UNIVERSITÄT BONN

Visual Failure Detection in Robotics

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich–Wilhelms–Universität Bonn

vorlegt von

Santosh George Thoduka

aus

Bangalore, Indien

Bonn, 2025

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich–Wilhelms–Universität Bonn.

Gutachter / Betreuer: Prof. Dr. Jürgen Gall

Gutachter: Prof. Dr. Paul G. Plöger

Tag der Promotion: 26.03.2026

Erscheinungsjahr: 2026

Abstract

Autonomous robots in human-centric environments can encounter unforeseen situations, making them prone to task execution failures. These failures can lead to unsafe conditions for both humans and robots and result in a loss of trust in robots. Therefore, robots should have the ability to prevent, detect, and respond to failures. In this thesis, we focus on the detection of failures, particularly using video data from the robot's camera. Several visual failure detection datasets have emerged in recent years; however, there is still a scarcity of datasets suitable for building general-purpose failure detection approaches. Existing work has shown the benefit of using multimodal data for failure detection, but determining the best data representation and fusion methods remains an open challenge. Additionally, although most approaches develop failure detection models for specific tasks, they often fail to incorporate task knowledge into the learning process. In our work, we introduce multimodal datasets and explore multimodal learning and task knowledge integration to enhance failure detection performance. We contribute two visual failure detection datasets: the Bookshelf dataset and the Handover Failure Detection dataset, which consist of various failures that occur when the robot is placing a book on a shelf and performing object handovers with people. For the Bookshelf dataset, we use video and proprioceptive data to detect anomalous situations by comparing expected and observed motions. For the Handover dataset and a visual-tactile dataset, we find that intermediate fusion of video, force-torque, tactile and proprioceptive features performs best. For the Handover dataset, video was found to be an essential modality, and learning to predict the human's and robot's actions as auxiliary tasks is also beneficial. Finally, we explore incorporating task knowledge to improve failure classification performance. We show that pre-processing video frames using the known temporal boundaries of the robot's actions and locations of objects in the scene improves results on a large-scale failure dataset, and a variable frame rate-based data augmentation method shows further improvements. Our results highlight the importance of using multimodal data and task knowledge for failure detection. However, the task-specific nature of existing models makes it impractical to collect data and train a separate model for every task. Using simulators to generate large-scale video data is a viable approach to this problem in future work. The emergence of general-purpose vision-language-action models also presents opportunities for task-agnostic failure detection. Incorporating failure data into their training datasets and, similar to our work, making use of multimodal data and task knowledge are likely to further accelerate progress in this area.

Acknowledgements

This work in this thesis has been supported by many people, to whom I am very grateful. The topic itself was motivated by my advisor Prof. Paul G. Plöger and by the work of my colleagues at Hochschule Bonn-Rhein-Sieg (H-BRS), Anastassia, Iman and Alex, who have all explored aspects of failures in robotics. The discussions, ideas, and support from Prof. Plöger throughout the thesis always managed to get me back on track whenever I felt I was at a dead end. I'm also grateful to my advisor, Prof. Jürgen Gall, for his guidance throughout my PhD. His precise advice on my ongoing work often helped improve the focus and quality of the research. Prof. Nico Hochgeschwender and Prof. Sebastian Houben were also mentors to me during my thesis and supported my work in various ways.

I would also like to thank all my colleagues at H-BRS, including Deebul, Alex, Argentina, Wasil, Sven, Iman, Anastassia, Djordje, Tim, Minh, Youssef, Max, Michal and Jordan, with whom I've discussed my work at various points, including during our PhD seminars. I worked with several students on topics related to my thesis, all of which contributed to my own understanding of the topic. Amongst others, I'd like to especially thank Pritesh, Adithya, Bharath, Sogol, Ludovico, Anh-Duy, Huy, Rashid, Pranav, Anna, Chandrika, Abdelrahman, Amine, Akhilan and Zaid for their work in R&D projects, theses and software development projects.

The Handover Failure Detection dataset would not have been possible without the help of the volunteers who interacted with the robot. I'm thankful for their time and consent for publishing the data.

Three years of my PhD were supported by a scholarship from the Graduate Institute (GI) at H-BRS. The GI also supported my PhD in other ways by organizing workshops and poster sessions, for which I am very grateful. My time at H-BRS was supported by the Horizon 2020 projects ROPOD[1] and METRICS[2], and the nationally funded Garrulus project, which gave me the opportunity to work on my PhD alongside other tasks. The robots and the laboratory that I used during my thesis were supported by H-BRS and the Bonn-Aachen International Center for Information Technology (b-it). For training models, I extensively used the Platform for Scientific Computing at H-BRS, which is administered and maintained by Prof. Rudolf Berrendorf and Javed Razzaq. Their work in keeping the systems running smoothly made my work easier. Finally, the last phase of my PhD has been supported by Fraunhofer IAIS, where I have had the freedom to complete the thesis with the support of Prof. Houben.

[1] grant agreement No. 731848 ROPOD
[2] grant agreement No. 871252 (METRICS)

Contents

List of Figures

List of Tables

Acronyms

ARMBench	Amazon Robotic Manipulation Benchmark
CNN	convolutional neural network
FC	fully-connected
F-T	force-torque
H2R	human-to-robot
HFD	Handover Failure Detection
HMM	hidden Markov model
HSR	Human Support Robot
I3D	inflated 3D ConvNet
KL	Kullback-Leibler
LLM	large language model
LSTM	long short-term memory
MFCC	Mel-frequency Cepstral Coefficients
MLP	multi-layer perceptron
MSE	mean-squared error
MS-TCN	Multi-Stage Temporal Convolutional Network
MViT	Multiscale Vision Transformer
OF	optical flow
R2H	robot-to-human
ReLU	Rectified Linear Unit
RGB	Red Green Blue
ROI	region of interest
ROS	Robot Operating System
VAE	variational autoencoder
VLA	vision-language-action model
VLM	vision-language model
VT	Visual-Tactile

CHAPTER 1

Introduction

1.1 Motivation

Robots in manufacturing operate primarily in controlled environments, which means that there is very little variability in their operating conditions, the objects that they interact with, and potential failures that might occur. The robots are typically fixed to a single location and are usually confined to performing a small set of pre-programmed tasks. In contrast, autonomous robots that operate in more open and dynamic environments, such as our homes, can encounter a wide variety of situations that need to be managed. As a consequence, they are enabled with higher autonomy, which allows them to make decisions based on their current perception and understanding of the situation. Currently, the perception, decision making, and actuation capabilities of robots are not sufficient for them to safely negotiate all possible situations that they might encounter. This is especially true of situations that are not foreseen by the developers of the robot, which are highly likely to lead to task failures, unsafe conditions for humans and the robot, property damage, etc. For example, in certain situations, driverless cars get stuck in wet concrete, crash into fire hydrants and enter cordoned-off areas [10], robots fail to grasp objects [8], damage objects [3] and fail to interact effectively with humans [11]. The consequences of robotic failures can include injuries to humans [12], damage to the robot and the environment, incomplete tasks, and a loss of trust in robots [13]. This highlights the importance of ensuring robots are robust to failures, firstly to prevent failures if possible, and second to manage failures effectively when they do occur.

The research into robot failures addresses failure prevention [14–17], failure detection [9, 18–20], diagnosis to determine the cause of failures [21–23], generation of explanations to communicate failures to humans [24–26], and failure recovery to perform actions that repair failures [25, 27–29]. While failure prevention deals with minimizing failures, the remaining areas of research are focussed on managing failures that have already occurred, by detecting, diagnosing and recovering from them. Failures can be caused by internal faults in the robot (such as a broken motor), or external faults, which are caused by external conditions in the environment, imprecise perception and actuation of the robot, adversarial actions by humans, etc. [30]. Some examples of failures during robot task execution can be seen in Fig. 1.1.

In some cases, failures cannot be prevented. The middle scenario in Fig. 1.1 is one such

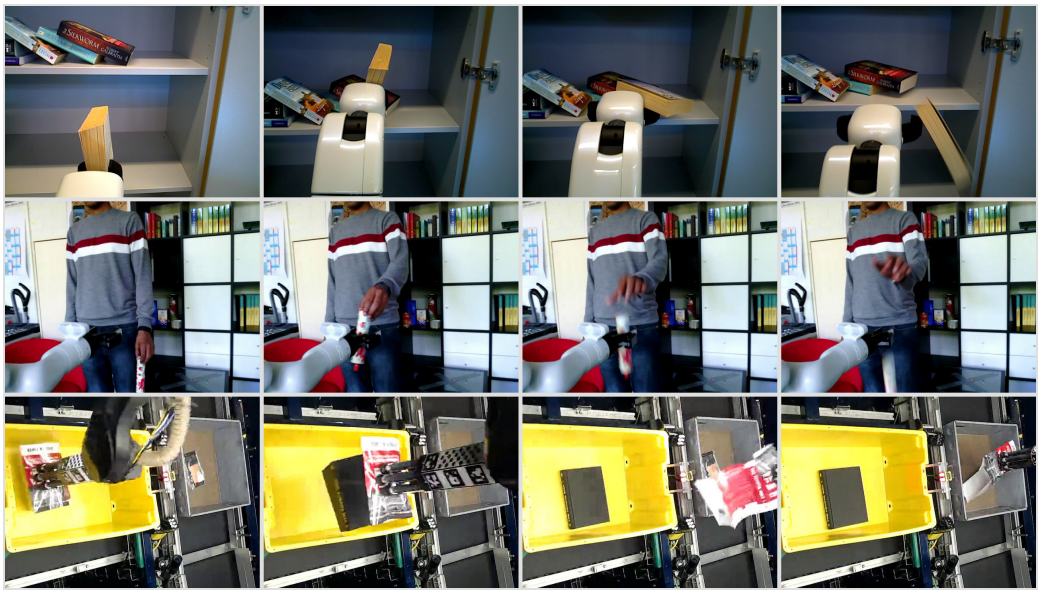


Figure 1.1: **Top:** a book falls off a shelf after the robot attempts to place it on top of another book [1], **Middle:** an object is dropped during a human-to-robot handover [2], **Bottom:** a book and its cover are separated as a robot attempts to pick up the book [3]. The full videos can be viewed by clicking on or scanning the QR code.

example. The person handing over the object to the robot drops the object before the robot has the chance to grasp it. In such cases, the robot must continue with the remaining failure management steps, namely, detecting the failure, diagnosing the reason and planning steps to recover from it. In other scenarios, failures may be caused by imprecise perception or manipulation (such as in the top example in Fig. 1.1, in which the robot fails to place the book sufficiently deep into the bookshelf), or the limited capabilities of the robot (such is in the bottom example in Fig. 1.1, where the grasping capabilities of the robot do not allow it to grasp the book and its cover in a stable manner). We are motivated by such scenarios in which failures *will* happen, and thus the robot *must* be capable of managing failures. In this thesis, we focus on failure detection, which is the first step of the failure management process.

An important part of developing robots is benchmarking and evaluating their performance in realistic scenarios, which includes subjecting them to conditions that might lead to failures. Similar to software testing, subjecting a robot to abnormal conditions and injecting faults during testing can be used to build arguments regarding its reliability and fault tolerance. Failure datasets are commonly used to evaluate *algorithms* for failure detection, while injecting failures during in-person tests using standardized test-beds [31–34] are used to evaluate the failure detection and management capabilities of robots with the failure detection algorithms integrated into the complete robot system [35]. We are interested in developing and improving failure detection algorithms, and therefore base our work on failure detection datasets. Integrating the developed failure detection methods into the robot's task execution and monitoring system is a necessary next step in the process, which is not our main focus in this thesis.

Video is a rich source of information for various tasks, and, as seen in Fig. 1.1, also contains sufficient information for failure detection. Due to the rapid development of vision-based learning methods, we are motivated to investigate vision-based failure detection in particular. Data-driven approaches for vision tasks such as object detection, segmentation and tracking, video analytics, and anomaly detection have progressed significantly since the introduction of deep convolutional neural networks (CNNs) [36] and vision transformers [5], and the availability of large-scale datasets. These methods have been applied to robotics as well, where images and videos from the robot's on-board camera can be used for tasks such as navigation [37], recognizing and anticipating actions of humans [38,39], grasp pose estimation [40], etc. Additionally, using video data with other available sensors on the robot, such as tactile and proprioceptive sensors, typically leads to more robust multimodal solutions for various tasks [41]. Therefore, in this thesis, we focus on vision-based failure detection methods, making use of the robot's camera as the primary source of data, while also considering multimodal approaches.

1.2 Problem Statement

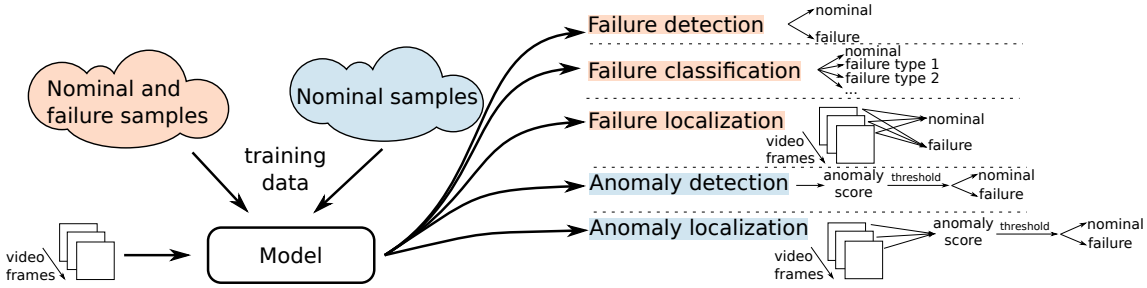


Figure 1.2: Depending on the training data and output of the model, the problem can be formulated as failure detection, failure classification, (temporal) failure localization, anomaly detection, and (temporal) anomaly localization. Orange indicates that both nominal and failure samples are available in the training data, whereas Blue indicates that only nominal samples are available during training.

The main problem addressed in this thesis is the visual detection of failures during the execution of tasks by a robot. For the purposes of this thesis, we define a failure as any outcome that does not fulfill the objectives of the task being performed. The objectives of a task may include those that directly achieve the robot's main goal, as well as auxiliary objectives related to factors such as safety, explainability and quality of social interaction. As an example, if a robot is tasked with fetching a coffee cup from the kitchen for a user, it would be considered a failure if the main objective is not achieved, namely, the user does not receive the coffee cup. However, it could also be considered a failure if the robot drops another object, collides with furniture, or sets the cup down without confirming whether the user wants it in their hand.

Consider a robot that performs a task $\mathcal{T}$, which is captured by the robot's camera as a video consisting of T frames, denoted by $I_{1:T}$. Given $I_{1:T}$, the goal of visual *failure detection* is classify the outcome of the task execution, $o \in \{\texttt{nominal}, \texttt{failure}\}$. In the case of multiple

failure types, the outcome can include the failure types, in addition to the nominal class, $o \in \{\texttt{nominal}, \texttt{failure type 1}, \texttt{failure type 2}, ..., \texttt{failure type N}\}$, which we term as *failure classification*.

For the purposes of failure diagnosis and recovery, it might be useful to know *when* the failure occurred, and thus we also consider the case of temporal failure localization, in which the objective is to classify each video frame with an outcome. The output is a sequence of outcomes, $o_{1:T} = (o_1, o_2, ..., o_T)$, for each frame, which could be binary or multi-class outcomes. For brevity, we use the term *failure localization* to refer to temporal failure localization, not to be confused with spatial failure localization, which is concerned with determining *where* the failure has occurred.

In many cases, the types of failures that could occur may not be known beforehand. For example, if *safe behaviour* is an auxiliary objective, there are numerous potential failure modes for this objective that cannot be anticipated beforehand. Therefore, the task of failure detection can also be formulated as an *anomaly detection* problem, where an anomaly is only an indicator that a failure *may* have occurred. The training data for anomaly detection only consists of nominal samples, as indicated in blue in Fig. 1.2. In this case, the goal is to learn a model of nominal executions using the available data, and to assign an anomaly score to the video (*anomaly detection*) or to individual frames of the video (*anomaly localization*). By applying a threshold to the anomaly score, the output can also be converted into o or $o_{1:T}$, similar to failure detection and failure localization. An illustration of these tasks can be seen in Fig. 1.2.

In this thesis, we consider the task of anomaly localization in Chapter 4, failure classification, detection and localization in Chapter 5, and failure classification in Chapter 6.

1.3 Opportunities and Challenges

In this section, we discuss the current opportunities and challenges related to vision-based failure detection for robotics, which we address in our work.

1.3.1 Datasets

There is a scarcity of datasets for visual failure detection in robotics. Most existing datasets are small-scale, task-specific and collected on a single robot [9, 15, 42]. We discuss existing datasets in more detail in Chapter 3.3. Since each robot has different sensors, and the relevance of the sensors depends on the task, the available multimodal data varies across datasets. This heterogeneity of available multimodal data makes it challenging to develop solutions that generalize to different robots and sensor configurations. Nevertheless, this presents an opportunity to contribute novel datasets, which we hope will advance the research in the field.

In our work, we contribute an anomaly localization dataset for the task of placing a book on a shelf in Chapter 4 and a multimodal handover failure detection dataset in Chapter 5. In Chapter 5 and Chapter 6, we also enhance the Visual-Tactile (VT) dataset [8] and the Amazon Robotic Manipulation Benchmark (ARMBench) dataset with additional annotations.

1.3.2 Multimodal Learning

Visual data from the robot's camera is a rich information source. Depending on the placement of the camera(s), the robot's video data can provide a view of the surrounding environment, the robot's arms and end-effectors and the objects that the robot is interacting with. Apart from colour images, motion and depth data may also be available. Robots typically have additional sensors that provide complementary data that could be useful for failure detection. Such sensors include tactile sensors, force-torque sensors, microphones, and proprioceptive sensors that measure the positions and velocities of the robot's joints. The availability of multimodal sensor data presents an opportunity for combining them with the visual data for more robust failure detection. Existing works [9, 19] already show the benefit of using multimodal data for anomaly and failure detection. The challenges of using multimodal data include determining suitable ways of encoding the data from each sensor, identifying multimodal fusion and learning approaches, and incorporating knowledge about the robot's embodiment (such as its 3D model and relative positioning of the sensors) into the fusion process.

In our work, we also propose approaches that use multimodal data. In Chapter 4, in addition to video data, we make use of proprioceptive data to model expected motions in the scene caused by the motion of the robot in nominal conditions. In Chapter 5, we present approaches that fuse video data with tactile and finger joint position information for failure localization, and video data with force-torque sensor data and gripper positions for failure detection.

1.3.3 Task Knowledge

A source of additional data that can be used to aid failure detection is the context surrounding the task that is being performed. Contextual task knowledge could include information such as past and current actions performed, locations of objects that are relevant for the task, expectations regarding the actions of humans involved in the task, time and location where the task is performed, etc. Using all available information could lead to improved failure detection performance. On the other hand, incorporating task-specific information could lead to solutions that are not generalizable to other tasks, which may necessitate developing different models for each task. The challenge is to develop general approaches that can apply to several tasks, which are still able to incorporate task-specific information at runtime.

In Chapter 5 and 6, we use available task information such as the temporal boundaries of the robot's actions, expectations regarding the actions of humans, and the location of task-relevant objects to enhance the performance of failure classification models.

