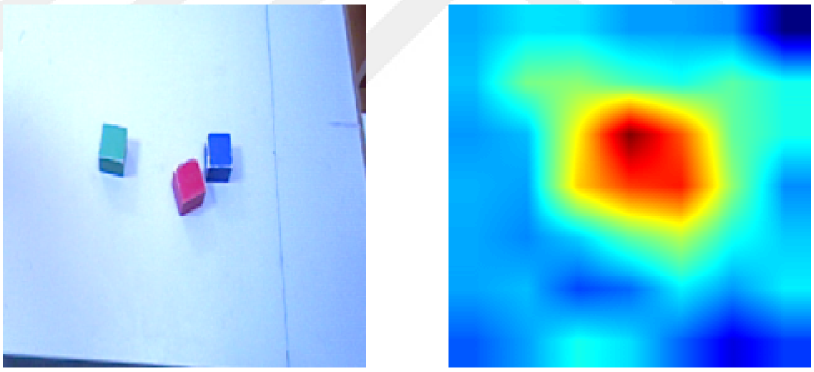


Figure 6.13 : CAM of a EUA case.

Figure 6.14 illustrates an SPC case and its CAM that maps the contribution of the regions on the input image. The corresponding CAM indicates that the container and its surrounding regions where the objects are spilled contribute more to the anomaly decision. Figure 6.15 illustrates the CAM for a DIS case where an object is taken out from the environment. The cam indicates an almost uniform color distribution in the central regions, and this is probably due to trying to detect the object in the scene.

Another CAM is presented in Figure 6.16 for a LOC case where the location of the object is changed by an external agent. The CAM indicates that the initial location of the object (left upper region) has a significant contribution to the decision.

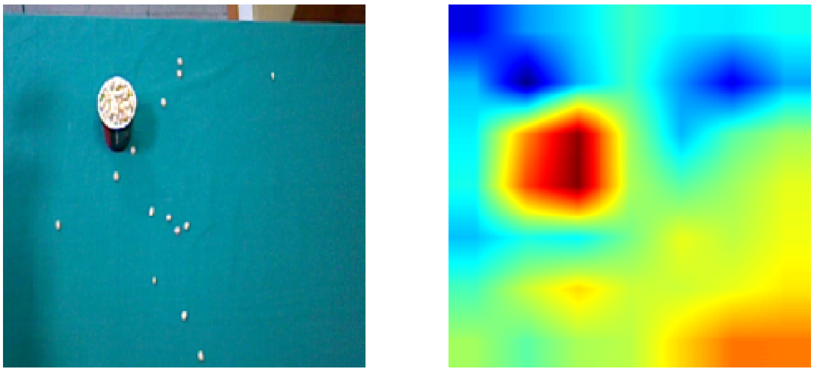


Figure 6.14 : CAM of an SPC case.

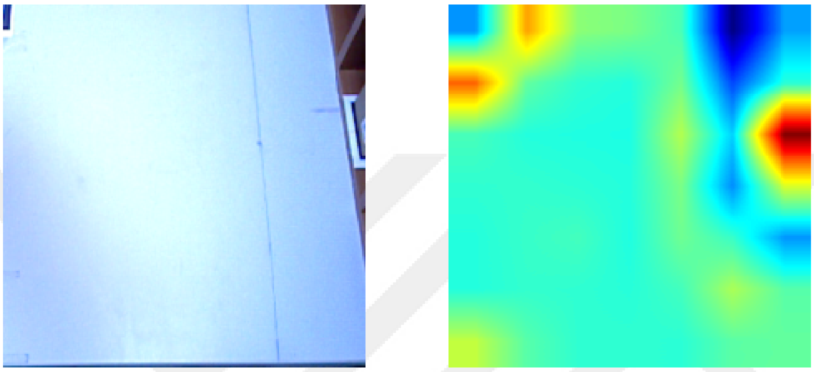


Figure 6.15 : CAM of a DIS case.

6.2.2.5 Performance analysis on noisy data

In this experiment, the performance of the presented CLUE-AI framework is demonstrated on noisy sensory test data. In order to do so, random noise is injected into the perceived data (i.e., each pixel of the image is flipped to zero with a given probability value).

Figure 6.17 presents the performance analysis of the CLUE-AI framework for different random noise probabilities with bar graphs. Each bar on the x-axis corresponds to a distinct noise ratio (i.e., with a probability of that value, the corresponding feature is flipped), and the y axis corresponds to the f-score value achieved by that setting. As can be seen from the bars, the setting without any injected noise provides the highest f-score. After the random noise probability is set as 0.3, the performance decreases faster, and the standard deviation values increase. As the injected noise increases, the performance of the framework degrades, ending up with an f-score slightly above 30% for the setting with a noise probability of 0.8.

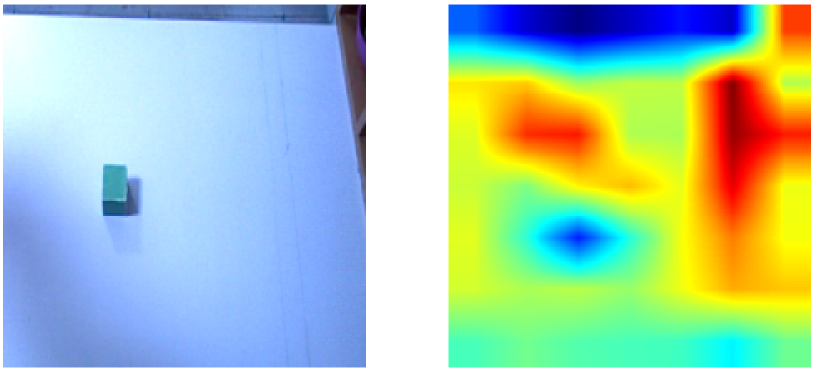


Figure 6.16 : CAM of a LOC case.