1.4 Contributions

Parts of this thesis have been published in the following papers:

- S. Thoduka, J. Gall, and P. G. Plöger, "Using Visual Anomaly Detection for Task Execution Monitoring," in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2021, pp. 4604–4610, doi: https://doi.org/10.1109/

IROS51168.2021.9636133.
We contribute an anomaly localization dataset and motion anomaly localization approach in Chapter 4. Apart from learning to predict future nominal motions in the scene, we show that using the robot's proprioceptive sensors to model expected motions in the scene caused by the motion of the robot's camera and body improves the anomaly localization performance.

- S. Thoduka, N. Hochgeschwender, J. Gall, and P. G. Plöger, "A Multimodal Handover Failure Detection Dataset and Baselines," in 2024 IEEE International Conference on Robotics and Automation (ICRA), 2024, pp. 17013–17019, doi: https://doi.org/10.1109/ICRA57147.2024.10610143.
We contribute a multimodal robot-human handover failure detection dataset and report results with baseline approaches in Chapter 5. We show that learning to temporally segment the robot's and human's actions as auxiliary tasks, and using multimodal data (video, force-torque, and gripper positions) both improve failure classification performance.

- P. Gohil, S. Thoduka, and P. G. Plöger, "Sensor Fusion and Multimodal Learning for Robotic Grasp Verification Using Neural Networks," in 2022 26th International Conference on Pattern Recognition (ICPR). IEEE, 2022, pp. 5111–5117, doi: https://doi.org/10.1109/ICPR56361.2022.9955646.
We contribute additional annotations to the VT dataset [8], and show that learning from vision, tactile sensors and gripper position data enhances the detection of grasp failures compared to using only individual modalities. This work is also presented in Chapter 5.

- S. Thoduka, S. Houben, J. Gall, and P. G. Plöger, "Enhancing Video-Based Robot Failure Detection Using Task Knowledge" in 2025 European Conference on Mobile Robots (ECMR), Padova, Italy, 2025, pp. 1–6, doi: https://doi.org/10.1109/ECMR65884.2025.11162998.
We enhance the ARMBench dataset [3] with annotations, including the temporal boundaries of the robot's actions, and the locations of source and target containers and the robot's end-effector in the video. We show that using the available task knowledge for video frame selection and pre-processing improves the performance of a video classification model for failure classification.

1.5 Thesis Structure

The remainder of the thesis is structured as follows:

Chapter 2 briefly discusses concepts that are referred to in the remainder of the thesis, including optical flow, neural network models such as autoencoders and variational autoencoders, the inflated 3D convolutional neural network, vision transformers, and temporal convolutional networks used for temporal action segmentation. It also describes our work in the context of robot task execution, monitoring and details regarding dataset creation.

Chapter 3 provides an in-depth discussion of related work in the area of failure and anomaly detection in robotics, general video anomaly detection, and failure datasets in robotics.

Chapter 4 introduces the Bookshelf dataset that consists of nominal and failed executions for placing a book on a shelf, and a method that uses the expected camera and robot body motions, and a variational encoder that learns to predict future optical flow images to perform anomaly localization on the dataset.

Chapter 5 introduces the multimodal Handover Failure Detection (HFD) dataset, which includes successful and failed object handovers between robots and humans. Baseline approaches based on 3D CNNs and temporal convolutional networks are presented as well. The chapter also describes enhancements to the VT dataset [8], and a multimodal approach to grasp failure detection.

Chapter 6 presents a method that utilizes task knowledge to improve the performance of existing failure classification methods. Task knowledge includes the temporal boundaries of the robot's actions and locations of task-relevant objects in the scene. The chapter also presents enhancements to the ARMBench dataset, via annotations of task knowledge.

Chapter 7 discusses the main results of the thesis, and possible directions for future work.

CHAPTER 2
Background

In this chapter, we discuss concepts that are referred to in the remainder of the thesis. We introduce optical flow (OF), video classification using 3D convolutional networks and transformers, autoencoders for unsupervised learning, and temporal action segmentation using temporal convolutional networks. We also discuss our work in the context of robot task execution and monitoring, and provide some details regarding our dataset collection setup.

2.1 Optical Flow

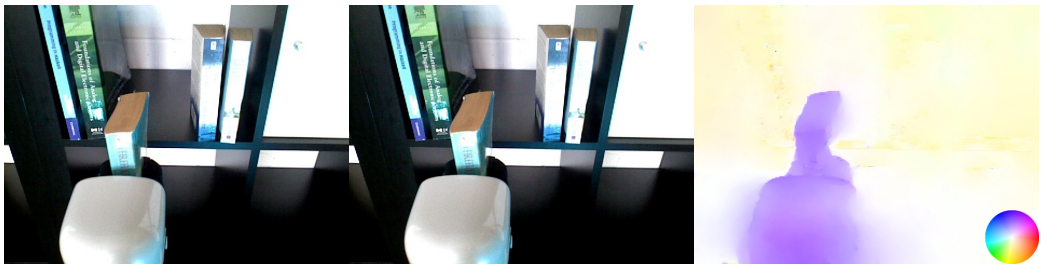


Figure 2.1: An illustration of the optical flow (right) computed between two consecutive video frames (left and middle), in which the robot arm and camera move upwards. The upward motion of the arm is seen in purple, whereas the motion of the camera results in an apparent downward motion of the background pixels, which is seen in yellow.

Optical flow in computer vision refers to the motion of pixels between consecutive frames captured by a camera. Motions are relevant for failure detection, since a variety of failures can be characterized by unwanted or unexpected motions, such as falling objects and sudden collisions. Motions are also relevant for video analytics in general, which is why some video models are trained with both RGB and optical flow sequences.

The optical flow is represented as a vector field, in which the OF vector of each pixel has a horizontal and vertical component quantifying the displacement of the pixel between frames. For learning tasks, OF can be represented as a two-channel image, with the intensities of the

Click or scan

Figure 2.2: An illustration of sparse optical flow computed between two consecutive video frames. The green arrows indicate the motion of trackable corner points in the image.

channels representing the horizontal and vertical components of the flow field respectively. The image is typically normalized to the range $[0, 1]$, with 0 and 1 representing the maximum motion in the negative and positive directions respectively. For visualization purposes, OF may also be represented using a colour wheel, in which the direction and magnitude of the vector are represented by the colour and intensity respectively. This is illustrated in Fig. 2.1, which shows the OF computed between consecutive frames from a video where the robot arm and camera are moving upwards. There are many algorithms for computing OF, including learning-based methods which use deep neural networks. OF computed using the Lucas-Kanade method [43] results in a sparse vector field, as illustrated in Fig. 2.2, since only pixels which can be reliably tracked (such as corners) are considered. Other methods such as Horn-Schunck [44], Farnebäck [45] and neural network-based methods [46], output a dense OF, which means that the motion of every pixel in the image is estimated.

In our work, we use dense OF images generated using the TV-L1 algorithm [47], shown in Fig. 2.1, and normalize the vectors in the range $[0, 1]$ after first clamping the maximum magnitude to 15 pixels. In Chapter 4, we use OF to detect motion-based anomalies, and in Chapter 5, we extract features from a pre-trained inflated 3D ConvNet (I3D) network [48] using sequences of OF images.

2.2 Autoencoders

An autoencoder is a type of neural network that consists of an *encoder* and a *decoder*. The encoder compresses the input into a lower-dimensional latent vector, and the decoder reconstructs the input from the latent vector, as illustrated in Fig. 2.3. By training an autoencoder

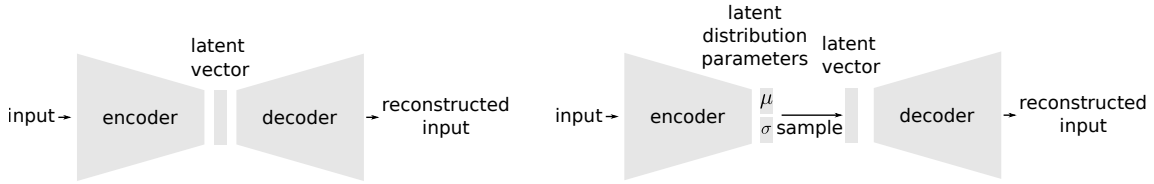


Figure 2.3: Autoencoder (left) and variational autoencoder (right)

to reconstruct the input, the network effectively learns to represent a high-dimensional input, such as an image, in the lower-dimensional latent space. Since they don't require any labels, autoencoders are often used for unsupervised representation learning; for anomaly detection they are trained to learn a representation of the nominal data. The main assumption for anomaly detection is that an autoencoder trained on nominal images will reconstruct anomalous images poorly compared to nominal images, since it has learned a good representation of nominal images, but not anomalous images. Thus, the reconstruction error can be used as an anomaly score.

A variational autoencoder (VAE) [49] is a probabilistic model that resembles an autoencoder, in which the encoder maps the input into a latent distribution rather than a single latent vector, and the decoder reconstructs the input from a latent vector that is sampled from the latent distribution. The encoder, $q_\phi(z|x)$, is a neural network parameterized by ϕ, which maps the input x to the latent distribution z, which is usually assumed to be Gaussian with a diagonal covariance matrix; i.e. for a data point $x^{(i)}$, $q_\phi(z|x^{(i)}) = \mathcal{N}(z; \mu^{(i)}, \sigma^{2(i)}I)$ [49]. The decoder, $p_\theta(x|z)$, is a neural network parameterized by θ, and maps a sampled latent vector z to a distribution over x, which is also modelled as a Gaussian in the case of image inputs. The sampling process is not differentiable, and therefore cannot be used as part of the gradient descent-based training process for neural networks. Thus, rather than sampling directly from the latent distribution, a noise vector is sampled from a standard normal distribution $\epsilon \sim \mathcal{N}(0, I)$, and transformed to the desired latent distribution as $z^{(i)} = \mu^{(i)} + \sigma^{(i)} \odot \epsilon$ [49]; this is also known as the "reparameterization trick", which makes the process differentiable and allows the VAE to be trained using gradient descent. The loss function includes a reconstruction loss (just as for an autoencoder), and an additional Kullback-Leibler (KL)-divergence term which encourages the latent distribution to be close to a standard normal distribution. We use a VAE in Chapter 4 to learn to predict a future OF image given a past OF image.

2.3 Inflated 3D Convolutional Neural Networks

The success of 2D convolutional neural networks for image recognition [36] was subsequently adapted for video data, initially using two-stream 2D convolutional networks [50], and later with 3D convolutional networks. A 3D convolutional kernel can operate on 3-dimensional inputs such as a *sequence* of image frames, which have a temporal dimension in addition to the spatial height and width dimensions in individual images. The I3D [48] model was the first 3D CNN to show superior performance compared to two-stream 2D CNNs on action recognition tasks.

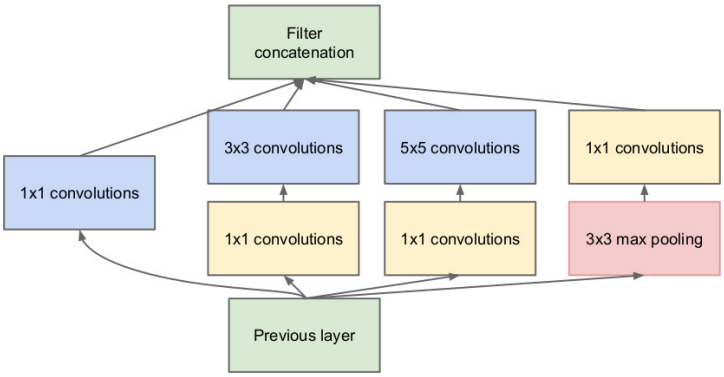


Figure 2.4: The 2D Inception module (figure reproduced from [4])

The network architecture for I3D is based on the 2D Inception architecture [4], whose main differentiating factor is the use of parallel convolutional filters of different sizes (i.e. 1×1, 3×3 and 5×5 convolutions, and a 3×3 max pooling layer). To reduce the computational cost of the larger filters, they are preceded by 1×1 convolutional filters. Finally, the outputs of all parallel filters are concatenated before being passed to the next layer, as shown in Fig. 2.4. For I3D, the authors initialize the network by "inflating" the filters of the 2D network with an extra dimension for time, and repeating the weights along that dimension [48]. The two-stream version of the I3D network consists of one stream for sequences of RGB images, and one stream for sequences of OF images, with late-fusion to combine the classification scores from the two streams, using a total of 64 input frames for each sequence. The I3D networks trained on the Kinetics dataset [48] have subsequently been used as feature extractors for video data, similar to how networks trained on ImageNet [51] have been used as feature extractors for images. In Chapter 5 and 6, we use I3D features for training the Multi-Stage Temporal Convolutional Network (MS-TCN) model, which we describe in more detail in Section 2.5, and we also fine-tune I3D models in Chapter 5 as one of the baseline methods for handover failure classification.

2.4 Vision Transformers

The transformer architecture, introduced by Vaswani et al. [52], is the architecture that has led to the development of large language models (LLMs), which form the basis of chatbot assistants such as ChatGPT[1], Gemini[2], Claude[3], etc. The same architecture has also been adapted for vision tasks, originally by the vision transformer [5] for image classification, and models such as the Multiscale Vision Transformer (MViT) [53] for video-based action recognition. The original transformer consists of an encoder and decoder, whereas vision transformers typically only consist of the encoder, as illustrated in Fig. 2.5 [5].

[1] https://chatgpt.com/
[2] https://gemini.google.com
[3] https://claude.ai/

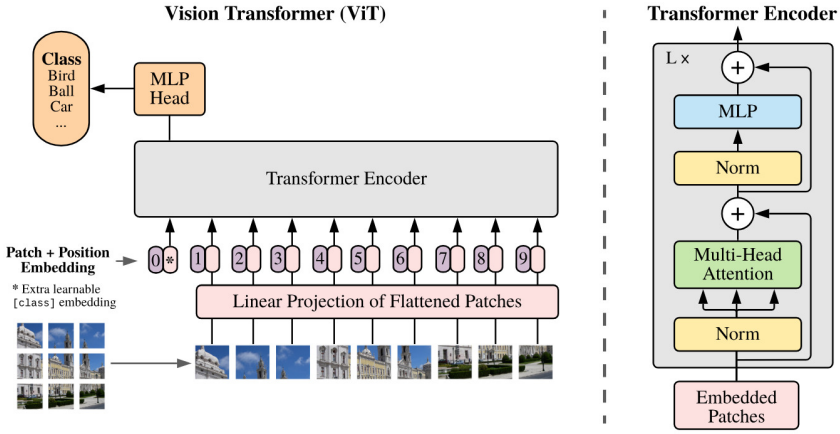


Figure 2.5: The vision transformer (left) takes patch and position embeddings from an image as inputs. A layer of the transformer (right) consists of the multi-head attention layer, and an multi-layer perceptron (MLP) layer along with normalization layers and residual connections (figure reproduced from [5]).

The transformer encoder consists of a stack of transformer blocks, each of which comprise of a self-attention layer and MLP, with normalization layers and residual connections between the two. The sequence of input embeddings for vision transformers is formed by splitting the image into patches, which are flattened and passed through an embedding layer. The patch embeddings are combined with learned or fixed position embeddings that encode the locations of the patches, since the transformer model itself is permutation invariant.

In the self-attention layer, linear layers are used to produce query (Q), key (K) and value (V) vectors for each of the input embeddings. The output embedding of the self-attention layer corresponding to one of the input embeddings is a weighted sum of the value vectors from all input embeddings, where the weights (or attention scores) are computed as the softmax of the dot product of the query for the current embedding with the keys from all input embeddings. This produces a sequence of output embeddings, in which each embedding is a combination of all input embeddings weighted by their relevance to that embedding. The attention layer is summarized by Eq. 2.1 [52], which uses the matrix forms for the queries, keys and values (so that the output embeddings can be computed for the whole sequence simultaneously), and includes the scaling term $\sqrt{d_k}$, where d_k is the dimension of the input embeddings.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

Typically, the attention layer consists of several attention "heads" (referred to as a multi-head attention), which correspond to multiple query, key and value vectors for each input, with the final output being computed as a concatenation over all attention heads.

For video transformers, the input to the model is a sequence of frames corresponding to a video clip. Each frame is converted into patches and concatenated, thus forming a long sequence of inputs for the transformer. Longer sequences result in higher computation and memory costs,

therefore video transformer models apply various strategies to reduce the sequence length. For example, ViViT [54] explores several model variants that factorize the spatial and temporal dimensions using separate encoders, separate self-attention layers and separate attention heads for the two dimensions. TimeSformer [55] similarly applies separate spatial and temporal attention. Video Swin Transformer [56] hierarchically applies self-attention only to local windows rather than the full sequence of inputs, and MViT [53] also adopts a hierarchical approach that, at each stage, reduces the sequence length and increases the channel dimension using a multi head pooling attention operator. In Chapter 6, we fine-tune the MViT model for failure classification.

2.5 Temporal Action Segmentation

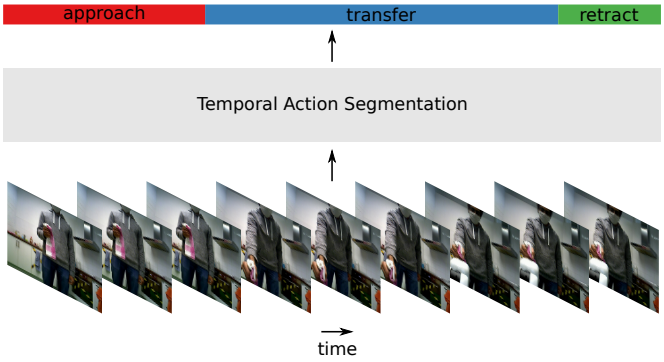


Figure 2.6: A temporal action segmentation model temporally segments a video into a sequence of actions by classifying each input frame into an action class.

Temporal action segmentation refers to the task of temporally segmenting an untrimmed video into a set of action classes [57]. The videos typically consist of some activity being performed, with the activity comprised of several actions. For example, for an object handover, the actions for both participants can include *approach*, *transfer* and *retract*, as illustrated in Fig. 2.6. Thus the output of temporal action segmentation is a label for each frame of the video corresponding to one of the action classes.

Since the input videos can be quite long, it is impractical to learn directly from the raw frames of the video. Instead, current methods extract spatio-temporal features from the video and use those as the inputs for the temporal action segmentation models, the majority of which are based on temporal convolutional networks or transformers. The MS-TCN [6] model is one such temporal convolutional network developed for the task of temporal action segmentation. The input to the model is a sequence of vectors representing the features of the frames of a video. The model consists of a series of non-casual, dilated 1D temporal convolutional layers, with the dilation factor being doubled after each layer, thus increasing the receptive field exponentially with each layer. Fig. 2.7 [6] illustrates the network architecture, and shows how the receptive field grows with each layer for a kernel size of 3. Since the convolutions are non-causal, the outputs depend on past, current and future inputs. The multi-stage version

of the network consists of multiple stages, in which each stage refines the predictions from the previous stage; the loss is computed as the sum of the losses from all stages. Apart from the standard cross-entropy loss for classification, the model also includes a smoothing loss which reduces over-segmentation. For the causal version of the model, illustrated in Fig. 2.8, the outputs only depend on the current and previous inputs, similar to the WaveNet model [7].