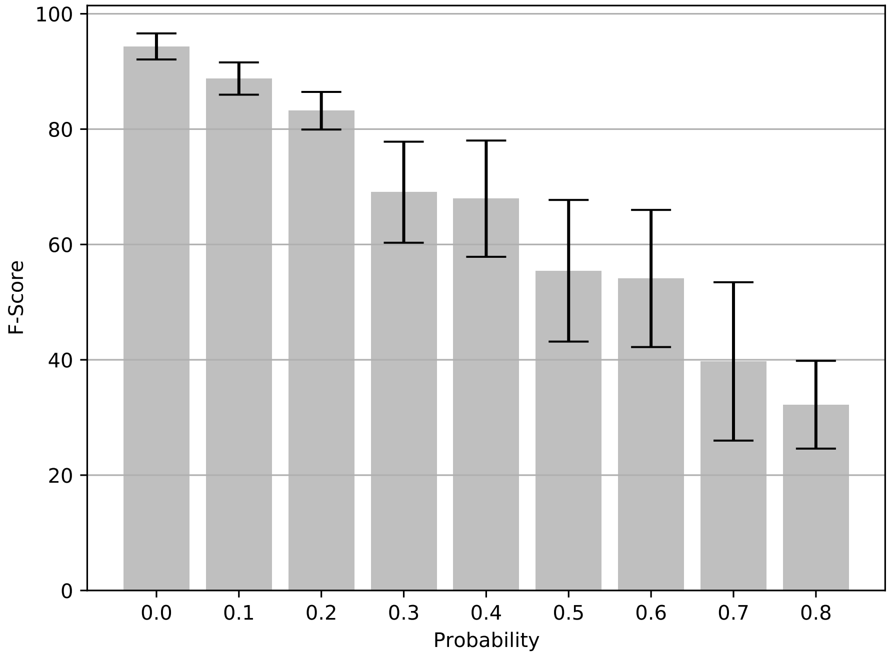


Figure 6.17 : Performance analysis of the CLUE-AI framework on noisy data.

6.2.2.6 Transformer analysis

In this experiment, the CLUE-AI framework is analyzed with a transformer structure replacing the attention-enabled LSTM structure of the framework. Transformers process sequential data by employing attention mechanisms [100, 121]. Transformer architectures include encoders and decoders that consist of a number of layers. Each layer is accompanied by a self-attention mechanism, and these attention mechanisms may include multiple attention heads. In this analysis, transformers with encoder layers are employed , and the analysis for a different number of layers and attention heads is presented.

Table 6.9 presents the analysis of the presented framework with different parameters settings. The first two columns present the settings where each row presents the related scores for each corresponding setting. As can be seen from the table, although changing the number of encoder layers provides similar scores, increasing the number of heads in the multi-head attention provides an improvement ending up with an f-score of 88.47%.

Table 6.9 : Analysis of the CLUE-AI framework with transformer layers.

# of encoder layers	# of multi-heads	Precision ($\mu \pm \sigma$)	Recall ($\mu \pm \sigma$)	F-Score ($\mu \pm \sigma$)
4	2	89.05 ± 2.62	88.02 ± 2.69	87.99 ± 2.71
	4	89.62 ± 3.04	88.45 ± 3.00	**88.47 ± 3.00**
6	2	88.40 ± 4.07	87.60 ± 4.12	87.59 ± 4.22
	4	89.48 ± 3.38	88.45 ± 3.66	**88.47 ± 3.67**

6.2.2.7 Analysis of audio processing

The CLUE-AI framework considers auditory data as 2D data since the stream is represented as a 2D matrix ([time, features]). Therefore, convolution layers with 2D kernels are employed. In this analysis, processing auditory data with different shaped kernels is evaluated. To do so, rectangular kernels (i.e., (16,4) and (16,5) with strides of the same size) are introduced to the auditory stage of the CLUE-AI framework.

Figure 6.18 illustrates the scores achieved by different shaped kernels for the auditory stream processing. The x-axis presents metrics for each setting, and the y-axis presents the corresponding scores. The first bars (the light gray bars) for each metric are for the setting with rectangular kernels and the second bars (the dark gray bars) are for the setting with square kernels. As can be seen from the plot, the CNN structure with square kernels on the auditory stream provides an improvement for each metric, especially an improvement of 4% in the f-score.

6.3 Concluding Remarks

This chapter presents an anomaly identification framework, CLUE-AI, that takes into account convolutional features by processing three sensory modalities, namely visual, auditory and proprioceptive to handle anomaly cases that may arise during everyday object manipulation tasks. The CLUE-AI framework consists of three phases in each

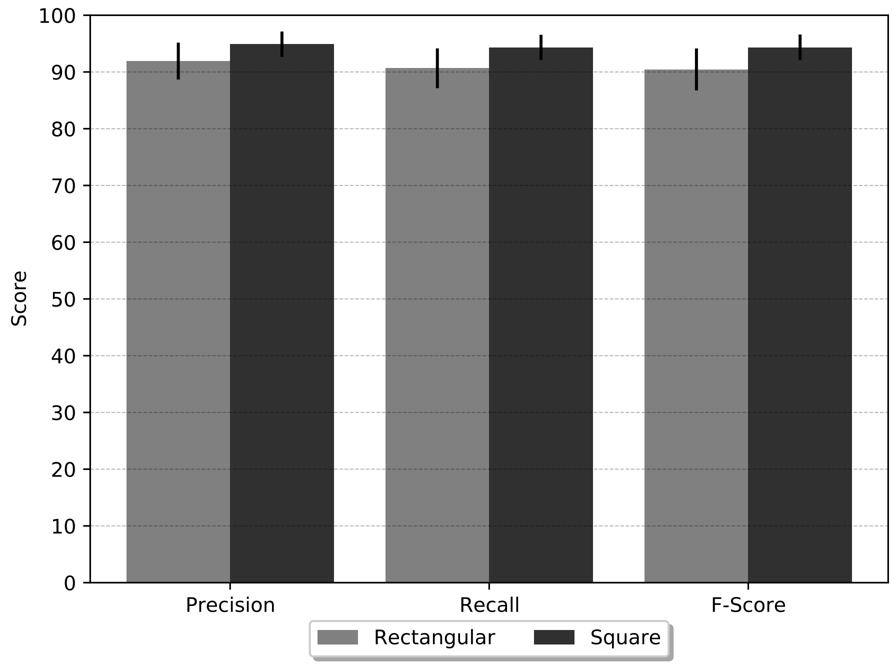


Figure 6.18 : Performance analysis of the CLUE-AI framework with different shaped kernels.

of which a distinct sensory stream is processed. First, convolutional visual features are extracted by VGG16, and they are fed into LSTM cells which is followed by a self-attention mechanism to capture anomaly indicators. Then, MFCC features are extracted from the auditory modality, then a CNN block is employed. Last, gripper-related data are processed with a designed CNN. Afterward, these sensory data are combined with a late fusion methodology. As there are no existing manipulation anomaly benchmarks, the performance of the CLUE-AI framework is evaluated on the data set collected by the Baxter robot in the Artificial Intelligence and Robotics Laboratory of Istanbul Technical University. A comparative analysis of the presented framework is also presented with the baseline methods. The results indicate that the framework has the ability to identify anomaly cases with an f-score of 94%.