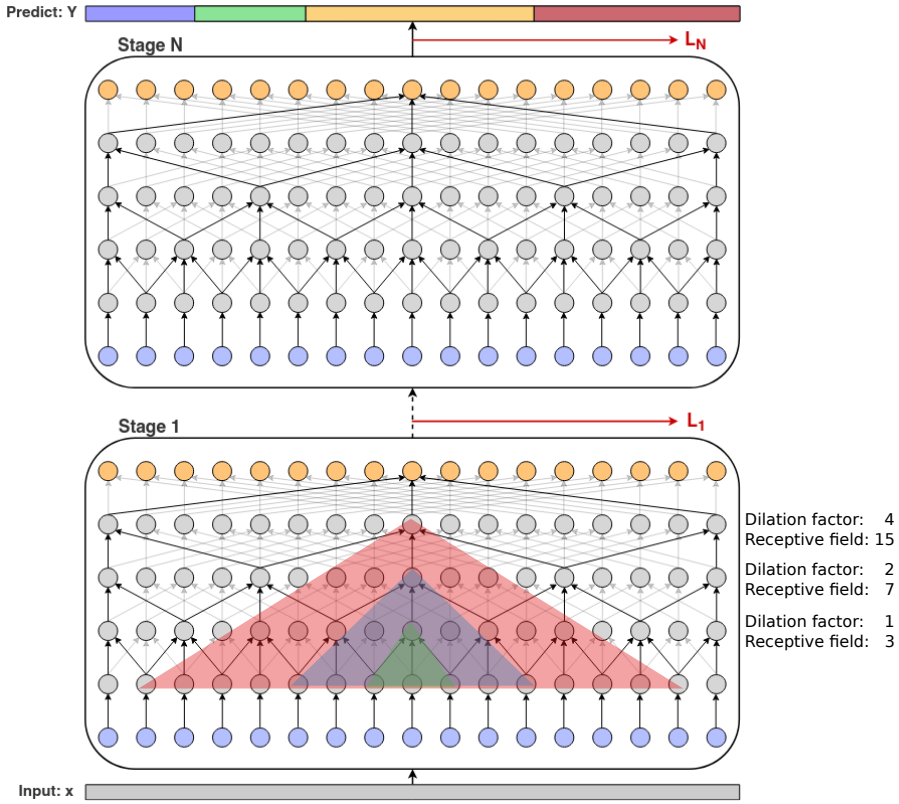


Figure 2.7: The MS-TCN model consists of multiple stages, with each stage comprising of dilated 1D temporal convolutional layers. The shaded regions show how increasing the dilation factor increases the receptive field. (figure reproduced from [6] with additional annotations added).

We use the MS-TCN model in Chapter 5 to learn to segment human and robot actions as an auxiliary task for learning to classify failures, and in Chapter 6 to extract the temporal boundaries of the robot's actions. As in [6], we extract features from I3D networks trained on the Kinetics dataset. Since the I3D network requires an input sequence of 64 frames, for each frame of the video, we consider the sequence of frames consisting of the current frame and 63 frames from the past. Each of the RGB and OF networks output a feature vector of dimension 1024; in Chapter 5, we concatenate the features such that each frame is represented by a 2048 dimensional vector, whereas in Chapter 6, we only use the features from the RGB network.

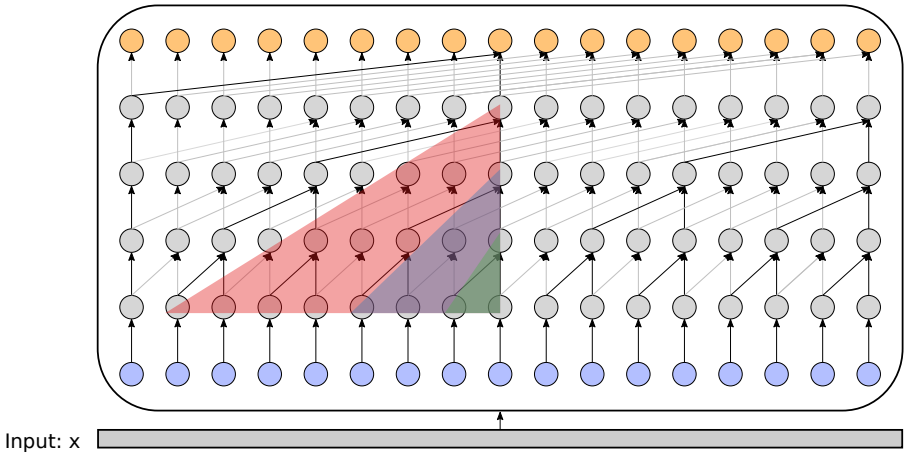


Figure 2.8: A single stage of a temporal convolutional network with causal convolutions. (figure adapted from [6] and based on the WaveNet model [7])

2.6 Task Execution, Monitoring and Dataset Acquisition

Robot perform tasks to achieve some overall goal, such as fetching objects for a person, tidying up a room, sorting objects in a warehouse, etc. These high-level tasks can be decomposed into lower-level tasks, such as navigating to the kitchen, handing over an object to a person and moving an object from one container to another. The lower-level tasks can again be decomposed into primitive actions of the robot, such as recognizing objects, grasping, approaching objects or people, path planning etc. An execution monitor runs in parallel with task execution in order detect and respond to failures or unexpected deviations from the objective by using observations from on-board sensors and other task-relevant information. Thus, our work, which focusses on failure and anomaly detection, forms part of the execution monitor, and is expected to observe the environment during task execution.

We propose failure detection methods based on machine learning; therefore it is necessary to collect training and validation data that would be observed by the robot during task execution. As described in Chapter 1, the training data could include both nominal and failure samples depending on the problem formulation; thus our data collection process records both types of samples. Since failures occur infrequently, we may intentionally induce failures during the data collection process, which allows us to control the number of failure samples and types of failures that are included in the dataset.

Robots typically have a heterogeneous set of sensors, such as RGB, depth and IR cameras, LIDARs, microphones, tactile sensors, force-torque sensors, encoders, etc. Different robots have different configurations of sensors, in terms of the available sensors and the spatial configuration of the sensors on the robot's body, and sensor data have various dimensions and are produced at different frequencies. Apart from sensor data, in some cases it is also useful to record data produced by the robot's perception and decision-making components, such as the location of a detected object, commanded trajectories of the manipulator, action execution parameters, etc. Particularly since sensors produce data at different frequencies, it is important that the raw

data is recorded at the native frequency of the sensor with timestamps for each sample. This allows for post-processing to align the data from different sensors and match their frequencies with up- or down-sampling if necessary.

In our work, we collect datasets using `rosbag`[4], and extract the relevant data during post-processing. Frames from the camera are extracted and saved either as individual images, or as a video. Sensor data, such as joint positions, are extracted into Numpy[5] arrays and saved in the `.npy` binary format. Typically, cameras have the lowest frequency; thus the remaining sensor data are down-sampled to the frequency of the images. Rather than down-sampling each sensor stream uniformly and independently, we instead sample them by find the closest data point from a given sensor stream to every frame in the camera stream by comparing their timestamps. This results in time-aligned data with a length equal to the number of video frames, which is convenient for multimodal learning.

Metadata, some annotations, and miscellaneous notes may also need to be recorded during the data collection process. Metadata can include the task configuration (such as objects used, or anonymized ID of the person interacting with the robot, etc.), which can later be used for performing data splits, or analysing the performance for different task configurations. We developed the `metrics_refbox`[6] tool for this purpose, which integrates recording ROS bag files and metadata, and coordinating the execution of tasks.

2.7 Metrics

For anomaly detection and localization, the output of the model is an anomaly score, which can be converted to a binary outcome by applying a threshold. A common approach for comparing such models is to compute detection metrics (such as true positive rate, precision, recall, etc.) after applying a range of thresholds, and computing the area under curves that trade-off the different metrics. For anomaly detection, we use the area under the receiver operating characteristic curve (AUC-ROC) and the AUC of the precision-recall (AUC-PR) curve. The ROC curve, illustrated in Fig. 2.9, plots the true positive rate versus the false positive rate for a range of thresholds, and the area under the curve represents how well a model is able to distinguish between nominal and anomalous samples. The true positive rate (TPR) (also termed recall) and false positive rate (FPR) are defined as in Eq. 2.2, where true positives (TP) are anomalous samples that are correctly detected, false negatives (FN) are anomalous samples which are not detected, false positives (FP) are nominal samples detected as anomalies, and true negatives (TN) are nominal samples which are correctly detected.

$$\begin{aligned} TPR = \text{recall} &= \frac{TP}{TP + FN} \\ FPR &= \frac{FP}{FP + TN} \end{aligned} \tag{2.2}$$

[4] a Robot Operating System (ROS) tool to record messages published by various processes
[5] `https://numpy.org/`
[6] `https://github.com/HEART-MET/metrics_refbox`

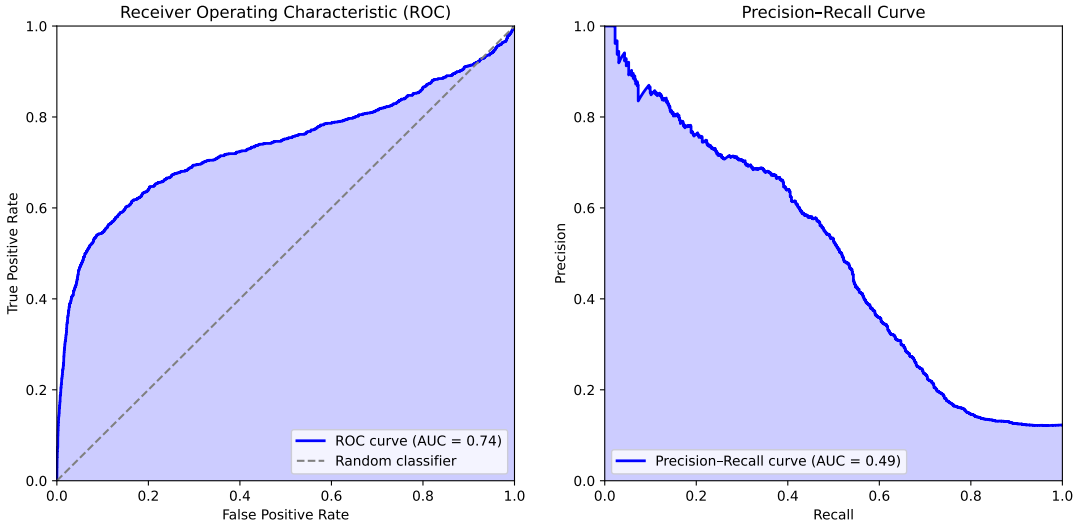


Figure 2.9: **Left:** The ROC curve plots true positive rate versus false positive rate for a range of thresholds. The area under the curve (indicated by the shaded region) represents how well the model is able to distinguish between nominal and anomalous samples. **Right:** The PR curve plots precision versus recall for a range of thresholds. The area under the curve (indicated by the shaded region) represents how well the model is able to detect anomalous samples.

The PR curve, also illustrated in Fig. 2.9, plots precision versus recall for a range of thresholds, and the area under the curve represents how well the model is able to detect anomalous samples. Precision is defined as in Eq. 2.3.

$$\text{precision} = \frac{TP}{TP + FP} \tag{2.3}$$

Accuracy measures the fraction of correctly predicted outcomes (see Eq. 2.4), and is used for failure classification.

$$\text{accuracy} = \frac{\text{correct predictions}}{\text{total predictions}} \tag{2.4}$$

For the binary case of failure detection, precision, recall and the F_1-score (see Eq. 2.5) are used, where TP, FP and FN are defined similar to the anomaly detection case, except that a positive detection refers to a failure detection rather than an anomaly detection.

$$F_1\text{-score} = 2 \times \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \times TP}{(2 \times TP) + FP + FN} \tag{2.5}$$

In Chapter 6, the test set of the failure classification dataset is imbalanced with a large proportion of nominal samples. Thus, instead of reporting the accuracy, we report the F_1-score (in addition to recall and FPR) by considering the classification of each failure type as a binary classification task.

CHAPTER 3

Related Work

3.1 Visual Anomaly Detection

Visual anomaly detection includes detection of anomalies in images and videos. While the literature on image anomaly detection primarily deals with detecting defects in industrial contexts, the literature on video anomaly detection is heavily focussed on surveillance videos and autonomous driving scenarios. Image anomaly detection is driven primarily by datasets such as the MVTec AD [58], VisA [59], BTAD [60] and CableInspect-AD [61]. The tasks include both image-level and pixel-level anomaly detection. The test sets of MVTec AD, ViSA and BTAD contain images of industrial objects with defects such as cracks, scratches, missing parts, dents with pixel-level masks for the anomalous regions. Training and evaluation is typically performed separately for each object in the dataset, using only nominal samples for training. CableInspectAD contains images of power line cables, with anomalous images being annotated with bounding boxes, pixel masks, defect categorization and severity level.

The main categories of methods for image anomaly detection are:

(a) reconstruction-based methods [62, 63], which train models such as autoencoders to reconstruct nominal images, and compute the anomaly score as the reconstruction error, which is expected to be higher for anomalous images,

(b) one-class classifiers [64, 65], that use features extracted from pre-trained networks in combination with methods such as One Class Support Vector Machines (OC-SVM) and Support Vector Data Description (SVDD),

(c) student-teacher networks [66, 67], in which a student network is trained to produce the same features as a pre-trained teacher network on nominal images, so that at inference time, the difference in features generated by the teacher and student networks is larger for anomalous images,

(d) memory bank-based methods [68–70], that store features from pre-trained networks in a memory bank and compute the anomaly score as the nearest neighbour distance to features in the memory bank, and

(e) normalizing flow-based methods [71, 72], that learn to model the distribution of nominal image features via maximum likelihood training of normalizing flow models, and use the negative log likelihood of image features as the anomaly score.

In all cases, a model of *nominality* is learned, such that deviations from the nominal model above a pre-defined threshold are considered anomalies.

In video anomaly detection, the research is driven by datasets such as ShanghaiTech [73], UCSD [74], CUHK Avenue [75], StreetScene [76] and UCF Crime [77], which consist of surveillance videos in outdoor scenes, and datasets such as RetroTrucks [78] and AnoVox [79], which consist of videos in driving scenarios. Surveillance videos normally focus on the same scene with fixed cameras, and involve scenarios such as bikers on a pedestrian walkway, dispersing crowds and crime-related activities such as arson and fighting. Anomalous videos recorded during driving scenarios involve dangerous traffic situations, accidents, unexpected obstacles on the road, etc. Apart from RGB video, some approaches for video anomaly detection also make use of dense optical flow, and thus consider both uni- and multi-modal approaches. The tasks can include anomaly detection (classifying a video clip as nominal or anomalous), temporal anomaly localization (identifying the anomalous frames within a video), and spatio-temporal anomaly localization (identifying anomalous frames, and the spatial location of the anomalies within the frame). Similar to image anomaly detection, the main approaches for video anomaly detection include: (a) reconstruction-based methods, which use 3D convolutional autoencoders [80, 81], VAEs and long short-term memorys (LSTMs) [82] to predict future RGB and optical flow images. (b) one-class classifiers [83–85], and (c) memory-bank based methods [81, 86].

In some cases, such as the UCF-Crime [77] dataset, both nominal and anomalous videos are present in the training set, but without frame-level labels. In such scenarios, several approaches [77, 87–89] use multiple-instance learning or multiple-sequence learning, which are weakly supervised learning approaches. The videos are split into segments, and it is assumed that at least one segment from each anomalous video is anomalous. The multiple-instance learning approach works by extracting features from segments from both types of videos and training a scoring network that enforces a higher score for the highest scoring instance of the anomalous video compared to the highest scoring instance of the nominal video.

The Oops! dataset [90] consists of videos with unintentional actions. The authors present various self-supervised video representation learning methods to train models in an unsupervised manner, and subsequently train a classifier on labeled videos to classify actions as intentional, unintentional or transitional. The self-supervision methods, motivated by differences in intentional and unintentional actions, include learning to predict video speed, a contrastive learning approach which predicts the feature representation of a clip given the feature representations of a previous and future clip, and a method to predict the correct order of clip representations.

Procedural videos are those which involve a person performing a sequence of actions according to some pre-specified procedure, such as assembling furniture, following a recipe for cooking etc. PREGO [91] consists of a video action recognition model, and a LLM which anticipates the next action based on the previous actions and context for egocentric videos of people performing procedural tasks. Errors in the procedural task are identified when there is a mismatch between the anticipated action and the detected action. Lee et al. [92] propose a

dataset and method for detecting errors in procedural tasks. The dataset, EgoPER, consists of egocentric video, audio, gaze, and hand pose captured during cooking tasks, with procedural errors such as omissions, additions, and modifications of steps. Their approach uses an action segmentation model, object detection, a graph convolutional network and k-means to learn features of prototypical action executions from nominal videos, and detects errors based on the dissimilarity of observed features to prototypical features of the identified action. The work by Wang et al. [93] also considers mistake detection in procedural tasks, but they use supervised training using video, hand-poses and eye gaze.

3.2 Anomaly and Failure Detection in Robotics

Failures in robotics can result in incomplete tasks and an undesired state of the environment (such as damaged or fallen objects). The use of multimodal data is common, although video data is not always used, especially in earlier works. Karg and Kirsch [94] present a framework for representing various kinds of expectations that can be compared to current observations to detect unusual situations during the execution of tasks. A given task may comprise a number of expectations, such as logical, temporal and probabilistic expectations, regarding the state of the environment, behaviour of humans, the robot's own state, etc. The expectations are represented in the form of a tree with child and parent nodes representing the dependencies between expectations. At run time, the validation scores from the expectations are aggregated to produce an overall *normality score*. The tree structure and the definitions of expectations encapsulates task and expert knowledge, and also allows for diagnosing the cause for an abnormal situation. In conceptually similar work, Gautam et al. [95] and Cao et al. [96], introduce an approach called *assumption-alignment tracking* for anomaly detection and classification during navigation and manipulation tasks. The developers of the robot skills define a set of assumptions and corresponding *checkers* that evaluate the veracity of the assumptions. The checkers verify properties of the sensors, actuators, environment and the output of the robot's skills; some examples include *camera sees expected colors*, *table is visible*, and *no collision is detected*. Feature vectors comprising the output of the checkers during nominal conditions are used to train a one-class classifier, and the Euclidean distance between a test feature vector and the decision boundary of the classifier is used as the anomaly score. The authors additionally show that with a few samples for each failure type, it is possible to classify the type of failure as well.

3.2.1 Hidden Markov Models

As described in Chapter 2, the nominal execution of a task typically follows a predictable sequence of low-level actions. Several works make use of hidden Markov models (HMMs) to model the temporal dynamics of task executions [27, 97–99]. The HMMs are trained with nominal sensor observations using the Baum-Welch algorithm, the outputs of which are the learned transition probabilities, i.e. the likelihood of transitioning between "hidden states", and the emission probabilities, i.e. the probability of sensor observations given the hidden state. At runtime, the model is used to compute the log-likelihood of the sequence of observations using the forward algorithm. If the log-likelihood is below a threshold, it is considered an anomaly.

Fox et al. [97] detect anomalies when the log-likelihood of the *most likely* sequence of states (computed using the Viterbi algorithm[1]) falls outside the range of log-likelihoods of the training data, or when the number of occurrences of a state is higher or lower than the maximum and minimum times the state occurs in the sequences of the training data. Luo et al. [27] use a Hierarchical Dirichlet Process HMM, which does not require specifying the number of hidden states beforehand, and anomalies are detected based on thresholding the gradient of the log-likelihood of the observation sequence. Park et. al [98] train an HMM using multimodal data from a PR2 robot during nominal executions of tasks such as closing doors, operating switches, pressing a toaster button and assisted feeding. A novelty of their approach is that the threshold on the log-likelihood varies based on the execution progress of the task, rather than a fixed threshold for the complete task. The execution progress is represented using a probability distribution over the hidden states of the HMM, and training data from nominal executions are used to form K clusters of the execution progress vectors, with associated mean and standard deviation of their log-likelihoods. At runtime, the execution progress is mapped to the closest cluster, and an anomaly is detected if the current log-likelihood is below $\hat{\mu}(L_{k^*}) - c\hat{\sigma}(L_{k^*})$, where $\hat{\mu}(L_{k^*})$ and $\hat{\sigma}(L_{k^*})$ are the mean and standard deviation of the log-likelihood of the closest cluster, and c is a constant. Their work is extended [100] to *classify* anomalies during the assisted feeding task using an MLP. The input to the MLP includes hand-crafted features such as frontal sound amplitude, x and y force, and distance from the robot to the person's mouth, and visual features extracted from an intermediate layer of a CNN using images from the wrist-mounted camera, and conditional log-likelihood of features from the HMMs, which measures the likelihood of each feature with respect to all other features. In addition, they use two separate HMMs to model nominal sensor data from different sensors, and flag an anomaly if the log-likelihood of either HMM falls below the task progress-based threshold. In [19], rather than using clustering, a Gaussian-process regressor that models the nominal log-likelihood for a given execution progress vector is found to improve anomaly detection performance overall.

Inceoglu et al. [99] experiment with using HMMs in a supervised setting by training HMMs with a nominal state and failure state for manipulation tasks such as picking, placing and pushing objects using a Baxter robot. Proprioceptive, audio, and visual sensor data are used to generate predicates such as *gripper open* and *gripper closing* for proprioception, *drop* and *ego-noise* for sound, and the difference in locations of the target object in the start and end state for vision. The predicates form the input features for the unimodal HMMs (with separate models for each modality) and multimodal HMM. A failure is detected when the likelihood for the failure state is higher than for the nominal state. For the *pick* action, the multimodal HMM and the proprioception HMM performed equally well, whereas for the *place* and *push* actions, a decision tree was found to perform similarly or better than the HMM.

In the work by Hegemann et al. [18], failures are detected using proprioceptive, force-torque and visual sensors during mobile manipulation tasks. In contrast to HMMs, they assume that the intermediate states of the robot are fully observable, and thus use Markov Chains to model the task. The sensor data are first grounded to symbolic action predicates such as `object_in_hand`, `object_placed`, `hand_opens`, etc. using predefined rules. A decision tree is

[1]The Viterbi algorithm uses dynamic programming to find the most likely sequence of states in an HMM given the observations.

used to map the grounded predicates to a robot state within an action performed by the robot. For example, states within the `pick` action include `reaching`, `grasping`, and `holding`. A task model, represented as a set of Markov Chains, is learned for each action of the robot (pick, carry, place) based on recorded nominal executions. At runtime, failures are detected by computing the likelihood of the state transitions based on the learned model for the currently executing action, and comparing it to a threshold. By using symbolic predicates and modeling the action executions using Markov Chains, the approach allows for interpretable failure detection and allows for better targeted recovery actions. The approach explicitly uses execution context, and task and expert knowledge through the use of predefined rules for mapping sensor data to predicates, and the decision tree to map predicates to pre-defined robot states.

3.2.2 Long Short-Term Memory Models

LSTM networks, which are a type of recurrent neural network that can better capture long-term dependencies, also model temporal dynamics and are used for anomaly and failure detection involving sequences of sensor observations [9, 101, 102]. Park et al. [102] use an LSTM-VAE model which learns to denoise noisy sensor signals from five sensors captured during nominal executions of a robot-assisted feeding task. The sensor signals include sound, end-effector force, joint torques and the positions of the spoon and mouth. Similar to their prior work [98], they incorporate the task progress in the learning system in two ways (a) by modelling the prior distribution for the latent space with a normal distribution whose mean is based on the progress of the task, and (b) by learning a mapping from the latent state (which varies based on the task progress) to an expected anomaly score using support vector regression, and setting the anomaly threshold based on the varying expected anomaly score.

Similar to [99], the authors in [101] extract manually designed features such as distance from the gripper to the object, sound class (no sound, ego noise, drop), gripper state, etc., and evaluate HMM, LSTM, and conditional random field-based approaches for failure classification for manipulation tasks such as *put-down*, *push*, *pick-up*, etc. They find that the LSTM model performs best overall.

Inceoglu et al. [9] use a convolutional LSTM network that learns to detect failures during manipulation tasks similar to [99]. The convolutional LSTM network operates on concatenated RGB and depth data, while audio features are extracted by passing Mel-frequency Cepstral Coefficients (MFCC) features through a 1D CNN. Intermediate fusion is performed between the visual and auditory features before classifying an execution as success or failure. The ablation study with different combinations of modalities showed that the version using RGB, depth and audio with the convolutional LSTM network performed the best on their dataset.

In [103], they follow an approach similar to [9] for failure classification, by using CNN/LSTM, MFCC and CNN-based feature extractors for visual, auditory and proprioceptive data respectively. Arapi et al. [104] apply a 1D convolution and LSTM-based network to anticipate grasping failures using acceleration and angular velocity signals measured by 16 inertial measurement units placed on soft hands.

3.2.3 Generative Models

As discussed in Sec. 2.2, autoencoder-based unsupervised learning approaches are used to learn lower-dimensional latent representations of nominal samples, and to reconstruct samples using the latent representation, such that samples with high reconstruction error are considered anomalous. Lee at al. [41] show that using a multimodal representation learned using variational autoencoders for learning manipulation policies results in robustness to external perturbations and sensor noise. The multimodal representation (based on RGB, depth, tactile and proprioceptive data) is learned in a self-supervised manner with variational autoencoders, which predict outputs such as the next end-effector pose, optical flow, and whether there will be a contact in the next step. In a follow-up work [105], they develop a crossmodal compensation model using similar techniques, with the addition of a reconstruction objective, which is able to detect corrupted sensors (based on reconstruction error), discard them and compensate for them with the remaining sensors.

Yoo et al. [106] construct a multimodal feature vector based on encoders for RGB, depth, sound and force-torque data and train an autoencoder to reconstruct the feature vector. Instead of using the reconstruction error as an anomaly score, the reconstructed input is again passed through the encoder, and the difference between the hidden layer outputs for the original input and the reconstructed input is used to compute the Normalized Aggregation along Pathway score [107], which is used as the anomaly score.

Normalizing Flows [108] are a type of generative model which can be used to estimate the probability density of data by transforming it into a known base distribution. If trained on nominal data only, it can be used for anomaly detection [20, 109] since the estimated likelihood of anomalous data would be lower compared to nominal data. The model consists of a series of invertible transformations that map the input sample to a latent space with a known, base distribution (such as a Gaussian), and is trained to minimize the negative log-likelihood of the training data. At test time, the data sample is transformed to the latent space, and the likelihood of the data sample is computed as a sum of the likelihood of its latent representation in the base distribution, and the log-determinant of the Jacobian of the transformation. Similar to reconstruction error, a threshold is applied on the likelihood to detect anomalies.

Brockmann et al. [20] use a multivariate normalizing flow model to detect anomalies using multivariate time-series data from a robot performing a pick-and-place task. The data includes mechanical and electrical variables such as target and measured motor position, velocity and torque, and various voltages and currents. The anomalies injected in the test set include collisions, invalid gripping position, variable weight of the target object, excess friction in the joints, etc.

Xu et al. [109] fit a continuous normalizing flow model to image features and robot states observed during nominal executions of their task policy. A dataset of nominal executions is also used to calibrate time-varying thresholds for the likelihood of observations output by the normalizing flow model.

3.2.4 Failure Anticipation

Failure anticipation considers the problem of predicting a failure before it occurs, such that actions can be taken to avoid the failure. The general approaches are similar to failure detection, but the models either classify the current state as likely to lead to a failure, or predict failure scores for future time steps.

Daftry et al. [110] use a two-stream CNN, corresponding to RGB and optical flow inputs, to anticipate collisions for an aerial vehicle navigating outdoors. Features extracted from the trained network are used to further train a linear SVM to classify whether the current state would lead to a collision.

Ji at al. [15] present a failure anticipation approach for an outdoor navigation task by making use of RGB images, point cloud from a 2D LiDAR, and the planned path. RGB image features are extracted from a pre-trained CNN, and features for the point cloud are learned using a VAE. An multi-head attention module is used to fuse features from the image and point cloud. The planned path is projected onto an image in the perspective of the RGB camera and passed through a CNN. A fully-connected layer then fuses the visual features (RGB and point cloud) and features from the planned path image and outputs a sequence of failure probabilities for future time steps. The training objective combines the binary cross entropy loss for failure predictions, and the reconstruction and KL-divergence loss for the variational autoencoder. At runtime, the failure probability score is computed as a weighted sum of failure probabilities within a prediction horizon. They find that the combination of camera and LiDAR data makes their approach more robust to sensor occlusions due to cluttered field environments.

Liu et al. [111] also perform failure anticipation by first predicting a future state in latent space, and then using a failure classifier to predict whether the future state would lead to a failure. Their approach is tightly coupled with their behaviour cloning policy for task execution by using a shared ResNet [112] encoder which embeds the camera image and the robot's proprioceptive sensor data into latent space embeddings. Future state prediction is performed by a conditional VAE, which takes as input the past latent space embeddings and the current action of the robot. The failure classifier, modelled using an LSTM, takes the current and past state embeddings and actions as input and outputs the probability of failure for tasks such as nut and gear assembly, threading and coffee pod packing.

Sogi et al. [113] use an initial image and planned actions to complete a long-horizon task to predict failures before any action is executed. They use the Recurrent State Space model [114], which uses input image features and actions for each timestep, to output latent variables for each future time step. An MLP-based decoder converts the latent variables into predicted image features, and a failure prediction network outputs a success score based on the predicted image features.

In [115], assumption checkers similar to [95] are implemented as nodes in a behaviour tree for a robot kitting task, and used to identify and explain situations when failures are likely to occur.

This *proactive* system of explaining failures before they occurred was found to be more understandable for humans, and led to a higher confidence in the selected failure cause. Ma et al. [116] explore the effect of faulty input images on failure anticipation. They use CNNs to anticipate failures in a tabletop pick-and-place task using as input noisy and blurry images,

masked images with the location of objects, and masked images with the planned final position of the end-effector. Finally, Farid et al. [117] present a neural network-based approach for anticipating failures with guaranteed bounds on the misclassification errors by making use of Probably Approximately Correct-Bayes theory.

3.2.5 Large Language Models and Vision Language Models

LLMs, vision-language models (VLMs) and vision-language-action models (VLAs) are increasingly being used in robotics for learning user preferences, reasoning, planning tasks and executing them [118–120]. In terms of failure detection, Liu et al. [25], demonstrate the use of an LLM that summarizes a robot's experience based on a sequence of scene graphs and audio events, provides an explanation for an execution or planning failure that has occurred and suggests actions to correct the failure. Failure detection is dependent on the quality of the constructed scene graphs, whereas the LLM is used for summarization and reasoning purposes. Mullen et al. [121] follow a similar approach for static scenes, by feeding a scene graph to an LLM and prompting it to classify the scene as normal, dangerous, unsanitary or dangerous for children. Cornelio et al. [122] construct an ontology describing the environment, predicates derived from audio and a scene graph, human preferences etc., which are tested against a set of logical rules to determine if a failure has occurred. An LLM is then provided with a set of recovery instructions for the detected failure type, the original plan and goal of the task and the current state of the environment to generate a new plan.

Works such as Duan et al. [119] and Agia et al. [123] use VLMs within a larger robot manipulation framework as success/failure classifiers, whereas Du et al. [124] investigate the use of VLMs as success detectors for robotics tasks in addition to other types of videos. Duan et al. [125] train their own VLM to detect failures and provide explanations for them by first developing a dataset of 49k image-text pairs of failure scenarios generated using the RLBench simulator. The input image to the VLM consists of a grid of images of the robot performing the task, with rows representing images from different viewpoints, and columns corresponding to the temporal dimension. A text prompt is generated for each subtask, in which the goal of the subtask is defined, and the model is prompted to answer whether it has succeeded with a 'Yes' or 'No' answer. In case of a failure, the VLM is additionally prompted to generate an explanation for the failure.

The work by Aduh et al. [126] combines embeddings from an LLM and a vision transformer to predict whether a target object will be damaged if it is picked up from a container during warehouse pick-and-place operations. The language model encodes object attributes such as product type, dimensions, weight, package type, and past history of damage, whereas the vision transformer encodes the camera image of the object in the container. This method falls in the category of failure anticipation, and the authors report a damage reduction rate of 64% by not picking objects that are predicted to be damaged. Zhou et al. [127] present a system for real-world robot evaluation, in which the success detector consists of the PaliGemma-3B VLM [128] fine-tuned on images from successful and failed examples of different scenes. Sliwowski and Lee [42] fine-tune a vision-language model to predict the phase of the action being performed given an image of the scene, and a natural language description of the action. At runtime, failures are detected by comparing the predicted phase to the expected phase

based on the current execution status. Their approach is applied to a robot that is preparing drinks, which performs the actions picking, pouring, placing and wiping, each of which have pre-, core- and post-action phases. The vision-language model is based on a frozen DINOv2 model [129] for extracting visual features and a frozen CLIP [130] text encoder for extracting textual features associated with the natural language description of the task. The trainable part of their ConditionNET network, consists of two transformer networks - the State Transformer, which encodes the state of the environment using visual features only, and the Condition Transformer, which predicts the phase of the action being performed using outputs from the State Transformer and the text features from the CLIP model.

The Gemini Robotics Embodied Reasoning 1.5 VLM and the Gemini Robotics 1.5 VLA both show failure detection capabilities [120]. The VLA is tasked with performing low-level actions to accomplish a sub-task, and when used in "thinking" mode is shown to have failure detection capabilities. These capabilities are demonstrated by the model attempting to repeat failed actions, and performing recovery actions. The VLM orchestrates the overall long-horizon task by providing instructions to the VLA, and can estimate the progress of the task and label the overall task as failed or successful.

The main motivation for using LLMs and VLMs for failure detection are their ability to generalize to a variety of tasks, without being trained on task-specific data. A drawback is that they require significantly more computational resources and longer execution times.

3.3 Datasets

Recording datasets with anomalies or failures in robotics is challenging since it is hard to consider *all* possible failures for a given task. Thus some types of failures might be over-represented, while rarely occurring failures are either under-represented or not present in the dataset at all. Frequently occurring failures might already be accounted for by the developer programming the robot, and thus it is especially important that anomaly and failure detection methods are able to detect the rarely-occurring failures. In the work discussed in the previous section, some authors release their training and evaluation datasets, some develop datasets but do not release them and some only collect training datasets and perform evaluations directly on the robot. In this section, we describe available anomaly or failure detection datasets that contain visual data.

The ARMBench [3] Defect Detection dataset consists of images and videos of a robot performing a pick-and-place task in a warehouse. The fixed robot manipulator picks objects from a source container, and places them in a destination container on a conveyor belt. During the pick-and-place operation, three types of failures are contained in the dataset: *1)* the robot picks up multiple objects with its suction gripper (`multi-pick`), or *2)* the object packaging is opened (`open`), or *3)* the object is separated into multiple parts (`deconstruction`). The image-based dataset consists of around 19k failure images from several viewpoints of the robot end-effector after it has picked up the object (during the so-called `transfer` phase), whereas the video-based dataset consists of around 4k failure videos which capture the full pick-and-place sequence. Since the `multi-pick` failure is not always visible from a single viewpoint, it is omitted from the video dataset, which only focusses on detecting `open` and `deconstruction` failures.

In addition to the failure samples, the image dataset includes 100k nominal samples, and the video dataset includes around 16k nominal samples. The video dataset is the largest video-based robot failure dataset available currently, consisting of around 20k samples. Compared to most other datasets, it is also the most representative of real-world conditions, since it was recorded in a warehouse during actual operation, and contains only naturally occurring failures. Samples from the video dataset are shown in Figs. 3.1, 3.2, and 3.3. The videos are annotated with the start time of the failure; in some cases, an `open` failure is followed by a `deconstruction` failure, in which case both start times are annotated, but the overall video is labelled as a `deconstruction` failure. The original annotations contain several errors, which we describe and correct, as described in Chapter 6.

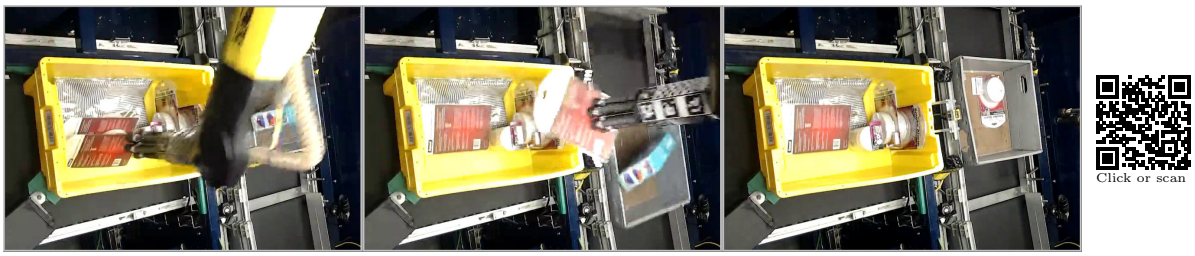


Figure 3.1: A nominal sample from the ARMBench Defect Detection video dataset in which a warehouse robot picks and places an object successfully

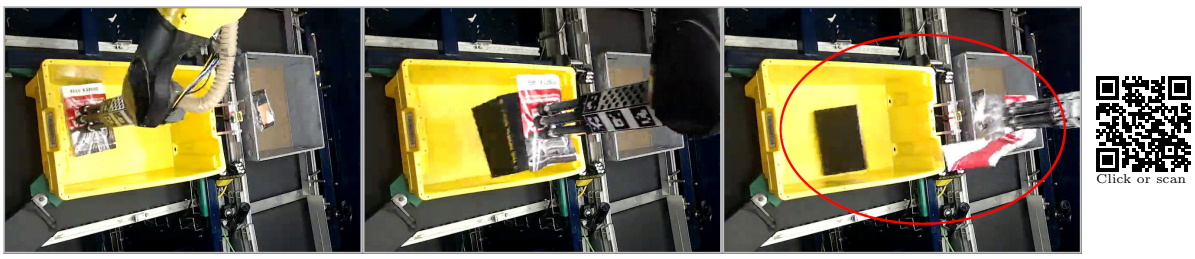


Figure 3.2: A `deconstruction` sample from the ARMBench Defect Detection video dataset in which the picked object comes apart into two pieces

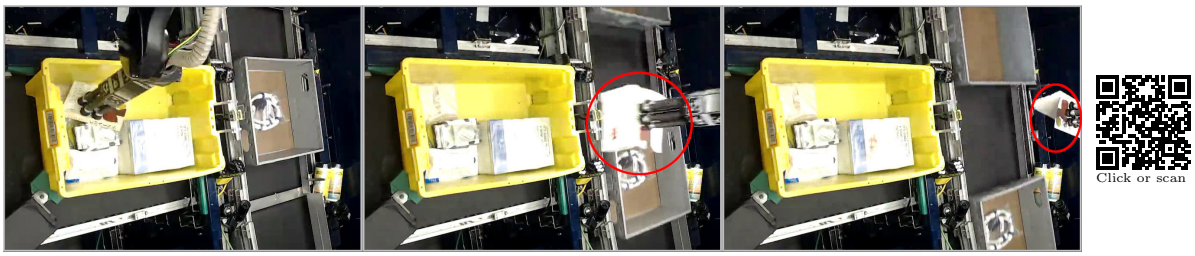


Figure 3.3: An `open` sample from the ARMBench Defect Detection video dataset in which the picked object is opened during the pick and place action

The VT [8] dataset contains executions of a fixed manipulator performing a pick-and-place task on a tabletop. The arm is fixed to a table, and the execution consists of approaching, grasping and lifting the object, and placing it back on the table. The dataset is comprised of RGB and depth videos from two directions (front and left), 16 tactile sensor data channels corresponding to the numbers shown in red in Fig. 3.4 and positions of the 8 finger joints. Three grasp strategies are used, namely, grasping from the back, right and top, and 10 objects are used, most of which are from the Yale-CMU-Berkeley object set [131]. In total, the dataset contains 1658 executions with the complete data, and an additional 892 executions without video and finger position data. Labels include whether or not the action was successful, and timestamps for the start of each sub-action (grasp, lift and release). In the failed executions, the robot either fails to pick up the object, or the object slips from the hand when the object is above the table. A sample failed execution is shown in Fig. 3.5.