7. CONCLUSIONS AND DISCUSSIONS

In this thesis, the anomaly identification problem is addressed for the robots performing everyday object manipulation tasks. The problem is investigated from different perspectives considering the processed feature types. To effectively solve this problem, two multimodal anomaly identification frameworks are introduced and presented in Chapters 5 and 6. The underlying idea behind the symbol-based anomaly identification framework (presented in Chapter 5) is to employ high-level domain symbols to classify anomalies that arise during task execution. These symbolic features are constructed from visual, auditory and proprioceptive sensor modalities that are gathered through an RGB-D camera, a microphone and a gripper, respectively. The visual features are extracted by a visual world modeling system, Violet [107]. The perceived sound data are classified with an SVM-based method [103]. Then, these tailored features are fused using an early fusion scheme, and they are fed into LSTM layers. The presented symbol-based anomaly identification framework is evaluated with a Baxter robot on real-world everyday object manipulation scenarios. A comparative analysis of the framework with GRU-based, HMM-based and CRF-based methods is presented. Furthermore, a multimodality analysis is also presented to show the contribution of each sensory modality on the identification performance. The results indicate that the presented symbol-based anomaly identification framework provides better results compared to the baselines with an f-score of 92% in the identification task due to capturing long-term dependencies effectively between anomaly indicators.

Chapter 6 presents and proposes a convolutional three-stream anomaly identification framework (CLUE-AI) to overcome the explained challenges as the second perspective. This framework consists of three streams each of which is a distinct type of sensory data, vision, audition and proprioception, respectively. Convolutional visual features are extracted and learned with a VGG16-LSTM-based method. Then, MFCC features are extracted from auditory stream, and it is passed through a CNN block designed for it instead of a classifier that returns high-level discreet classes. Afterward, proprioceptive data are processed with a CNN block. The results obtained from each sensory modality processing are combined with a late fusion scheme. The framework is tested on the Baxter robot with an extended anomaly set compared to the one used by the symbol-based anomaly identification framework, and analyses including performance, comparison, training and multimodality are presented. The CLUE-AI framework achieves an f-score of 94% in classifying anomaly types. Considering the obtained results of the symbol-based anomaly identification and the CLUE-AI frameworks, it can be inferred that CLUE-AI w/ VGG16-LSTM outperforms the symbol-based solution. The upcoming section presents a discussion of the investigated problem from several aspects.

7.1 Discussion

The symbol-based anomaly identification framework presented in Chapter 5 necessitates the usage of a visual world modeler and a classifier to map detected sounds. Such systems may not always be present in a robot framework, and these systems may require domain expertise and feature engineering. In such cases, learning from only raw sensory data is more beneficial when preprocessing is not possible (i.e., a visual modeling system for visual data). Furthermore, in highly cluttered environments, this preprocessing may become expensive and processing convolutional features is more efficient. In order to cope with this and avoid such necessities, a framework (CLUE-AI) that takes into account convolutional features without a need for high-level feature engineering is presented in Chapter 6. Processing convolutional features, as the CLUE-AI framework does, is a less costly solution without additional feature tailoring, and it provides better results.

The presented frameworks in Chapters 5-6 construct anomaly models to learn anomalies with a training phase. After these models are constructed, they are then used for identifying anomaly cases. On the other hand, it is almost impossible to predict all potential anomalies that may arise in unstructured environments, and unknown anomalies that are not encountered before are likely to occur in such environments. Consequently, in case of an occurrence of an unknown anomaly, such methods would classify it by taking into account the common indicators of the anomaly.

A robot may always fail on a particular task/operation in a particular context. For instance, let us assume that a service robot always fails in opening a specific type of jar without any help from other robots in the environment. Relevant hypotheses can be constructed by invoking a learning algorithm to associate the corresponding anomaly to the anomaly context [11]. The robot then may ask for help in upcoming tasks if one of the constructed hypotheses indicates so.

The variety of sensor modalities that are processed is crucial for the identification task. Since the anomaly identification problem builds on processing sensor modalities to capture anomaly indicators, the amount of data related to the environment depends on the processed modalities. Different sensors may help in perceiving a different environmental clue on an anomaly. For example, an odor sensor may help in the classification of anomaly cases related to olfactory anomalies.

7.2 Limitations and Future Work

The anomaly identification problem in cognitive robotics is still wide open for further research, and many aspects of the problem are still under investigation. The solutions presented in this thesis can be extended for further improvements. Some possible future directions are listed as follows:

- Considering the situation that the proposed frameworks in this thesis have the ability to identify modeled anomaly cases, the framework may be extended in such a way that it is also capable of dealing with unknown anomalies that the robot has not encountered before. To cope with the occurrence of unknown anomalies, an online learning procedure of the corresponding model can take place after the unknown anomaly is labeled by a domain expert.
- It may be likely that multiple anomalies occur simultaneously during execution. To handle such cases, multiple models may be trained for each anomaly class to identify such cases [60]. Such a technique may not scale well with the increasing number of anomalies. Considering the scores for each anomaly class returned by a single model, and deciding on the anomaly type accordingly may be yet another solution to address such issues.

- The anomaly identification problem is handled from a multi-class classification perspective in this research. This problem can be represented and addressed as a casual reasoning problem to infer the root causes of potential anomaly cases.

7.3 Impact

This Ph.D. research investigates an important problem in cognitive robotics to increase the self-awareness capabilities of the service robots. It presents an important step to equip robots to perform everyday object manipulation tasks with awareness procedures for safety in unstructured environments. In such environments, continuous perception of the environment is essential. Therefore, such a requirement necessitates processing these perceived observations to be monitored carefully to detect a potential anomaly that may threaten and violate the safety of the environment, robots or humans. In case of an anomaly, this situation needs to be identified for efficient recovery. Ethical issues also need to be taken into account for service robots, especially for the cases where other robots or humans exist in the environment. Therefore, the robot should avoid potential anomaly cases that may lead to failures resulting in harming itself or the others in the environment. Consequently, the utilization of such diagnostic procedures contributes to robots fulfilling robust and safe task execution.