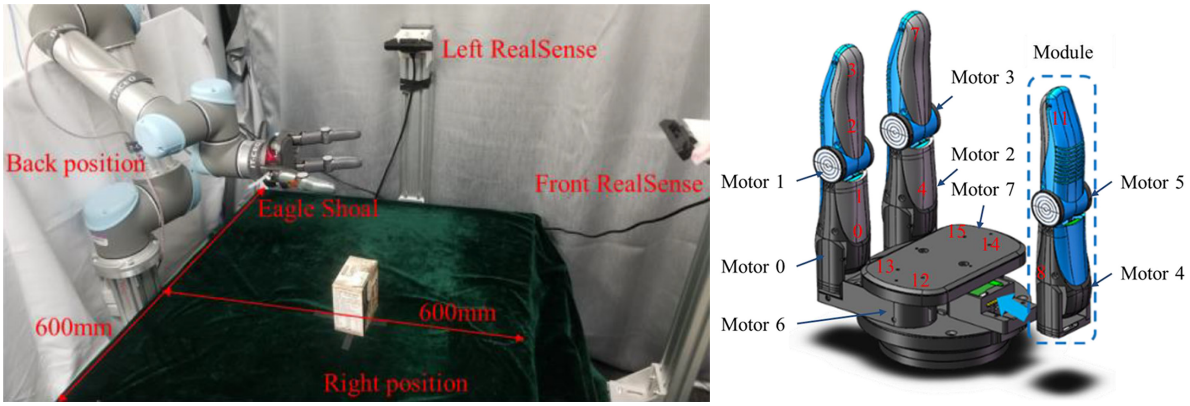


Figure 3.4: **Left:** Experiment setup for the VT dataset. **Right:** Shoal hand with 16 tactile sensors (labelled in red) and 8 motors for the finger joints (images reproduced from [8])

Figure 3.5: Sample failed execution from the VT dataset

The FAILURE dataset introduced in [9] consists of 229 successful and failed executions of five manipulation tasks, namely, *push*, *pick-and-place*, *pour*, *place-in-container*, and *put-on-top*. A Baxter robot is used to perform the tasks, and the dataset contains RGB and depth video from a head-mounted 3D camera and audio from a microphone on the torso. The experimental setup is shown in Fig. 3.6, and a sample failed execution is shown in Fig. 3.7.

The (Im)PerfectPour Dataset [42] was developed using a teleoperated robot that prepares drinks. The samples are split based on the task being performed, which are picking, pouring, placing and wiping. A total of 1096 samples from 544 demonstrations are part of the

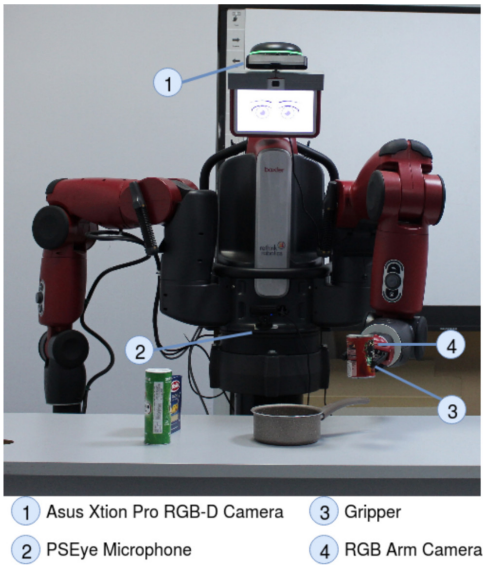


Figure 3.6: Experiment setup for the FAILURE dataset (image reproduced from [9])

Figure 3.7: Sample failed execution from the FAILURE dataset

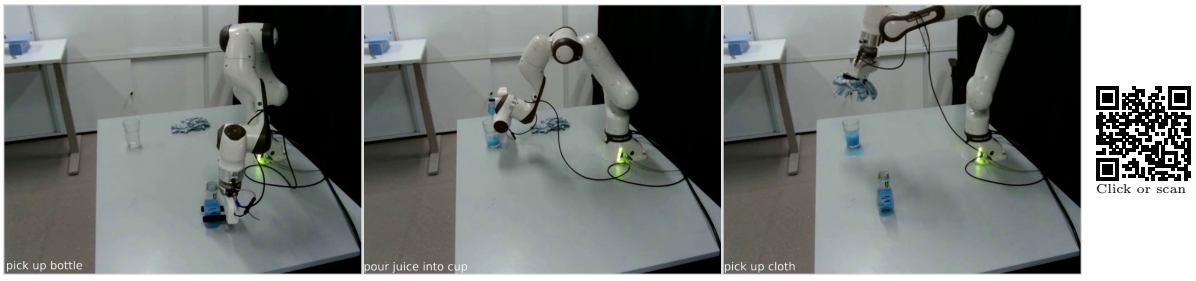


Figure 3.8: Sample tasks in the (Im)PerfectPour dataset

dataset, and failures include spills, and missing and fallen objects. Each sample contains a video of the task, proprioceptive data and force-torque data, and is annotated with the task being performed, failure types (if any), and temporal boundaries of the pre-, core and post-phases of the task. Since the tasks are performed in sequence, the post- and pre-phases of the tasks may overlap; for example, the post-phase of the `pick up bottle` task overlaps with the pre-phase of the `pour juice in cup` task. Sample task executions can be seen in Fig. 3.8. More recently, Sliwowski et al. [132] have also released the REASSEMBLE dataset consisting of 4551 demonstrations performed on the NIST Assembly Task Board [133], including 516 failed demonstrations. The samples consist of data from an event camera, RGB cameras, microphones, force-torque sensor and proprioceptive sensors.

The BC-Z Robot dataset [134] consists of more than 25k teleoperated and shared-autonomy episodes of a robot arm performing 100 different manipulation tasks on a tabletop. The tasks include placing objects in or on other objects, changing the orientation of objects, dragging objects, etc. The 100 tasks are based on the various different combinations of objects used for the base action being performed, such as *pick up the ceramic cup*, and *push the eraser across the table*, and *stand the bottle upright*. Although the dataset was developed for training imitation learning models, the dataset contains failed episodes which could be used for learning to detect failures. The data in each episode consists of a sequence of RGB images from the robot's head-mounted camera, state of the end-effector (such as pose, and state of the gripper), a sentence describing the task, and a sentence embedding for the task description. However, the RGB sequences are not contiguous, with missing frames corresponding to periods of time when the robot is not moving or is moving slowly. Some episodes are also incomplete, with the robot pausing before completing the task. Some successful episodes are incorrectly labelled, such as a different object being manipulated than specified in the task. Due to the non-contiguous video, incomplete episodes and incorrect labels, this dataset is not ideal for evaluating failure detection approaches.

The dataset introduced in PAAD [15] is recorded on a robot performing outdoor navigation, and contains images and range readings from the robot's camera and LIDAR, the coordinates of the planned 2D trajectory, and is labelled with success or failure for the next 10 time steps.

The authors of the REFLECT [25] approach, which uses LLMs to detect failures, developed the RoboFail dataset that consists of 130 samples of failure scenarios for a variety of tasks. A majority of the samples (100) are recorded using the AI2-THOR simulator [135] for 10 tasks, and the remaining 30 scenarios are recorded with a real-world robot for 11 tasks. Some sample tasks include *boil water*, *water plant*, *store egg*, and *serve coffee*, and failures include both planning and execution failures. Since the dataset only contains failure scenarios, it is well suited as a benchmark dataset for zero-shot detection approaches. Another simulator-generated dataset is the SafetyDetect dataset [121], which consists of 1000 static scenes in a home environment generated using the VirtualHome simulator [136]. Each scene may contain hazards such as spills, broken items, choking hazards, stove on, etc.

A summary of the most relevant failure datasets containing visual data is presented in Table 3.1, including the datasets that we contribute in this thesis.

Table 3.1: Summary of visual failure datasets

Dataset	Year	Count	Data Types
Visual-Tactile [8]	2019	1,658	RGB video, depth, tactile, proprioception
FAILURE [9]	2021	229	RGB video, depth, sound
Bookshelf †	2021	121	RGB video, proprioception, force-torque
BC-Z Robot [134]	2022	25,877	RGB frames, proprioception, task description embedding
ARMBench [3]	2023	20,008	RGB video
Handover Failure Detection †	2024	589	RGB video, proprioception, force-torque
(Im)PerfectPour [42]	2025	1,096	RGB video, proprioception, force-torque
REASSEMBLE [132]	2025	4,551	Event camera, sound, RGB video, proprioception, force-torque

†Ours

3.4 Discussion

The work on anomaly and failure detection in robotics often make use of multimodal data, with visual data becoming more prominent since the development of deep learning models. Some form of temporal modeling is performed, using HMMs, LSTMs, 3D CNNs, or action segmentation models, in order to account for the fact that the robot's actions normally follow a predictable temporal pattern for a given task. Some approaches convert sensor observations to predicates as an intermediate step before learning, and some make use of pre-trained feature extractors to work with features derived from raw sensor data. In recent years, several open datasets containing multimodal data for failure detection have been developed. We contribute to this development with two datasets (see Chapter 4 and 5) and improving annotations for the ARMBench and VT datasets. The use of multimodal data is common, but the research on fusing *video* with proprioceptive, force-torque and tactile sensor data is still in development. We focus on this aspect of multimodal fusion involving video in Chapter 5. Some works such as [18] explicitly model the task as a sequence of known actions, and others [116, 121] use the locations of detected objects as part of the learning process. However, a majority of approaches do not make use of such expert knowledge and other task-relevant knowledge (such as object locations, 3D model of the robot, etc.), and instead learn only using sensor data. For methods trained on specific tasks, we hypothesize that incorporating additional task knowledge should improve the learning process. We explore this in Chapter 4, where we use the robot's 3D model to represent expected motions of the scene caused by its own motion, and in Chapter 5 and 6, where we learn to temporally segment the robot's actions, and use the temporal boundaries of the robot's actions and locations of objects to pre-process frames used for video-based failure detection.

CHAPTER 4

Future Frame Prediction-Based Anomaly Detection

The contents of this chapter are partially based on the following publication:

S. Thoduka, J. Gall, and P. G. Plöger, "Using Visual Anomaly Detection for Task Execution Monitoring," in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2021, pp. 4604–4610, doi: https://doi.org/10.1109/IROS51168.2021.9636133. [1]

The contributions of the authors are as follows:

- **Santosh Thoduka** Proposed the main idea for the paper, collected the dataset, ran the experiments, and wrote most of the paper.
- **Jürgen Gall** Contributed to discussions, writing, and overall supervision.
- **Paul G. Plöger** Contributed to discussions, writing, and overall supervision.

4.1 Introduction

As discussed in Chapter 3, the literature on video anomaly detection focuses mainly on detecting anomalies in street scenes, captured from static surveillance cameras [75–77] or autonomous driving scenarios [78, 79]. In robotics, anomalies often occur when they are performing a task, in which case the robot, its camera, and manipulators could be in motion. Thus, we cannot rely on the assumption of a static camera and a fixed scene for detecting anomalies. The anomaly detection or localization methods should account for the motion of the camera, and motions of the robot's own body which might be visible in the camera. In this chapter, we consider a use-case in which a robot is placing a book on a shelf, during which, in addition to general motions in the scene, these two types of motions are present - namely, the head-mounted camera of the robot moves during the execution of the task, and the robot's arm is visible in the camera's viewpoint, often occluding the scene in front of it. In particular, we adopt approaches used in

the video anomaly detection literature, with additional components to account for the camera and robot body motion.

As discussed in Chapter 3, anomaly detection methods develop some model of *nominality*, such that deviations from the model are considered anomalous. In this chapter, we primarily focus on motion-related anomalies. In our proposed method, we model the expected motion during nominal task executions in our Bookshelf dataset. For general motions in the scene, we use a self-supervised approach, which learns to predict future optical flow from nominal executions. Anomalies are detected by comparing the predicted and observed optical flow during the execution. In order to account for camera and robot body motion, we use the robot's proprioceptive sensors and 3D meshes to model the expected motions in the scene caused by the camera and body motion, and compare it to the observed motion.

4.2 Dataset

Figure 4.1: Sample sequence of the robot successfully placing a book on a shelf in the Bookshelf dataset.

Figure 4.2: Sample sequence of the robot causing the other books on the shelf to be disturbed significantly.

We introduce the Bookshelf dataset, which consists of trials of a robot placing a book on a shelf.[1] The dataset is recorded in a robot lab using the Toyota Human Support Robot (HSR), which is a domestic service robot with a 5-degrees-of-freedom manipulator, two-finger gripper and a head-mounted Asus Xtion Pro camera. The robot is placed in front of a bookshelf

[1]The datasheet for the dataset can be found in Appendix A.

with a book already grasped in its gripper, and it executes a pre-defined trajectory of the arm to reach a target pose on the shelf, where it releases the book and retracts its arm. The dataset consists of 121 trials, of which 61 are nominal and 60 are anomalous. The anomalies are either induced manually or occur naturally without external intervention. The manually induced anomalies include occluding the camera and disturbing the robot through external collisions. Such external disturbances could happen in everyday environments where the robot works in close proximity with humans. The naturally occurring anomalies include unsuccessful placement of the book, due to falling on or off the shelf, and significant disturbances to books which are already on the shelf. Both scenarios could also happen in everyday environments due to imprecise perception, incorrect planning, or incorrect execution of the planned trajectory. A sample nominal sequence is shown in Fig. 4.1, and Fig. 4.2 illustrates a sequence containing anomalies, in which other books on the shelf are disturbed significantly by the robot's actions. As observed in the figures, the head (and hence the head-mounted camera) moves along the vertical axis as the robot arm is extended and retracted. The data recorded includes the (a) color and depth images from the head-mounted 3D camera, recorded at 30 Hz with a resolution of 640×480; (b) joint states, which include the positions, velocities and efforts of every joint of the robot, including the arm, and torso, sampled at 50 Hz; (c) and force-torque data from the wrist-mounted force-torque sensor sampled at 100 Hz. The 3D meshes and model of the robot are also available from the manufacturer, which, in combination with the recorded joint positions, allows us to reconstruct the motion of the complete robot body in 3D, and render it from the perspective of the head-mounted camera as shown in Fig. 4.3.

Click or scan

Figure 4.3: The real image (left) and rendered image (right) from the point of view of the robot's camera in the Bookshelf dataset.

For a single trial, each frame in the video sequence is labeled as nominal or anomalous. The frames are annotated by manually identifying the start and end frame of the anomaly, and marking all frames within that range as anomalous. We split the dataset into a training, validation and test set as specified in Table 4.1, in which a trial is considered nominal only if all frames are nominal.

Table 4.1: Training, validation and test sets for the Bookshelf dataset

	Nominal		Anomalous	
Split	**Trials**	**Frames**	**Trials**	**Frames**
Training	48	6,732	0	0
Validation	6	749	0	0
Test	7	7,969	60	1,108
Total	61	15,450	60	1,108

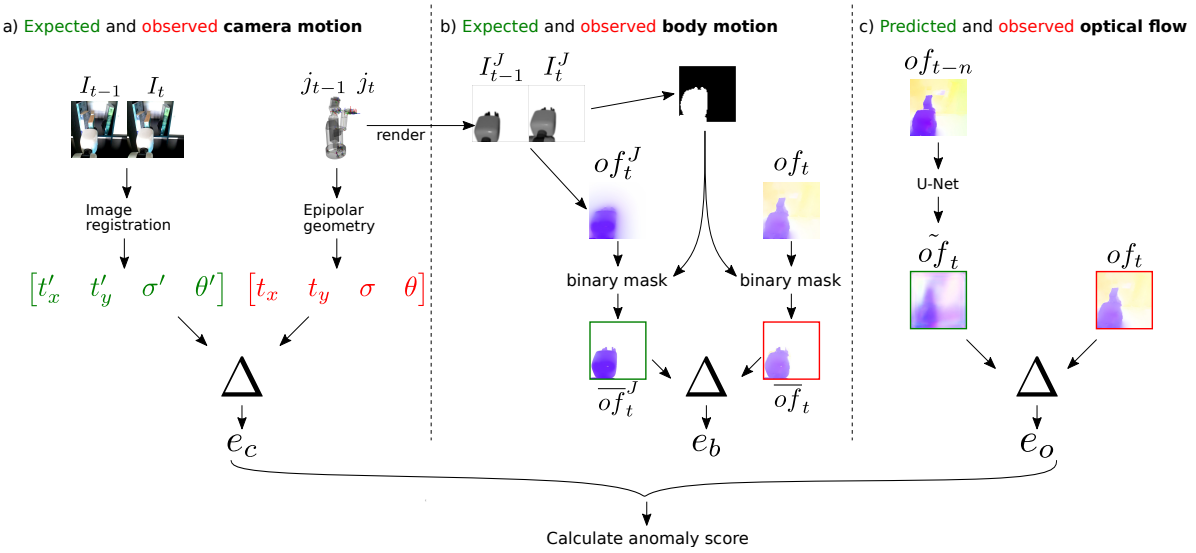


Figure 4.4: The expected motion is compared to the observed motion for three types of motions: a) **Camera motion:** the transformation between images obtained via image registration is compared with the expected transformation from known camera motion obtained from the robot's joint states b) **Body motion:** the observed optical flow of the robot's body is compared with the optical flow obtained by rendering the robot model after masking regions other than the robot's body c) **Optical flow:** the full optical flow image is compared to the optical flow image predicted by a trained U-Net network, given an optical flow image from the past. An anomaly score is calculated based on the residuals in each case (see text for notation).