REFERENCES

[1] **Altan, D.** and **Sariel, S.** (2021). What went wrong? Identification of everyday object manipulation anomalies, *Intelligent Service Robotics*, *14*(2), 215–234, https://doi.org/10.1007/s11370-021-00355-w.

[2] **Toledo, T.** and **Katz, R.** (2009). State dependence in lane-changing models, *Transportation research record*, *2124*(1), 81–88.

[3] **Ersen, M.**, **Oztop, E.**, and **Sariel, S.** (2017). Cognition-Enabled Robot Manipulation in Human Environments: Requirements, Recent Work, and Open Problems, *IEEE Robotics Automation Magazine*, *24*(3), 108–122.

[4] **Ackerman, E.** (2021). It's (Still) Really Hard for Robots to Autonomously Do Household Chores, https://spectrum.ieee.org/hard-for-robots-autonomous-household-chores.

[5] **Grinbaum, A.**, **Chatila, R.**, **Devillers, L.**, **Ganascia, J.**, **Tessier, C.**, and **Dauchet, M.** (2017). Ethics in Robotics Research: CERNA Mission and Context, *IEEE Robotics Automation Magazine*, *24*(3), 139–145.

[6] **Lin, P.**, **Abney, K.**, and **Bekey, G.** (2011). Robot ethics: Mapping the issues for a mechanized world, *Artificial Intelligence*, *175*(5-6), 942–949.

[7] **Zacharaki, A.**, **Kostavelis, I.**, **Gasteratos, A.**, and **Dokas, I.** (2020). Safety bounds in human robot interaction: A survey, *Safety Science*, *127*, 104667, https://www.sciencedirect.com/science/article/pii/S0925753520300643.

[8] **Ravichandar, H.**, **Polydoros, A. S.**, **Chernova, S.**, and **Billard, A.** (2020). Recent advances in robot learning from demonstration, *Annual Review of Control, Robotics, and Autonomous Systems*, *3*, 297–330.

[9] **Maxion, R. A.** (1990). Anomaly detection for diagnosis, *Digest of Papers. Fault-Tolerant Computing: 20th International Symposium*, IEEE Computer Society, pp.20–21.

[10] **Isermann, R.** and **Ballé, P.** (1997). Trends in the application of model-based fault detection and diagnosis of technical processes, *Control engineering practice*, *5*(5), 709–719.

[11] **Karapinar, S.** and **Sariel, S.** (2015). Cognitive robots learning failure contexts through real-world experimentation, *Autonomous Robots*, *39*(4), 469–485, https://doi.org/10.1007/s10514-015-9471-y.

[12] **Carlson, J.**, **Murphy, R. R.**, and **Nelson, A.** (2004). Follow-up analysis of mobile robot failures, *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 5, IEEE, pp.4987–4994.

[13] **Karapinar, S.**, **Altan, D.**, and **Sariel-Talay, S.** (2012). A robust planning framework for cognitive robots, *Proceedings of the AAAI-12 workshop on cognitive robotics (CogRob).*

[14] **Lopez, F.**, **Saez, M.**, **Shao, Y.**, **Balta, E. C.**, **Moyne, J.**, **Mao, Z. M.**, ..., and **Tilbury, D.** (2017). Categorization of anomalies in smart manufacturing systems to support the selection of detection mechanisms, *IEEE Robotics and Automation Letters*, *2*(4), 1885–1892.

[15] **Fritz, C.** (2005). *Execution Monitoring – A Survey*, (technical report), University of Toronto.

[16] **Pettersson, O.** (2005). Execution monitoring in robotics: A survey, *Robotics and Autonomous Systems*, *53*, 73–88.

[17] **Isermann, R.** (2005). Model-based fault-detection and diagnosis–status and applications, *Annual Reviews in control*, *29*(1), 71–85.

[18] **Chandola, V.**, **Banerjee, A.**, and **Kumar, V.** (2009). Anomaly detection: A survey, *ACM computing surveys (CSUR)*, *41*(3), 1–58.

[19] **Khalastchi, E.** and **Kalech, M.** (2018). On fault detection and diagnosis in robotic systems, *ACM Computing Surveys (CSUR)*, *51*(1), 1–24.

[20] **Pettersson, O.**, **Karlsson, L.**, and **Saffiotti, A.** (2007). Model-free execution monitoring in behavior-based robotics, *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, *37*(4), 890–901.

[21] **Mendoza, J. P.**, **Veloso, M. M.**, and **Simmons, R.** (2012). Mobile robot fault detection based on redundant information statistics.

[22] **Fourlas, G. K.**, **Karkanis, S.**, **Karras, G. C.**, and **Kyriakopoulos, K. J.** (2014). Model based actuator fault diagnosis for a mobile robot, *Industrial Technology (ICIT), 2014 IEEE International Conference on*, IEEE, pp.79–84.

[23] **Stavrou, D.**, **Eliades, D. G.**, **Panayiotou, C. G.**, and **Polycarpou, M. M.** (2015). Fault detection for service mobile robots using model-based method, *Autonomous Robots*, 1–12.

[24] **Orendt, E. M.** and **Henrich, D.** (2015). Design of robust robot programs: Deviation detection and classification using entity-based resources, *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, pp.1704–1710.

[25] **Khalastchi, E.**, **Kalech, M.**, **Kaminka, G. A.**, and **Lin, R.** (2015). Online data-driven anomaly detection in autonomous robots, *Knowledge and Information Systems*, *43*(3), 657–688.

[26] **Mendoza, J. P.**, **Veloso, M.**, and **Simmons, R.** (2015). Plan execution monitoring through detection of unmet expectations about action outcomes, *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp.3247–3252.

[27] **Mendoza, J. P.**, **Veloso, M.**, and **Simmons, R.** (2014). Focused optimization for online detection of anomalous regions, *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, IEEE, pp.3358–3363.

[28] **Rubinstein, R.** (1999). The cross-entropy method for combinatorial and continuous optimization, *Methodology and computing in applied probability*, *1*(2), 127–190.

[29] **Häussermann, K.**, **Zweigle, O.**, and **Levi, P.** (2015). A Novel Framework for Anomaly Detection of Robot Behaviors, *Journal of Intelligent & Robotic Systems*, *77*(2), 361–375.

[30] **Hovland, G. E.** and **McCarragher, B. J.** (1998). Hidden Markov models as a process monitor in robotic assembly, *The International Journal of Robotics Research*, *17*(2), 153–168.

[31] **Bouguerra, A.**, **Karlsson, L.**, and **Saffiotti, A.** (2007). Handling uncertainty in semantic-knowledge based execution monitoring, *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, IEEE, pp.437–443.

[32] **Park, D.**, **Erickson, Z.**, **Bhattacharjee, T.**, and **Kemp, C. C.** (2016). Multimodal execution monitoring for anomaly detection during robot manipulation, *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp.407–414.

[33] **Park, D.**, **Kim, H.**, and **Kemp, C. C.** (2018). Multimodal anomaly detection for assistive robots, *Autonomous Robots*, 1–19.

[34] **Azzalini, D.**, **Castellini, A.**, **Luperto, M.**, **Farinelli, A.**, and **Amigoni, F.** (2020). HMMs for Anomaly Detection in Autonomous Robots, *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pp.105–113.

[35] **Di Lello, E.**, **De Laet, T.**, and **Bruyninckx, H.** (2012). Hierarchical dirichlet process hidden markov models for abnormality detection in robotic assembly, *Neural Information Processing Systems, Date: 2012/12/03-2012/12/08, Location: Lake Tahoe, Nevada*.

[36] **Goel, P.**, **Dedeoglu, G.**, **Roumeliotis, S. I.**, and **Sukhatme, G. S.** (2000). Fault detection and identification in a mobile robot using multiple model estimation and neural network, *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 3, IEEE, pp.2302–2309.

[37] **Olivato, M.**, **Cotugno, O.**, **Brigato, L.**, **Bloisi, D.**, **Farinelli, A.**, and **Iocchi, L.** (2019). A Comparative Analysis on the use of Autoencoders for Robot Security Anomaly Detection, *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp.984–989.

[38] **Wellhausen, L.**, **Ranftl, R.**, and **Hutter, M.** (2020). Safe Robot Navigation via Multi-Modal Anomaly Detection, *IEEE Robotics and Automation Letters*, *5*(2), 1326–1333.

[39] **Park, D.**, **Hoshi, Y.**, and **Kemp, C. C.** (2018). A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder, *IEEE Robotics and Automation Letters*, *3*(3), 1544–1551.

[40] **Ullah, W.**, **Ullah, A.**, **Haq, I. U.**, **Muhammad, K.**, **Sajjad, M.**, and **Baik, S. W.** (2020). CNN features with bi-directional LSTM for real-time anomaly detection in surveillance networks, *Multimedia Tools and Applications*, 1–17.

[41] **Adam, S.**, **Larsen, M.**, **Jensen, K.**, and **Schultz, U. P.** (2014). Towards rule-based dynamic safety monitoring for mobile robots, *Simulation, Modeling, and Programming for Autonomous Robots*, 207–218.

[42] **Kapotoglu, M.**, **Koc, C.**, **Sariel, S.**, and **Ince, G.** (2014). Action Monitoring in Cognitive Robots (in Turkish), *The 22nd Signal Processing and Communications Applications Conference (SIU)*, pp.2154–2157.

[43] **Kvarnström, J.**, **Heintz, F.**, and **Doherty, P.** (2008). A Temporal Logic-Based Planning and Execution Monitoring System., *ICAPS*, pp.198–205.

[44] **Doherty, P.**, **Kvarnström, J.**, and **Heintz, F.** (2009). A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems, *Autonomous Agents and Multi-Agent Systems*, *19*(3), 332–377.

[45] **Micalizio, R.** (2013). Action Failure Recovery via Model-based Diagnosis and Conformant Planning, *Computational Intelligence*, *29*(2), 233–280.

[46] **McCarthy, J.** (1963). *Situations, actions, and causal laws*, Technical Report, DTIC Document.

[47] **Gspandl, S.**, **Podesser, S.**, **Reip, M.**, **Steinbauer, G.**, and **Wolfram, M.** (2012). A dependable perception-decision-execution cycle for autonomous robots, *IEEE International Conference on Robotics and Automation (ICRA)*, pp.2992–2998.

[48] **Steinbauer, G.** and **Wotawa, F.** (2009). Robust plan execution using model-based reasoning, *Advanced robotics*, *23*(10), 1315–1326.

[49] **Schleyer, G.** and **Russell, R. A.** (2011). Disturbance and failure classification in walking robots, *Australasian Conference on Robotics and Automation 2011 (Tom Drummond 7 December 2011 to 9 December 2011)*, Australian Robotics and Automation Association, pp.1–8.

[50] **Abid, A.**, **Khan, M. T.**, and **de Silva, C.** (2015). Fault detection in mobile robots using sensor fusion, *Computer Science & Education (ICCSE), 2015 10th International Conference on*, IEEE, pp.8–13.

[51] **Verma, V.**, **Gordon, G.**, **Simmons, R.**, and **Thrun, S.** (2004). Real-time fault diagnosis [robot fault diagnosis], *Robotics Automation Magazine, IEEE, 11*(2), 56–66.

[52] **Verma, V.**, **Simmons, R.**, and **Fernandez, J.** (2002). Probabilistic models for monitoring and fault diagnosis, *The Second IARP and IEEE/RAS Joint Workshop on Technical Challenges for Dependable Robots in Human Environments*, pp.43–67.

[53] **Flores-Quintanilla, J.**, **Morales-Menendez, R.**, **Ramirez-Mendoza, R.**, **Garza-Castanon, L.**, and **Cantu-Ortiz, F.** (2005). Towards a new fault diagnosis system for electric machines based on dynamic probabilistic models, *Proceedings of the 2005 American Control Conference*.

[54] **Rigatos, G. G.** (2009). Particle and Kalman filtering for fault diagnosis in DC motors, *Vehicle Power and Propulsion Conference, 2009. VPPC'09. IEEE*, IEEE, pp.1228–1235.

[55] **Di Lello, E.**, **Klotzbucher, M.**, **De Laet, T.**, and **Bruyninckx, H.** (2013). Bayesian time-series models for continuous fault detection and recognition in industrial robotic tasks, *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, IEEE, pp.5827–5833.

[56] **Luo, S.**, **Wu, H.**, **Lin, H.**, **Duan, S.**, **Guan, Y.**, and **Rojas, J.** (2018). Fast, robust, and versatile event detection through hmm belief state gradient measures, *IEEE Intern:(in–print)*.