4.3 Approach

For a given trial, we first downsample all data to 10 Hz, and make use of the sequence of T RGB video frames $I_{1:T} = (I_1, ..., I_T)$, rendered images of the robot's body from the perspective of the robot's camera $I^J_{1:T}$ (see Section 4.3.2), two-channel optical flow frames $of_{1:T}$, two-channel optical flow frames of the rendered images $of^J_{1:T}$, and joint states $j_{1:T}$. Optical flow is computed using the TV-L1 algorithm [47] for every consecutive pair of RGB frames, $of_t = \text{optical\_flow}(I_t, I_{t-1})$, and the first optical flow frame is duplicated, such that $of_1 = of_2$. Given these inputs, the task of anomaly localization is to classify each frame as `nominal` or `anomalous`, thus resulting in a sequence of outcomes $o_{1:T}$, where $o_t \in \{\texttt{nominal}, \texttt{anomalous}\}$.

Our approach consists of three components that focus only on the motions in the scene, and learns a nominal model of the motions in the scene using the nominal trials in the training set. At inference time, the task is to estimate an anomaly score for each frame by monitoring the deviation from the nominal model. By observing the *motion*, we focus naturally on the salient regions of the scene - particularly those that are relevant for motion-related anomalies. Motions of the robot's camera and body can be modelled using the internal sensors of the robot, and are less affected by the dynamics of the external environment in nominal scenarios. We therefore use an optical flow prediction network to model the overall motion, but also model the camera and robot body motion separately using the robot's kinematics and joint states. In all three cases, the error between the expected and observed motion is measured and used to calculate an anomaly score. Fig. 4.4 illustrates the three types of expected and observed motions which are compared. For camera motion, we compare the *expected* pixel motion calculated using the known camera motion against the *measured* pixel motion using image registration. For body motion, we compare the optical flow from *rendered* images with the optical flow from *real* images. For general motions, we use a probabilistic U-Net model [137] to learn to predict the current optical flow, of_t, given an optical flow frame from the past, of_{t-n}. The errors between the expected (or predicted) output and the measured output, $e_c, e_b,$ and e_o, are combined to calculate the anomaly score. Each of the three errors are described in detail in the following sections.

4.3.1 Camera Motion

The vertical motion of the camera during task execution causes an apparent motion of the entire scene, which can be modelled using epipolar geometry. The exact motion of the camera in 3D space with respect to the robot's base is known based on the positions of the torso and head, as recorded in $j_{1:T}$. Thus, the correspondence between image points in consecutive frames is obtained using Eq. 4.1 [138, Eq. (9.7)].

$$\mathrm{x}' = K'RK^{-1}\mathrm{x} + K'\mathrm{t}/Z \tag{4.1}$$

In Eq. 4.1, x' and x are the image points in the two consecutive camera views corresponding to a point X in the scene, $K = K'$ is the intrinsic camera matrix at the two time points, R and t are the rotation matrix and translation vector between the two camera positions and Z is the depth of the point X in 3D space. In our case, we simplify the equation by assuming no

rotation of the camera (since it only moves up and down), and by assuming a uniform depth in the scene, and thus use Eq. 4.2 [138, Eq. (9.6)]:

$$\mathrm{x}' = \mathrm{x} + K'\mathrm{t}/Z \tag{4.2}$$

where Z is the assumed fixed depth of the scene. Since we assume a fixed depth, neither the RGB nor depth images from the camera are used in this step. With a minimum of two such correspondences from the two views, we can compute the similarity transform between the two sets of points, shown in Eq. 4.3 [138, pg. 39], in order to retrieve the translation, t_x, t_y, scale σ, and rotation θ) between corresponding pixels in consecutive frames.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \sigma\cos\theta & -\sigma\sin\theta & t_x \\ \sigma\sin\theta & \sigma\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \tag{4.3}$$

In practice, more than two correspondences should be used to account for errors in the intrinsic parameters and noise in the depth data and proprioceptive sensors. The similarity transform here corresponds to the expected motion of the scene, since it is based solely on proprioception; i.e. it is the *expected* motion of the pixels based on the known motion of the camera. To compute the *observed* motion of the scene due to camera motion, image registration is performed on pairs of consecutive images. We use the Fourier-Mellin transform [139] to register image pairs; it estimates the similarity transform, $[t'_x, t'_y, \sigma', \theta']$, between the images based on the Fourier shift theorem. Since this method assumes that the dominant motion in the image is caused by camera motion, we mask out the region of the image where the robot body motion is visible before registration. Given the *expected* motion of pixels due to camera motion (represented by $[t_x, t_y, \sigma, \theta]$), and the *observed* motion of pixels due to camera motion (represented by $[t'_x, t'_y, \sigma', \theta']$), and assuming that the camera only translates during the execution of the task, we calculate the difference between the expected and observed motion as in Eq. 4.4.

$$e_c = |t_x - t'_x| + |t_y - t'_y| \tag{4.4}$$

4.3.2 Body Motion

The robot's manipulator and end-effector are typically in view of the camera during the execution of a manipulation task. Their motions must therefore be considered when modelling the nominal motion in the scene. We model the expected motion caused by the robot's own body by first rendering the robot's body from the point of view of the camera. This is achieved by using the current joint positions to configure a 3D model of the robot, and capturing an image in the 3D scene from the point of view of the robot's camera[2]. This results in an image as seen in Fig. 4.3. A sequence of such images $I^J_{1:T}$ is rendered using the joint states $j_{1:T}$, and the *expected motion* caused by the robot's body is represented by the corresponding optical flow images $of^J_{1:T}$. Optical flow images from the real and rendered images are both masked so that only motions of the robot body remain, resulting in $\overline{of}$ and $\overline{of}^J$. The mask is created by

[2]We use pyrender to render the images `https://github.com/mmatl/pyrender`

applying binary thresholding and contour detection on I^J, and can be seen in Fig. 4.4. The error between the two optical flow images is calculated as the absolute difference between the median magnitude (denoted by med)[3] in both directions x and y, as defined in Eq. 4.5:

$$e_b = |med(\overline{of}^J_x) - med(\overline{of}_x)| + |med(\overline{of}^J_y) - med(\overline{of}_y)| \tag{4.5}$$

4.3.3 Future Optical Flow Prediction

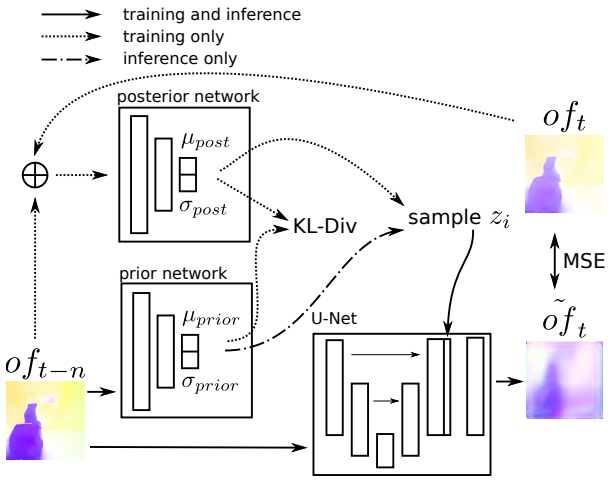


Figure 4.5: The probabilistic U-Net architecture consists of a posterior and prior network in addition to the standard U-Net. The network predicts a future optical flow image conditioned on a past optical flow image and a latent vector sampled from the distribution output by the posterior network (during training) or the prior network (during inference).

To produce an expected optical flow image, we use a neural network which learns to predict the current optical flow image, of_t, given a past optical flow image, of_{t-n}. We use a probabilistic U-Net [137] model which combines a conditional VAE with a U-Net [140], by the addition of a prior and posterior network, as illustrated in Fig. 4.5. This network architecture is chosen since it allows training with multiple ground truth outputs for a given input, and is relatively small compared to other frame prediction networks, with around 5 million trainable parameters. In our case, we consider the prediction of an optical flow frame to be time-agnostic [141]; the past optical flow frame which can best predict the current frame is selected from a range of past frames rather than from a fixed offset in the past. During training, an optical flow image from the past, of_{t-n}, is selected randomly within a certain range, where $n \in [A, ..., A+B]$, with the target optical flow image as of_t. At inference time, however, $\widetilde{of}_t$ is predicted B times using inputs of_{t-n} for all n, and the prediction with the lowest error is selected. The lower limit of the range, A, and the span of the range, B, are hyperparameters whose values we determine experimentally.

[3]We use the median instead of the mean to be robust to outliers

The prior network produces a distribution $P(z|of_{t-n})$ represented by the parameters of a multi-dimensional Gaussian with a diagonal covariance matrix $(\mu_{prior}, \sigma_{prior})$. At inference time, a latent vector, z_i, is sampled from this distribution, concatenated with the last activation map of the U-Net, and passed through some final convolutional layers to produce $P(of_t|UNet(of_{t-n}), z_i)$ (where the input to the U-Net is also of_{t-n}). Multiple $\widetilde{of}_t$ can be generated by sampling multiple times from the prior distribution.

During training, the latent vector is sampled from the distribution, $(\mu_{post}, \sigma_{post})$, parameterized by the posterior network, $Q(z|of_t, of_{t-n})$, instead of the prior network. The input to the posterior network is a concatenation along the channel dimension of the ground truth, of_t, and the input to the prior network, of_{t-n}. The posterior network learns to produce a latent vector that will result in the exact ground truth provided in its input. The training objective (Eq. 4.6) is that of a VAE [49], namely, the sum of mean-squared error (MSE) between the predicted output and the ground truth, and the KL divergence between the distributions output by the posterior and prior networks.[4] The KL-divergence term, which is weighted by β, a hyperparameter, encourages the prior distribution to move close to the posterior distribution.[5]

$$\begin{aligned}\mathcal{L}(of_t, of_{t-n}) = \mathbb{E}_{z\sim Q(\cdot|of_t, of_{t-n})}[-\log P(of_t|UNet(of_{t-n}), z)] \\ +\beta \cdot D_{KL}(Q(z|of_t, of_{t-n})||P(z|of_{t-n}))\end{aligned} \tag{4.6}$$

The input optical flow images are resized and center-cropped to 64×64 pixels. All models are trained for 50 epochs using a batch size of 128 and learning rate of 0.0001. At inference time, we sample M outputs from the network for each n (resulting in a total of $M \times (B+1)$ predictions), and the output, $\widetilde{of}_t$, with the minimum prediction error is used to compute the error as in Eq. 4.7.

$$e_o = min_{n\in[A,\ldots,A+B]} min_{j\in[1..M]}(mse(of_t, \widetilde{of}_t^{nj})) \tag{4.7}$$

4.3.4 Anomaly Score

The three error terms, e_c, e_b, and e_o represent deviations from nominal motions. In order to compute an overall anomaly score from the three error terms, we compare two strategies.

- Mean of errors: the mean of the three errors, after normalizing each error by twice their corresponding maximum errors in the training set.
- Binary threshold for camera and body motion errors: e_o is normalized by twice the maximum error in the training set, and a binary threshold is applied to the camera and body motion errors, setting the overall anomaly score to 1.0 if either is above the threshold, and e_o otherwise.

The motivation for the second strategy is that, in contrast to e_o, which is dependent on camera observations that are subject to domain shift, we do not expect a significant change in e_c and e_b

[4] The loss function is identical to that used in [137] with the exception that the output distribution $P(of_t)$ is considered to be a normal distribution instead of a categorical distribution

[5] We use $\beta = 10.0$, which is the default in the implementation by [137]

between the training and nominal test data since they are based on the robot's proprioceptive sensors. Therefore, we use a fixed threshold for the camera and body motion, which are their maximum errors in the training set, as in Eq. 4.8.

$$\begin{aligned} e_{c_{thres}} &= \max_{train}(e_c) \\ e_{b_{thres}} &= \max_{train}(e_b) \end{aligned} \tag{4.8}$$

The final anomaly score for the second strategy is calculated as:

$$\text{score} = \begin{cases} 1, & \text{if } e_c \geq e_{c_{thres}} \text{ or } e_b \geq e_{b_{thres}} \\ \frac{e_o}{2 \times \max_{train}(e_o)}, & \text{otherwise} \end{cases} \tag{4.9}$$

4.3.5 Evaluation Metrics

The area under the receiver operating characteristic curve (AUC-ROC) is the most common metric used in video anomaly detection. It allows us to compare methods based on their ability to discriminate between anomalous and non-anomalous frames at different thresholds for the anomaly score. The AUC of the precision-recall (AUC-PR) curve is an additional metric which summarizes how well the classifier is able to detect the anomalous frames at different thresholds. In both cases, an area of 1.0 represents a perfect detector. Unlike several video-anomaly detection methods (as reported in [142]), we do not normalize the anomaly score per trial, since this assumes that at least one anomaly occurs during a trial.

4.4 Experiments

4.4.1 Input Range

We determine ideal values for the upper and lower limit of the range $(A, A+B)$ of past frames to be used as the input to the probabilistic U-Net. As seen in Fig. 4.6, there is a marginal improvement of AUC-ROC as the lower limit and span are increased up to a certain point. Similar trends were observed for other combinations of A and B, and for AUC-PR. We obtain the best results for $A = 5$ and $B = 4$, namely by using an optical flow frame between 5 and 9 time steps in the past to predict the current frame.

4.4.2 Comparison to Other Anomaly Detection Methods

We compare the performance of the probabilistic U-Net to two other future prediction methods, U-Net + GAN [143] and VRNN [82], and an HMM-based method similar to [19]. We train the probabilistic U-Net with $n = 1$ and $M = 1$, such that it only predicts the next optical flow frame, and with $A = 5, B = 4$ and $M = 1$. For a fair comparison with the future prediction models, we do not consider the body and camera motion, so that the anomaly score is e_o. Both future prediction models are trained to predict the next RGB frame given a sequence of input RGB frames. We fit the HMM using the Baum-Welch algorithm with multivariate features comprising of the maximum magnitude from optical flow images and magnitudes of

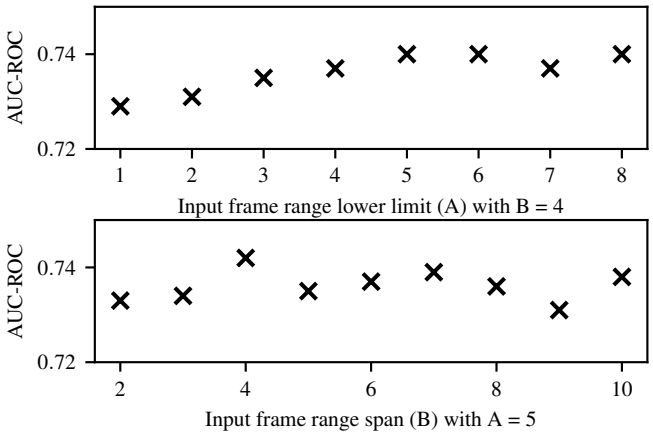


Figure 4.6: AUC-ROC for different ranges of input optical flow images

observed body motion and image motion due to camera motion. The anomaly score for each frame in the test set is computed using the log likelihood of the sequence of observations up to that frame. We also report a baseline which just copies the previous frame as the predicted next optical flow frame. The results in Table 4.2 show that both the "copy" baseline and the probabilistic U-Net have an advantage over the two future prediction methods and the HMM, particularly considering the AUC-PR metric. Varying the range (i.e. $A = 5, B = 4$) of past optical flow frames which are used to predict the current frame shows a minor improvement in the AUC-ROC.

Table 4.2: Comparison to other methods

Method	**AUC-ROC**	**AUC-PR**
Copy baseline	0.703	0.412
U-Net + GAN [143]	0.675	0.339
VRNN [82]	0.533	0.166
HMM	0.657	0.197
Prob. U-Net ($A = 1, B = 0, M = 1$)	0.725	0.463
Prob. U-Net ($A = 5, B = 4, M = 1$)	**0.737**	**0.464**

4.4.3 Error Combination

Table 4.3 compares the performance when using the two different strategies for combining the errors from the three motion types. Using the mean of the scores results in a drop in performance compared to using just the future optical flow prediction error. The performance improves when a binary threshold is applied to the camera motion error, but there is no change when the threshold is applied to the body motion error. We find, firstly, that very few samples exceed the threshold for the body motion error, and secondly that the OF errors are already

quite high for the samples where the body motion error is greater than the threshold. In contrast, a large number of samples with high camera motion error have relatively low OF error. An example of this can be seen in the bottom example in Fig. 4.7, where, after the initial occlusion, the OF prediction model is able to predict reasonably well that the next frame will be similar to the previous one (i.e. still occluded), whereas the camera motion error, which is based on proprioception, remains high.

Table 4.3: Results for computing the anomaly score using two strategies

Strategy	**Motion error type**			**AUC-ROC**	**AUC-PR**
	OF	**Body**	**Camera**		
Mean	✓			0.734	0.464
	✓	✓		0.710	0.391
	✓		✓	0.737	0.470
	✓	✓	✓	0.733	0.468
Binary threshold for body and camera motion	✓			0.734	0.464
	✓	✓		0.734	0.463
	✓		✓	**0.789**	**0.551**
	✓	✓	✓	**0.789**	0.550

4.4.4 Input Optical Flow Variants

In this section, we compare four types of optical flows used as input to the probabilistic U-Net model. The inputs to the probabilistic U-Net model are:

1. **OF**: the OF computed on consecutive frames without any modifications.
2. **Registered OF**: the OF is calculated after registering consecutive images. In effect, the camera motion is removed from the optical flow images in this variant.
3. **Masked OF**: regions where the robot's body are visible are masked out in the OF (using the inversion of the mask described in 4.3.2). This removes observed motions of the robot body from the OF.
4. **Masked registered OF**: a masked version of the registered OF.

Table 4.4 shows the AUC-ROC and AUC-PR for all optical flow variants, with the final anomaly score computed as in Eq. 4.9. We also include the results without using the probabilistic U-Net model, and only using the body and camera motion errors. All models are trained five times and the mean results are reported, using $A = 5$, $B = 4$ and $M = 10$. When only the body and camera motion errors are considered to compute the anomaly score (1-3), the best AUC-PR is achieved, though with a much lower AUC-ROC compared to using the optical flow prediction network. This suggests that a combination of the two error types is useful for detecting anomalous instances, but can result in a high false positive rate.

Table 4.4: Results from using different input optical flow variants

ID	Prob. U-Net input type	Motion error type			AUC-ROC	AUC-PR
		OF	Body	Camera		
1	None		✓		0.505	0.524
2				✓	0.541	0.581
3			✓	✓	0.545	**0.582**
4	OF	✓			0.734	0.464
5		✓	✓		0.734	0.463
6		✓		✓	0.789	0.551
7		✓	✓	✓	0.789	0.550
8	Registered OF	✓			0.720	0.444
9		✓	✓		0.720	0.442
10		✓		✓	0.775	0.532
11		✓	✓	✓	0.775	0.531
12	Masked OF	✓			0.737	0.485
13		✓	✓		0.737	0.484
14		✓		✓	**0.797**	0.570
15		✓	✓	✓	**0.797**	0.569
16	Masked registered OF	✓			0.720	0.446
17		✓	✓		0.720	0.444
18		✓		✓	0.779	0.537
19		✓	✓	✓	0.779	0.537

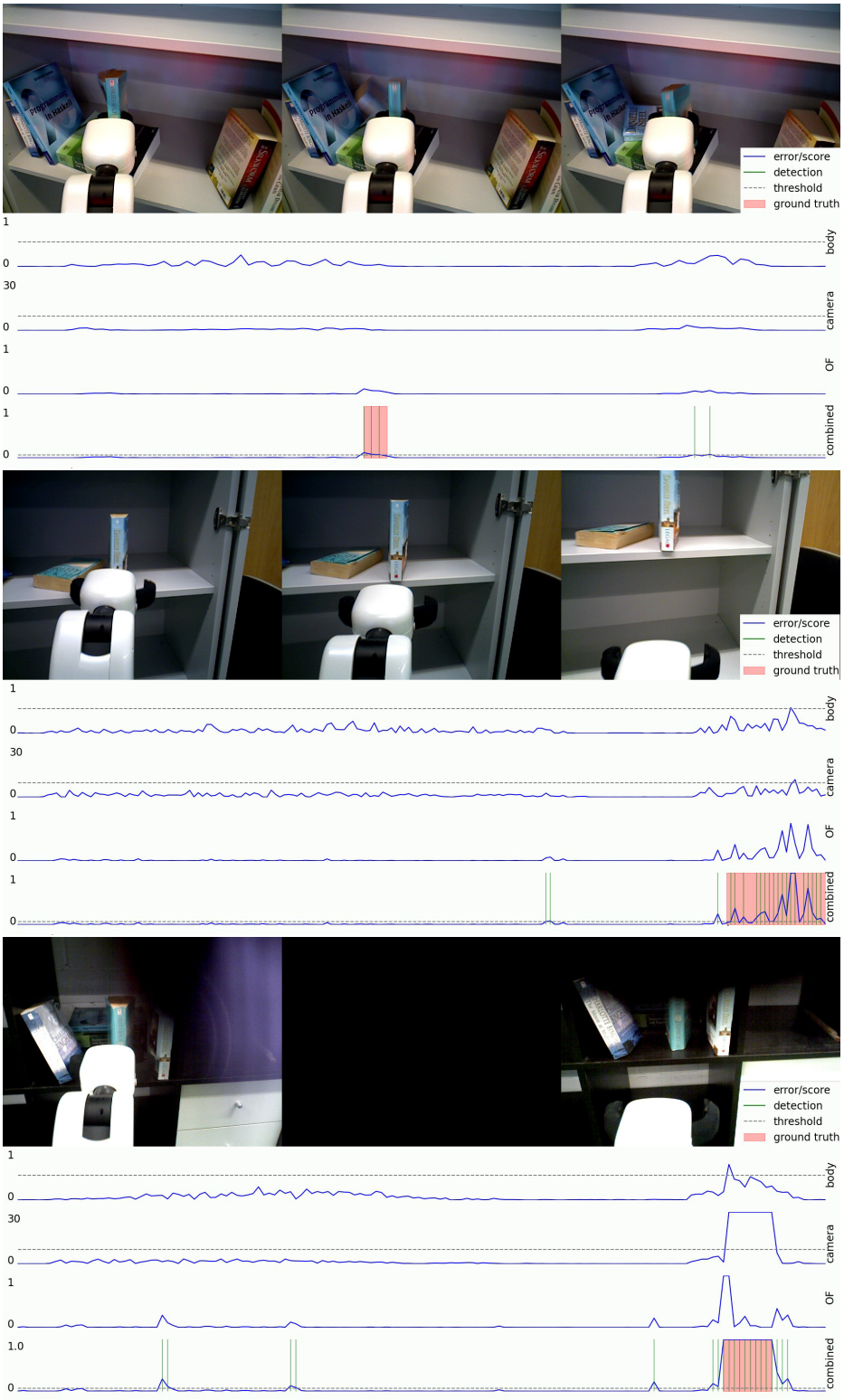


Figure 4.7: The figures show sample errors and the combined anomaly scores. The video frames correspond to the ground truth anomaly regions. **Top**: a book on the shelf falls while the robot approaches the shelf. **Middle**: an external disturbance is applied to the robot. **Bottom**: the robot's camera is occluded.

Combining all three error types (4-7), results in an improved AUC-ROC, suggesting improved separation of nominal and anomalous samples, with a slight drop in AUC-PR. Similar to the results in Table 4.3, incorporating the body motion error has little or no effect on the performance (6-7, 10-11, 14-15, 18-19). When registered OF images are used (8-11), we see a drop in performance, possibly due to additional noise added by the registration process to the optical flow images. However, when body motions are masked in the OF (12-15), both AUC-ROC and AUC-PR improve, and results in the best overall performance. For the masked, registered OF (16-19), the performance is between the registered (8-11) and masked versions (12-15). The results indicate that it is better to incorporate the prior information about the robot's body motion by masking the optical flow images, and that it is better to use the prior information about the camera motion for computing the overall anomaly score. Some sample detections using the model from (15) are shown in Fig. 4.7; the threshold with the best F_1 score (i.e. the furtherest point to the top-right in the PR curve) is selected for the combined score. We note that a large majority of the false positives occur when there are minor lighting changes, and during fast motions of the robot, both of which lead to large changes in the optical flow between consecutive frames.

4.4.5 Handover Failure Detection Dataset

In Table 4.2, we noted that the performance of the probabilistic U-Net model is only slightly better than the "copy" baseline. Observations of the predicted optical flow images also suggest that the model generally expects similar motions in the future as the past observed motion. Therefore, additional motions which were not present in the input optical flow would be seen as anomalies. Due to the nature of the anomalies in the Bookshelf dataset, which are all associated with *additional* motions in the scene, it is not known whether *missing* expected motions would also be detected by the network. The Bookshelf dataset also does not contain arbitrary motions in the background, since the robot faces a bookshelf against a wall.

Therefore, we evaluate the approach on a subset of the HFD dataset, which consists of some failures which are caused by a *lack* of motion, and also contains other motions in the background which are not relevant to the task. The dataset, which is described in more detail in Section 5.2.1, contains failures during robot-human handovers, in which the human may drop the object, not approach the robot, not grasp the object, or not release the object. During a nominal execution of the task, the expected motions include both robot body and camera motion, in addition to motions by the human approaching and retreating from the robot. We consider a subset of the dataset, using only trials with the Toyota HSR, such that the training and validation sets only contain nominal trials. This subset consists of 73 trials (6757 frames) for training, 18 trials (1250 frames) for validation, and 97 trials (7682 frames, of which 2140 are anomalous) for testing. For the failures `drop` and `no release`, the anomalous frames correspond to the annotated human actions `dropped` and `not released` respectively. For the failures `no approach` and `no grasp`, we label the region 2.5 seconds after the end of the robot's `approach` action until the end of the robot's `retract` action as anomalous.[6]

[6]The 2.5 second offset is based on the mean time after the end of the robot's `approach` action that the human starts interacting with the robot during nominal handovers.

The results in Table 4.5 show that only considering the body or camera motion (1-3) results in random guessing (AUC-ROC ≈ 0.5), but the addition of the score from the probabilistic U-Net model results in scores far below 0.5, which means that a majority of the frames which are nominal are considered to be anomalous and vice-versa. This highlights the previously discussed limitation of the motion detection network, namely, that it mainly anticipates motions to remain similar to previous motions, rather than predicting motions based on the task being performed. The AUC-ROC scores for OF when considering failure types individually are $0.260, 0.133, 0.417, 0.486$ for `no approach`, `no grasp`, `no release` and `drop` respectively, which also indicates that failures that have less motions than expected (`no approach`, `no grasp`) are harder to discriminate than failures that have more motions than expected (`drop`, `no release`). Applying the camera motion error, masking the optical flow image based on the robot body motion, and registering images before computing the optical flow all result in minor improvements in performance similar to the Bookshelf dataset, except that using registered OF in this case leads to a minor improvement.

Table 4.5: Results for the Handover Failure Detection dataset

ID	**Prob. U-Net input type**	**Motion error type**			**AUC-ROC**	**AUC-PR**
		OF	**Body**	**Camera**		
1	None		✓		**0.500**	0.155
2				✓	0.478	0.199
3			✓	✓	0.478	0.199
4	OF	✓			0.264	0.216
5		✓	✓		0.264	0.215
6		✓		✓	0.264	0.219
7		✓	✓	✓	0.264	0.219
8	Registered OF	✓			0.278	0.223
9		✓	✓		0.278	0.223
10		✓		✓	0.272	0.222
11		✓	✓	✓	0.272	0.222
12	Masked OF	✓			0.290	0.225
13		✓	✓		0.290	0.224
14		✓		✓	0.290	0.228
15		✓	✓	✓	0.290	0.228
16	Masked registered OF	✓			0.310	**0.241**
17		✓	✓		0.310	**0.241**
18		✓		✓	0.302	0.234
19		✓	✓	✓	0.302	0.234

4.5 Summary

In this chapter, we first presented the Bookshelf dataset, consisting of nominal and anomalous trials of a robot placing a book on a shelf, with anomalies such as falling objects and camera occlusions. We presented an approach that models nominal motions in the scene, by considering general motions, and motions induced by the robot's own actions, namely motions of the robot's body in the image, and the apparent motion of the scene due to camera motion. The difference between expected and observed motions in all three cases are combined to compute an overall anomaly score for every frame, which indicates how far the current observation has deviated from the expected (nominal) motion. General motions are modeled using a future optical flow prediction network, expected body motions are modeled by computing the optical flow of rendered images of the robot body, and expected scene motions due to camera motion are derived using Epipolar geometry. The results show that incorporating the known information about the robot's motions leads to an improved anomaly detection performance. In particular, using the known robot body motion to mask the input optical flow images for the optical flow prediction network, and applying a binary threshold on the camera motion error and incorporating it into the final anomaly score leads to improved performance. For the flow prediction network, we also find that predicting optical flow by using a range of past optical flow images results in an improved performance compared to using only the previous frame. A limitation of the approach is that the future optical flow prediction network predicts motions that are generally similar to the past motion, and thus cannot detect anomalies which are characterized by a *lack* of motions, as seen in the results for the HFD dataset.

CHAPTER 5

Multimodal Data for Failure Detection

The contents of this chapter are partially based on the following publications:

- S. Thoduka, N. Hochgeschwender, J. Gall, and P. G. Plöger, "A Multimodal Handover Failure Detection Dataset and Baselines," in 2024 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2024, pp. 17013–17019, doi: https://doi.org/10.1109/ICRA57147.2024.10610143. [2]

  The contributions of the authors are as follows:

  - **Santosh Thoduka** Collected the dataset, developed the main solution, ran the experiments, and wrote most of the paper.
  - **Nico Hochgeschwender** Contributed to overall supervision, discussions, writing and integration to the METRICS project.
  - **Jürgen Gall** Contributed to the overall supervision, discussions, and writing.
  - **Paul G. Plöger** Contributed to the overall supervision, discussions, and writing.

- P. Gohil, S. Thoduka, and P. G. Plöger, "Sensor Fusion and Multimodal Learning for Robotic Grasp Verification Using Neural Networks," in 2022 26th International Conference on Pattern Recognition (ICPR). IEEE, 2022, pp. 5111–5117, doi: https://doi.org/10.1109/ICPR56361.2022.9955646. [144]

  In this chapter, the contents from the paper are the additional annotations, description of data processing and representation (Section 5.3.1), and the clip-wise classifier model (Section 5.3.2.2) which we use to compute the clip-wise failure localization results (Section 5.4.2).

  The contributions of the authors are as follows:

  - **Pritesh Gohil** Developed the main solution, wrote the code, ran the experiments, and wrote most of the paper.
  - **Santosh Thoduka** Formulated the problem statement and general approach, contributed to the writing and discussions, and added annotations to the dataset.
  - **Paul G. Plöger** Contributed to discussions and overall supervision of the project.

5.1 Introduction

In this chapter, we explore the use of multimodal data for detecting failures associated with a human-robot interaction task and table-top object manipulation (see examples in Figs. 5.2 and 5.9). Prior work, including [9] and [19], has shown the benefit of using multimodal data for anomaly and failure detection. Sensors such as proprioceptive, force-torque and tactile sensors provide additional information regarding physical interactions with the environment and the robot's own motions, which are complementary to the information from visual sensors. In Chapter 4, we used proprioceptive data in combination with the robot's 3D model to model expected motions. In this chapter, we use proprioceptive data, force-torque data and tactile sensors as additional inputs to a neural network and explore methods to fuse them with video data.

In contrast to the anomaly localization method described in Chapter 4, here we consider supervised training for failure detection, classification and localization since the datasets contain both successful and failure samples in the training sets. Firstly, we consider *failure classification* during object handovers between a robot and human, for which we use visual, proprioceptive and force-torque data. We collect a dataset of robot-to-human (R2H) and human-to-robot (H2R) handovers, with human-induced failures such as dropping the object, ignoring the robot etc. We perform failure classification using an I3D network using late and intermediate fusion of the three modalities, and an MS-TCN model that jointly learns to classify failures and to segment the actions of the robot and human using concatenated features from the three modalities. Secondly, we perform *failure detection* and *temporal failure localization* on the VT dataset [8] using a combination of tactile, joint position and visual data during the execution of a table-top pick-and-place task. Failure detection is performed using the I3D network, and for failure localization we use a 3D ResNet model to perform clip-wise detection (thus resulting in a coarse-grained failure localization output), and an MS-TCN model for fine-grained failure localization. Our main focus in this chapter is on how to make use of multimodal data in the learning process for failure classification and localization, including how to represent data for learning and how to fuse multimodal data effectively.

5.2 Handover Failure Detection

An object handover between a human and a robot is a common action during tasks involving human-robot collaboration and physical human-robot interaction. It requires both the giver and the receiver to communicate and coordinate their actions to ensure a safe and successful handover. Failures in object handovers are likely, even for human-human handovers, for example, due to miscommunication, incorrect interpretation of intentions, object properties, or improper behaviour of either participant. The survey on object handovers by Ortenzi et al. [145] identifies works which consider error handling during handovers; most approaches that consider failures in handovers typically focus on the physical handover phase, which is concerned with a failure to coordinate between the receiver and the giver. Overall, the survey reports that only a minority of approaches incorporate error handling in their work, with the primary focus being to detect disturbances and preventing falling objects. For example, both

Eguiluz et al. [16] and Davari et al. [146] try to differentiate between an object being pulled (as part of the nominal robot-to-human (R2H) handover task) versus other disturbances, so that the robot does not release the object unexpectedly. Similarly, Parastegari et al. [147, 148] detect slipping objects and regrasp them to prevent failures caused by objects falling. Rosenberger et al. [149] prevent failures by aborting the handover if certain error conditions are met, such as the human moving too close to the grasp position, or if a collision is detected. Liu et al. [150] describe approaches for increasing safety by adapting the behaviour of the robot depending on the uncertainty of its perception systems. For example, to prevent collisions in darker environments, the robot maintains a larger distance to the human since the uncertainty of the skeleton detection system increases. Han and Yanco [151] investigate the expectations that a person has when unrecoverable handover failures occur, and find that people expect the robot to provide explanations, and not just non-verbal cues, for why a failure has occurred. In other work [152, 153], specific failures, such as occlusions or no detections of the human's hands, are considered to either disregard an experimental trial, or to abort the handover. Iori et al. [154] consider two scenarios which deviate from a nominal handover, in which *(i)* the person pauses while approaching the robot before continuing, and *(ii)* the person's arm is pushed by another person during the approach. The paper presents a reactive trajectory generation method which is able to adapt to the perturbations. Grigore et al. [155] aim to increase safety in R2H handovers by considering human engagement signals (e.g. gaze) in addition to physical signals (e.g. current and torque) to decide if and when to release the object, and Mohandes et al. [156] use vision and joint torques to detect the person's intentions and actions.

Prior research related to handover failures are concerned with preventing failures, or detecting conditions which might lead to a failure, and typically only deal with failures in the pre-handover and the physical handover phases. The post-handover phase should also be considered as it might take some time for the receiver to stabilize the object in their hand and get accustomed to its physical properties such as temperature or weight. In our work, we are concerned with detecting failures after they have occurred, with a specific emphasis on human-induced failures. We also consider failures in the post-handover phase for both H2R and R2H handovers.

5.2.1 Dataset

We introduce the Handover Failure Detection (HFD) dataset in order to address the requirement for failure detection benchmarks for object handovers in robotics. The dataset consists of R2H and H2R handovers, with both successful and failed trials. As described earlier, the existing work on error handling during object handovers primarily deals with preventing failures, such as detecting slipping objects [148], detecting unintended events such as pulling, pushing and bumping the object using a tactile sensor [146], and adapting a handover controller to different lighting conditions and human behaviours [150]. However, the main focus of our dataset is to evaluate the detection of unpreventable failures caused by the human participant *after* they have occurred.

5.2.1.1 Dataset Development

The failures in the dataset are induced by the participants based on prior instructions. We select the failures by following the procedure we defined in [157], a process similar to failure mode and effects analysis (FMEA) that requires defining the objective of the task and identifying failure modes which may be associated with negations of the objective. The objective of a handover task is to "safely transfer an object from the giver to the receiver". This is further decomposed into the objectives of the three phases of a handover, which are defined as follows: *(i)* approach: both participants approach each other, with the giver holding the object, *(ii)* transfer: the receiver and giver coordinate to grasp and release the object respectively in a safe manner such that the object is transferred to the receiver *(iii)* retract: both participants move away from each other, with the receiver holding the object. The objectives for both participants during each phase of both types of handovers are defined in Table 5.1. For the human participant, we specify actions which do not satisfy the objective (i.e. the failure modes), such as not approaching, not releasing the object, and dropping the object. During the dataset collection, the participants are instructed to induce the failure modes described in the table. In principle, the "object is dropped" failure mode can be caused by either participant or both participants; for the purposes of our dataset, this failure is always caused by the human participant.

Table 5.1: Objectives of the giver and receiver during R2H and H2R handovers and the associated failure modes

	Phase	Objective	Failure modes
R2H	Approach	the giver approaches the receiver with the object the receiver approaches the giver	the human does not approach the robot
	Transfer	the receiver grasps the object the giver releases the object once it is safe to do so	the human does not grasp the object the object is dropped
	Retract	the giver releases the object and moves away from the receiver the receiver continues to grasp the object	the object is dropped
H2R	Approach	the giver approaches the receiver with the object the receiver approaches the giver	the human does not approach the robot the object is dropped
	Transfer	the receiver grasps the object the giver releases the object once it is safe to do so	the object is dropped
	Retract	the giver releases the object and moves away from the receiver the receiver continues to grasp the object	the human does not release the object

5.2.1.2 Dataset Details

We recorded the dataset in a robot lab set up to resemble a home environment, with a living room, dining room and kitchen. Two robots were used, namely, the Toyota HSR, which is a domestic service robot with a 5-degrees-of-freedom manipulator and a two-finger gripper, and a Kinova Gen3-based dual-arm robot. The HSR has a head-mounted Asus Xtion Pro camera and a wrist-mounted force-torque sensor. For the dual-arm robot, both arms have a RealSense D410 camera module, therefore, we use one arm for performing the task, and use the camera from the second arm for recording the video. The arms do not have a force-torque sensor, but

the driver provides an approximation of the force and torque that would be sensed at the tool tip, which is derived from the torque sensors of the individual joints.