[57] **Wu, H.**, **Guan, Y.**, and **Rojas, J.** (2019). A Latent State-Based Multimodal Execution Monitor with Anomaly Detection and Classification for Robot Introspection, *Applied Sciences*, *9*(6), 1072, `http://dx.doi.org/10.3390/app9061072`.

[58] **Wu, H.**, **Guan, Y.**, and **Rojas, J.** (2019). Analysis of multimodal Bayesian nonparametric autoregressive hidden Markov models for process monitoring in robotic contact tasks, *International Journal of Advanced Robotic Systems*, *16*(2), 1729881419834840.

[59] **Altan, D.** and **Sariel-Talay, S.** (2014). Probabilistic Failure Isolation for Cognitive Robots, *Proc. Florida Artificial Intelligence Research Society Conference (FLAIRS)*.

[60] **Altan, D.** and **Sariel, S.** (2016). Empirical Analysis of Probabilistic Methods for Failure Isolation in Robots, *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.(IROS) Workshop Cognit. Robot.(CogRob)*.

[61] **Bowkett, J.**, **Burdick, J.**, **Matthies, L.**, and **Detry, R.** (2018). Semantic Understanding of Task Outcomes: Visually Identifying Failure Modes Autonomously Discovered in Simulation, *Representing a Complex World:*

Perception, Inference, and Learning for Joint Semantic, Geometric, and Physical Understanding (ICRA 2018 Workshop).

[62] **Lu, C.**, **Wang, Z.-Y.**, **Qin, W.-L.**, and **Ma, J.** (2017). Fault diagnosis of rotary machinery components using a stacked denoising autoencoder-based health state identification, *Signal Processing*, *130*, 377–388.

[63] **Wen, L.**, **Gao, L.**, and **Li, X.** (2019). A New Deep Transfer Learning Based on Sparse Auto-Encoder for Fault Diagnosis, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *49*(1), 136–144.

[64] **Park, D.**, **Kim, H.**, **Hoshi, Y.**, **Erickson, Z.**, **Kapusta, A.**, and **Kemp, C. C.** (2017). A multimodal execution monitor with anomaly classification for robot-assisted feeding, *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp.5406–5413.

[65] **Baghernezhad, F.** and **Khorasani, K.** (2016). Computationally intelligent strategies for robust fault detection, isolation, and identification of mobile robots, *Neurocomputing*, *171*, 335–346.

[66] **Göbelbecker, M.**, **Keller, T.**, **Eyerich, P.**, **Brenner, M.**, and **Nebel, B.** (2010). Coming Up With Good Excuses: What to do When no Plan Can be Found., *Cognitive Robotics*, *10081*.

[67] **Elmoufidi, A.**, **El Fahssi, K.**, **Jai-Andaloussi, S.**, **Sekkaki, A.**, **Gwenole, Q.**, and **Lamard, M.** (2017). Anomaly classification in digital mammography based on multiple-instance learning, *IET Image Processing*, *12*(3), 320–328.

[68] **Scime, L.** and **Beuth, J.** (2018). Anomaly detection and classification in a laser powder bed additive manufacturing process using a trained computer vision algorithm, *Additive Manufacturing*, *19*, 114 – 126.

[69] **Meng, W.**, **Liu, Y.**, **Zhang, S.**, **Pei, D.**, **Dong, H.**, **Song, L.** and **Luo, X.** (2018). Device-agnostic log anomaly classification with partial labels, *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, IEEE, pp.1–6.

[70] **Usug, C. U.**, **Altan, D.**, and **Sariel-Talay, S.** (2012). Robots that create alternative plans against failures, *Robot Control*, volume 10, pp.461–466.

[71] **Bacchus, F.** and **Ady, M.** (2001). Planning with resources and concurrency: A forward chaining approach, *IJCAI*, volume 1, Citeseer, pp.417–424.

[72] **Tang, J.**, **Zhang, J.**, **Wang, X.**, **Xia, Z.**, **Hu, Y.**, and **Zhang, J.** (2013). Gaining diagnostic information for fault isolation, *Information and Automation (ICIA), 2013 IEEE International Conference on*, IEEE, pp.665–670.

[73] **Fox, M.**, **Gerevini, A.**, **Long, D.**, and **Serina, I.** (2006). Plan Stability: Replanning versus Plan Repair., *ICAPS*, volume 6, pp.212–221.

[74] **Gerevini, A.**, **Saetti, A.**, and **Serina, I.** (2003). Planning through stochastic local search and temporal action graphs in LPG, *Journal of Artificial Intelligence Research*, 239–290.

[75] **Van Der Krogt, R.** and **De Weerdt, M.** (2005). Plan repair as an extension of planning, *ICAPS*, volume 5, pp.161–170.

[76] **Younes, H. L.** and **Simmons, R. G.** (2003). VHPOP: Versatile heuristic partial order planner, *Journal of Artificial Intelligence Research*, 405–430.

[77] **Gerevini, A.** and **Serina, I.** (2000). Fast Plan Adaptation through Planning Graphs: Local and Systematic Search Techniques., *AIPS*, pp.112–121.

[78] **Bechon, P.**, **Barbier, M.**, **Lesire, C.**, **Infantes, G.**, and **Vidal, V.** (2015). Using hybrid planning for plan reparation, *Mobile Robots (ECMR), 2015 European Conference on*, IEEE, pp.1–6.

[79] **Bechon, P.**, **Barbier, M.**, **Infantes, G.**, **Lesire, C.**, and **Vidal, V.** (2014). HiPOP: Hierarchical Partial-Order Planning., *STAIRS*, pp.51–60.

[80] **Schmill, M.**, **Josyula, D.**, **Anderson, M. L.**, **Wilson, S.**, **Oates, T.**, **Perlis, D.** and **Fults, S.** (2007). Ontologies for reasoning about failures in AI systems, *Proceedings from the Workshop on Metareasoning in Agent Based Systems at the Sixth International Joint Conference on Autonomous Agents and Multiagent Sytems*.

[81] **Brandstötter, M.**, **Hofbaur, M. W.**, **Steinbauer, G.**, and **Wotawa, F.** (2007). Model-based fault diagnosis and reconfiguration of robot drives, *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, IEEE, pp.1203–1209.

[82] **Chang, G.** and **Kulic, D.** (2013). Robot task error recovery using Petri nets learned from demonstration, *Advanced Robotics (ICAR), 2013 16th International Conference on*, IEEE, pp.1–6.

[83] **Peynot, T.**, **Fitch, R.**, **McAllister, R.**, and **Alempijevic, A.** (2011). Autonomous Reconfiguration of a Multi-Modal Mobile Robot, *Workshop on Automated Diagnosis, Repair and Re-Configuration of Robot Systems, IEEE International Conference on Robotics and Automation (ICRA)*.

[84] **Morais, M. G.**, **Meneguzzi, F. R.**, **Bordini, R. H.**, and **Amory, A. M.** (2015). Distributed fault diagnosis for multiple mobile robots using an agent programming language, *Advanced Robotics (ICAR), 2015 International Conference on*, IEEE, pp.395–400.

[85] **Cantrell, R.**, **Talamadupula, K.**, **Schermerhorn, P.**, **Benton, J.**, **Kambhampati, S.**, and **Scheutz, M.** (2012). Tell me when and why to do it! Run-time planner model updates via natural language instruction, *Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on*, pp.471–478.

[86] **Nan, C.**, **Khan, F.**, and **Iqbal, M. T.** (2008). Real-time fault diagnosis using knowledge-based expert system, *Process Safety and Environmental Protection*, *86*(1), 55 – 71.

[87] **Url-1**, `https://www.rethinkrobotics.com/`, data retrieved 1-June-2021.

[88] **Quigley, M.**, **Conley, K.**, **Gerkey, B.**, **Faust, J.**, **Foote, T.**, **Leibs, J.**, ..., and **Ng, A. Y.** (2009). ROS: an open-source Robot Operating System, *ICRA workshop on open source software*, volume 3, Kobe, Japan, p. 5.

[89] **Chitta, S.**, **Sucan, I.**, and **Cousins, S.** (2012). Moveit![ros topics], *IEEE Robotics & Automation Magazine*, *19*(1), 18–19.

[90] **Rojas, R.** (1996). *Neural networks : a systematic introduction*, Springer-Verlag, Berlin ;New York.

[91] **Du, K.-L.** and **Swamy, M. N.** (2013). *Neural networks and statistical learning*, Springer Science & Business Media.

[92] **Hochreiter, S.** and **Schmidhuber, J.** (1997). Long short-term memory, *Neural computation*, *9*(8), 1735–1780.

[93] **Zhang, J.**, **Li, Y.**, **Tian, J.**, and **Li, T.** (2018). LSTM-CNN Hybrid Model for Text Classification, *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pp.1675–1680.

[94] **Xia, K.**, **Huang, J.**, and **Wang, H.** (2020). LSTM-CNN Architecture for Human Activity Recognition, *IEEE Access*, *8*, 56855–56866.

[95] **Troiano, L.**, **Villa, E. M.**, and **Loia, V.** (2018). Replicating a Trading Strategy by Means of LSTM for Financial Industry Applications, *IEEE Transactions on Industrial Informatics*, *14*(7), 3226–3234.

[96] **You, X.**, **Li, X.**, **Xu, Y.**, **Feng, H.**, **Zhao, J.**, and **Yan, H.** (2020). Toward Packet Routing With Fully Distributed Multiagent Deep Reinforcement Learning, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 1–14.

[97] **Cho, K.**, **Van Merriënboer, B.**, **Gulcehre, C.**, **Bahdanau, D.**, **Bougares, F.**, **Schwenk, H.** and **Bengio, Y.** (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation, *arXiv preprint arXiv:1406.1078*.

[98] **Bahdanau, D.**, **Cho, K.**, and **Bengio, Y.** (2014). Neural machine translation by jointly learning to align and translate, *arXiv preprint arXiv:1409.0473*.

[99] **Luong, M.-T.**, **Pham, H.**, and **Manning, C. D.** (2015). Effective approaches to attention-based neural machine translation, *arXiv preprint arXiv:1508.04025*.

[100] **Vaswani, A.**, **Shazeer, N.**, **Parmar, N.**, **Uszkoreit, J.**, **Jones, L.**, **Gomez, A. N.**, ..., and **Polosukhin, I.** (2017). Attention is all you need, *Advances in neural information processing systems*, pp.5998–6008.

[101] **Baum, L. E.** and **Petrie, T.** (1966). Statistical inference for probabilistic functions of finite state Markov chains, *The annals of mathematical statistics*, *37*(6), 1554–1563.

[102] **Lafferty, J. D.**, **McCallum, A.**, and **Pereira, F. C. N.** (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data, *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, p.282–289.

[103] **Inceoglu, A.**, **Ince, G.**, **Yaslan, Y.**, and **Sariel, S.** (2018). Comparative assessment of sensing modalities on manipulation failure detection, *IEEE ICRA Workshop on Perception, Inference and Learning for Joint Semantic, Geometric and Physical Understanding*.

[104] **Inceoglu, A.**, **Ince, G.**, **Yaslan, Y.**, and **Sariel, S.** (2018). Failure Detection Using Proprioceptive, Auditory and Visual Modalities, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp.2491–2496.

[105] **Han, W.**, **Chan, C.-F.**, **Choy, C.-S.**, and **Pun, K.-P.** (2006). An efficient MFCC extraction method in speech recognition, *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, IEEE, pp.4–pp.

[106] **Saltali, I.**, **Sariel, S.**, and **Ince, G.** (2016). Scene Analysis Through Auditory Event Monitoring, *Proceedings of the International Workshop on Social Learning and Multimodal Interaction for Designing Artificial Agents*, DAA '16, ACM, New York, NY, USA, pp.5:1–5:6, `http://doi.acm.org/10.1145/3005338.3005343`.

[107] **Inceoglu, A.**, **Koc, C.**, **Kanat, B. O.**, **Ersen, M.**, and **Sariel, S.** (2018). Continuous Visual World Modeling for Autonomous Robot Manipulation, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 1–14.

[108] **Hinterstoisser, S.**, **Cagniart, C.**, **Ilic, S.**, **Sturm, P.**, **Navab, N.**, **Fua, P.** and **Lepetit, V.** (2012). Gradient response maps for real-time detection of textureless objects, *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *34*(5), 876–888.

[109] **Ersen, M.**, **Sariel-Talay, S.**, and **Yalcin, H.** (2013). Extracting Spatial Relations Among Objects for Failure Detection, *on Visual and Spatial Cognition*, 13.

[110] **Trevor, A. J. B.**, **Gedikli, S.**, **Rusu, R. B.**, and **Christensen, H. I.** (2013). Efficient Organized Point Cloud Segmentation with Connected Components, *3rd Workshop on Semantic Perception, Mapping and Exploration (SPME)*, Karlsruhe, Germany.

[111] **Crammer, K.**, **Kulesza, A.**, and **Dredze, M.** (2009). Adaptive regularization of weight vectors, *Advances in neural information processing systems*, pp.414–422.

[112] **Saputro, D. R. S.** and **Widyaningsih, P.** (2017). Limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method for the parameter estimation on geographically weighted ordinal logistic regression model (GWOLR), *AIP Conference Proceedings*, volume1868, AIP Publishing, p.040009.

[113] **Okazaki, N.**, (2007), CRFsuite: a fast implementation of Conditional Random Fields (CRFs), `http://www.chokkan.org/software/crfsuite/`.

[114] **Kingma, D. P.** and **Ba, J.** (2014). Adam: A Method for Stochastic Optimization, *CoRR*, *abs/1412.6980*, `http://arxiv.org/abs/1412.6980`, `1412.6980`.

[115] **He, K.**, **Zhang, X.**, **Ren, S.**, and **Sun, J.** (2016). Deep residual learning for image recognition, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp.770–778.

[116] **Krizhevsky, A.**, **Sutskever, I.**, and **Hinton, G. E.** (2012). Imagenet classification with deep convolutional neural networks, *Advances in neural information processing systems*, *25*, 1097–1105.

[117] **Simonyan, K.** and **Zisserman, A.** (2014). Very deep convolutional networks for large-scale image recognition, *arXiv preprint arXiv:1409.1556*.

[118] **Russakovsky, O.**, **Deng, J.**, **Su, H.**, **Krause, J.**, **Satheesh, S.**, **Ma, S.**, ..., and **Fei-Fei, L.** (2015). ImageNet Large Scale Visual Recognition Challenge, *International Journal of Computer Vision (IJCV)*, *115*(3), 211–252.

[119] **McFee, B.**, **Raffel, C.**, **Liang, D.**, **Ellis, D. P.**, **McVicar, M.**, **Battenberg, E.** and **Nieto, O.** (2015). librosa: Audio and music signal analysis in python, *Proceedings of the 14th python in science conference*, volume 8, Citeseer, pp.18–25.

[120] **Selvaraju, R. R.**, **Cogswell, M.**, **Das, A.**, **Vedantam, R.**, **Parikh, D.**, and **Batra, D.** (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization, *Proceedings of the IEEE international conference on computer vision*, pp.618–626.

[121] **Devlin, J.**, **Chang, M.-W.**, **Lee, K.**, and **Toutanova, K.** (2018). Bert: Pre-training of deep bidirectional transformers for language understanding, *arXiv preprint arXiv:1810.04805*.

CURRICULUM VITAE

Name Surname: Doğan Altan

EDUCATION:

- **B.Sc.:** 2011, Istanbul Technical University, Faculty of Computer and Informatics Engineering, Computer Engineering
- **M.Sc.:** 2014, Istanbul Technical University, Graduate School of Science Engineering and Technology, Computer Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2021–ongoing Postdoctoral Fellow at Simula Research Laboratory, Oslo, Norway
- 2011–2021 Research Assistant at Istanbul Technical University, Faculty of Computer and Informatics Engineering, Istanbul, Turkey

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Altan, D.**, Sariel, S., 2022?. CLUE-AI: A convolutional three-stream anomaly identification framework for service robots. (under preparation for journal submission)
- **Altan, D.**, Sariel, S., 2021. What went wrong? Identification of everyday object manipulation anomalies. Intelligent Service Robotics. https://doi.org/10.1007/s11370-021-00355-w
- **Altan, D.**, Ak, A.C., Sariel, S., 2018. Bilişsel Robotlarda Gündelik Etkileşim Anomalilerinin Tanısı, Türkiye Robotbilim Konferansı (ToRK), 2018, Istanbul, Turkey.

OTHER PUBLICATIONS, PRESENTATIONS AND PATENTS:

- Akdogan, F., **Altan, D.**, Peker, E., Ustun, A., Sariel, S., 2020. Optimization of Real-World Outdoor Campaign Allocations, Turkish Journal of Electrical Engineering & Computer Sciences. DOI: 10.3906/elk-1907-122.
- **Altan, D.**, Sariel, S., 2016. Empirical Analysis of Probabilistic Methods for Failure Isolation in Robots, Tenth International Workshop on Cognitive Robotics (CogRob) at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, Daejeon, South Korea.

- Sariel, S, Yildiz, P., Karapinar, S., **Altan, D.**, Kapotoglu, M., 2015. Robust Task Execution through Experience-based Fuidance for Cognitive Robots, IEEE International Conference on Advanced Robotics (ICAR), 2015, Istanbul, Turkey.
- **Altan, D.**, Sariel, S., 2015. Bilişsel Robotlar için Olasılıksal Hata Tanısı, Türkiye Robotbilim Konferansı (ToRK), 2015, Istanbul, Turkey.
- **Altan, D.**, Sariel, S., 2014. Diagnosis of action execution failures for cognitive robots, 22nd IEEE Signal Processing and Communications Applications Conference (SIU), 2014, Trabzon, Turkey.
- **Altan, D.**, Sariel-Talay, S., 2014. Probabilistic Failure Isolation for Cognitive Robots, Florida Artificial Intelligence Research Society Conference (FLAIRS), 2014, Pensacola Beach, FL, USA.
- **Altan, D.**, Sariel-Talay, S., 2014. Hierarchical HMM-based Failure Isolation for Cognitive Robots, 6th International Conference on Agents and Artificial Intelligence (ICAART), 2014, Angers, France.
- Usug, C.U., **Altan, D.**, Sariel-Talay, S., 2012. Robots that create alternative plans against failures, 10th IFAC Symposium on Robot Control (SYROCO), 2012, Dubrovnik, Croatia.
- Karapinar, S., **Altan, D.**, Sariel-Talay, S., 2012. A robust planning framework for cognitive robots, International Workshop on Cognitive Robotics (CogRob) at the Twenty-Sixth AAAI Conference on Artificial Intelligence, 2012, Toronto, ON, Canada.
- Usug, C.U., **Altan, D.**, Karapinar, S., Ersen, M., Sariel-Talay, S., 2012. A robust planning framework for cognitive robots, 5th International Conference on Cognitive Systems, 2012, Vienna, Austria.