The data for each trial includes RGB video from the camera, joint states of the robot arm (position, velocity and effort for each joint, including the gripper) and force-torque (F-T) sensor data. The video is recorded at 30 Hz with resolution of 640×480. Since the joint states and F-T data are recorded at a higher frequency than the video, we also include resampled joint states and F-T data which are aligned to the timestamps of the video frames. Resampling is performed by selecting the latest available joint state and F-T reading for each video frame.

Overall, the dataset consists of 589 trials performed with the two robots, 17 participants and 22 object classes. The class distribution and distribution between handover type and robot platform is summarized in Table 5.2. The training, validation and test sets are split based on the participants, resulting in 337 trials from 11 participants for training, 101 trials from 3 participants for validation, and 151 trials from 3 participants for testing. Additional details can be found in the datasheet of the dataset in Appendix A.

Table 5.2: Handover Failure Detection dataset statistics

	R2H					H2R				
	success	no approach	no grasp	drop	**total**	success	no approach	no release	drop	**total**
Toyota HSR	68	50	49	58	225	51	46	57	66	220
Kinova Gen3	18	17	17	20	72	19	18	17	18	72
Total	**86**	**67**	**68**	**76**	**297**	**70**	**64**	**74**	**84**	**292**

5.2.1.3 Annotations

The outcome of the trials, robot used, anonymized participant ID, and task (R2H or H2R) are annotated at the time of recording. In addition to the outcome, we also annotate the start and end of the actions of the person and the robot, such that each frame of the video is associated with a robot action and a human action. The robot's actions follow the same temporal order in all trials, and are $A_R = \{\texttt{Idle}, \texttt{Approach}, \texttt{Interact}, \texttt{Retract}, \texttt{Post-Idle}\}$. The start and end of the actions are annotated based on the robot's joint states, as visualized in Fig. 5.1. The `Approach` action starts when the robot's arm starts extending and ends when the arm stops moving. Similarly, the `Retract` action starts when the arm starts returning and ends when it stops moving. The remaining actions are bounded by the start and end of the trial and start and end of the `Approach` and `Retract` actions. In general, the actions of the robot are assumed to be known and their temporal boundaries can also be retrieved from an action executor, such as a state machine.

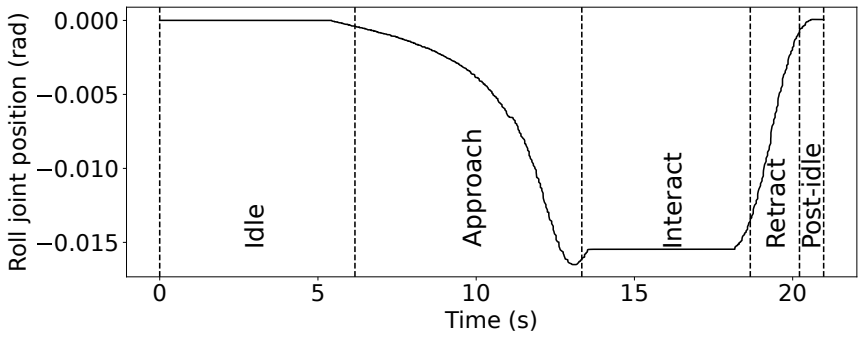


Figure 5.1: The temporal boundaries of the robot's actions are annotated based on when the arm joints start and stop.

The person's actions are manually annotated and they include the handover phases and non-nominal actions: $A_H = \{\texttt{Idle}, \texttt{Approach}, \texttt{Interact}, \texttt{Retract}, \texttt{Post-Idle}, \texttt{Not Released}, \texttt{Dropped}\}$. The `Interact` action was annotated by observing the F-T data and video to determine the start and end of the physical interaction of the person with the object and robot. The start of `Approach` and the end of `Retract` were annotated by observing the motion of the person's arm in the video. A visualization and video of the data and annotations can be seen in Fig. 5.2.

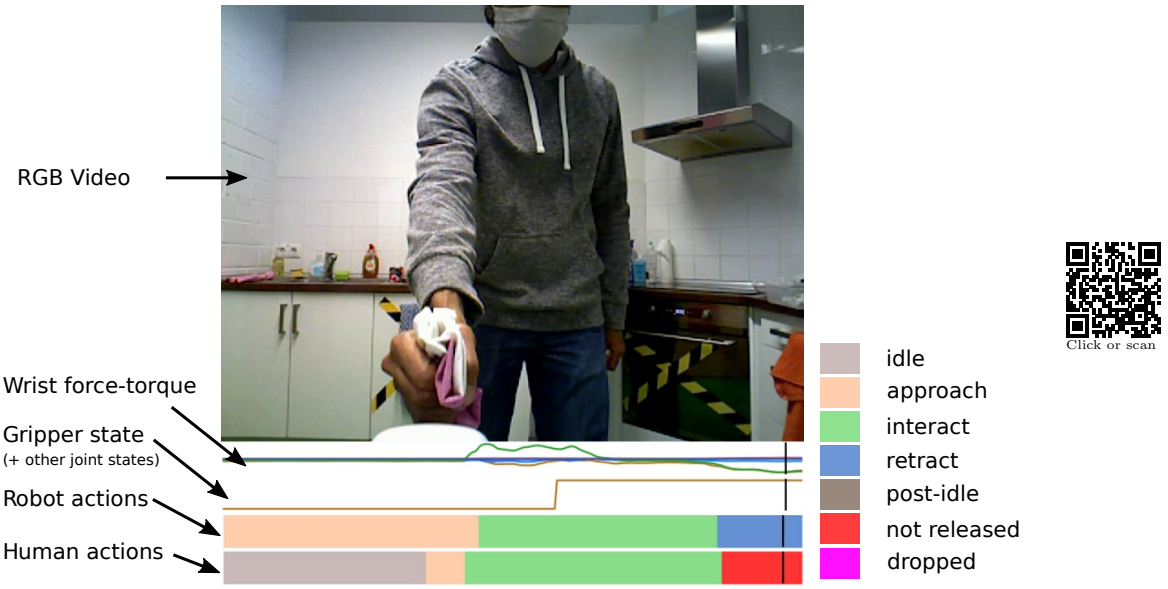


Figure 5.2: The HFD dataset consists of video, force-torque readings, and robot joint states, and contains annotations for the robot and human actions.

5.2.2 Approach

The input data consists of a sequence of T video frames $I_{1:T} = (I_1, ..., I_T)$, F-T measurements $ft_{1:T} = (ft_1, ..., ft_T)$ and the state of the gripper $g_{1:T} = (g_1, ...g_T)$. The vector ft consists of the force (f_x, f_y, f_z) and torque (τ_x, τ_y, τ_z) in three directions expressed in Newton and Newton-meter, respectively, and $g_t \in$ {-0.5, 0.0, 0.5} which refer to {`open`, `partially closed`, and `closed`}, respectively. The gripper is `partially closed` when it is grasping an object and is `closed` when the gripper further closes such that the finger joints cross a threshold. The latter case typically occurs when the gripper closes with no object, but also occurs when the gripper is grasping a thin object such as a towel. Additional annotations are the current human action $h_{1:T} = (h_1, ..., h_T)$ being performed, where $h_t \in A_H$ and which is only available during training, and the current robot action $r_{1:T} = (r_1, ..., r_T)$ being performed, where $r_t \in A_R$ and which is available during training and inference time. Given $I_{1:T}$, $ft_{1:T}$, $g_{1:T}$ and $r_{1:T}$, the task of a learning-based model, f_θ, parameterized by θ, is to perform *failure classification* to determine the outcome, o, of the handover action, such that $o = f_\theta(x_{1:T}, ft_{1:T}, g_{1:T}, r_{1:T}) \in \{$`success`, `no approach`, `no grasp`, `drop`, `no release`$\}$, where `no grasp` only applies to R2H handovers and `no release` only applies to H2R handovers.

Since a sequence of inputs is available, we consider two methods that can handle sequential data. Firstly, we make use of pre-trained I3D [48] models for video, and 1D convolutional networks for the F-T and gripper data. Since I3D models accept a sequence of 64 frames, we sample 64 frames (and corresponding F-T and gripper data) at equal intervals from the input sequence. Secondly, we use the temporal action segmentation model, MS-TCN, for video, and 1D convolutional networks for F-T and gripper data. The MS-TCN network accepts input features of arbitrary sequence length, and we therefore use frozen I3D features for every video frame, and the output of trainable 1D convolutional networks for the corresponding F-T and gripper data. We describe these two approaches in more detail in the following sections.

5.2.2.1 Video Classification

The I3D network requires 64 frames of size 224×224, and outputs a vector of scores corresponding to the classes. We use RGB and optical flow I3D networks, and replace the final layer with a fully-connected (FC) layer to output logits corresponding to o. For the input, we sub-sample 64 equidistant RGB frames from the input video $\tilde{I} \sim \text{Subsample}(I_{1:T}, 64)$, and the corresponding 2-channel optical flow frames. The frames are cropped to 448×448 (randomly during training, and center-cropped at inference time), and resized to 224×224.

The F-T and gripper data are similarly sub-sampled to a sequence-length of 64, and the F-T measurements are normalized based on the standard deviation and mean of measurements in the training set. We use two trainable 1D convolutional blocks for both modalities, as illustrated in Fig. 5.3. The 1D convolutional block consists of a 1D convolutional layer, batch normalization layer and a Rectified Linear Unit (ReLU) activation layer. The video, F-T and gripper modalities are combined using late and intermediate fusion in four variants, as illustrated in Fig. 5.3. For late fusion, at inference time, we sum the logits from the individual modalities before applying the `softmax` function. For intermediate fusion, we concatenate the features along the channel dimension.

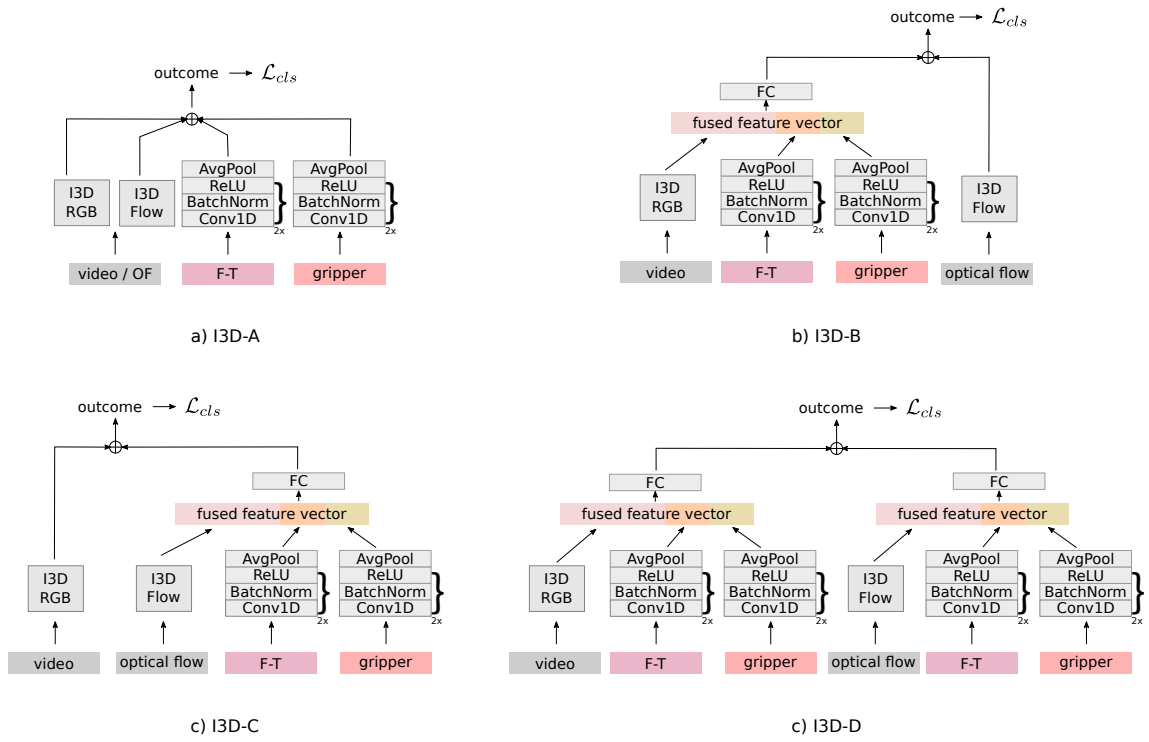


Figure 5.3: a) **I3D-A:** I3D network with late fusion of modalities. b) **I3D-B:** I3D network with intermediate fusion of F-T and gripper features with RGB stream. c) **I3D-C**, I3D network with intermediate fusion of F-T and gripper features with the Flow stream. d) **I3D-D:** I3D network with intermediate fusion of F-T and gripper features with RGB and Flow streams.

Thus, for the various networks, the inputs are the RGB frames $\tilde{I} \in \mathbb{R}^{3\times64\times224\times224}$, optical flow frames $\widetilde{of} \in \mathbb{R}^{2\times64\times224\times224}$, F-T data $\widetilde{ft} \in \mathbb{R}^{6\times64}$, and gripper state $\tilde{g} \in \mathbb{R}^{1\times64}$.

(a) **I3D-A**: networks to classify the outcome from video (with separate RGB and optical flow streams), F-T and gripper state are trained separately, and their outputs are combined with late fusion.

(b) **I3D-B**: intermediate fusion and a FC layer are used to combine the features of the RGB stream, F-T and gripper, whose output is combined with the output of the optical flow stream with late fusion.

(c) **I3D-C**: intermediate fusion and a FC layer are used to combine the features of the optical flow stream, F-T and gripper, whose output is combined with the output of the RGB stream with late fusion.

(d) **I3D-D**: intermediate fusion is used to combine features of the F-T and gripper streams with both RGB and optical flow features, before applying late fusion to combine the two outputs.

We use the cross-entropy loss to train the networks to predict the outcome of the network as defined in Eq. 5.1, where $\hat{o}_c$ is the predicted softmax score for the ground-truth class c.

$$\mathcal{L}_{cls} = -\log(\hat{o}_c) \tag{5.1}$$

Intermediate and late fusion of features from various modalities is a common approach in the literature for learning-based tasks. For example, Li et al. [158], use pre-trained networks to extract features from RGB images and tactile sensor images and use fully-connected and LSTM layers to detect slipping objects. In FINO-Net [9], intermediate fusion of visual and audio features is performed for failure detection. Lee et al. [41] fuse features from RGB, depth, force-torque and prioprioceptive sensors for self-supervised representation learning. In their case, they use CNNs for RGB and depth data, a causal temporal convolutional network force-torque data, and a MLP for proprioceptive data.

5.2.2.2 Temporal Action Segmentation

The failures in the dataset are caused either by the person not performing the expected action (such as not reaching out) or performing an unexpected action (such as dropping the object). Therefore, we consider the auxiliary task of classifying the actions of the human as a means of detecting unperformed and unexpected actions. Temporal action segmentation addresses this task by classifying every frame of the input sequence into an action class. We thus use a temporal action segmentation model to jointly classify the actions of the human, classify the actions of the robot and to classify the outcome of the task. Recent approaches for temporal action segmentation in untrimmed videos make use of temporal convolutions [6, 159, 160] or transformer-based architectures [161]. We select MS-TCN [6] due to its good performance on action segmentation datasets and since transformer-based methods require larger datasets. Although improved versions of MS-TCN exist [57], we use the original architecture here for the baseline.

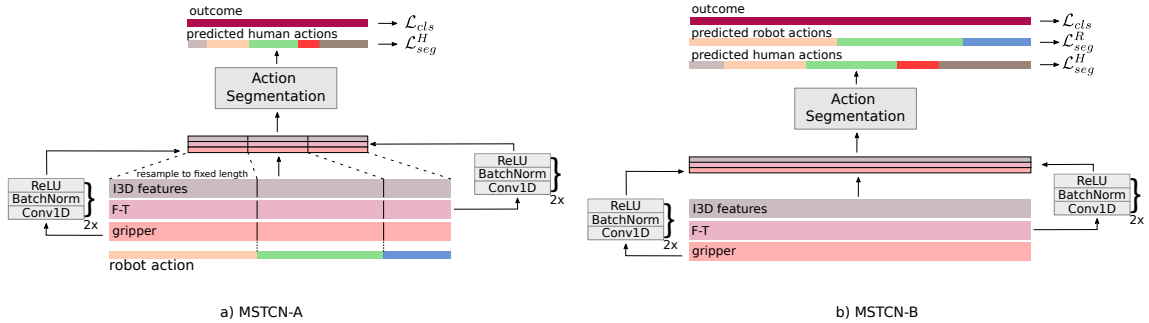


Figure 5.4: a) **MSTCN-A:** MSTCN network that takes I3D, F-T and gripper features as input and predicts the human actions for each frame and the outcome. The input is resampled to a fixed length based on the robot actions. b) **MSTCN-B:** MSTCN network with original-length inputs and an additional prediction head for the robot actions.

As described in Section 2.5, MS-TCN uses dilated temporal convolutions on a sequence of clip-level features to perform action segmentation. Each stage of the network outputs a prediction, which is refined by the next stage. Here, we use a 2-stage MS-TCN network with 10 layers each for segmenting the actions of the human and extend it to the multimodal setting. RGB and optical flow features are extracted for each frame from pre-trained I3D networks using a 64-frame window from the past, resulting in a feature vector of length 2048 for each frame. F-T and gripper state are passed through trainable 1D convolutional blocks (identical to the ones used in Section 5.2.2.1), producing 16-dimensional features each before being concatenated to the I3D features. Therefore, the input to the segmentation network is a sequence of feature vectors $F \in \mathbb{R}^{T \times 2080}$, representing the video, F-T and gripper state. The target for the network is the sequence of human actions $h_{1:T}$ and the overall outcome o. The predicted overall outcome is output as a sequence $\hat{o}_{1:T}$, although only $\hat{o}_T$ is considered at test time. Compared to the video classification approach which samples 64 frames from the input, this approach makes use of the full sequence of inputs.

Since a handover is a coordinated task, we expect that the actions of the participants are correlated to each other in the nominal case; i.e. we expect that both participants would approach each other at a similar time, the `interact` action happens simultaneously, and both participants also retract at a similar time. Fig. 5.5 shows histograms for each human action corresponding to the robot action being performed in nominal R2H handovers from the training set. As expected, it can be seen, for example, that the person typically approaches the robot when the robot is approaching the person and during the initial stages of `interact`. In order to exploit this correlation, we consider two ways in which the robot action can be used to improve the classification of the human action (see Fig. 5.4):

(a) **MSTCN-A**: the input features are resampled such that the features corresponding to each robot action are represented by a fixed length sequence. The input features therefore have a dimension of $F \in \mathbb{R}^{80 \times 2080}$, with a sequence of 16 features being sampled from each of the robot's five actions.

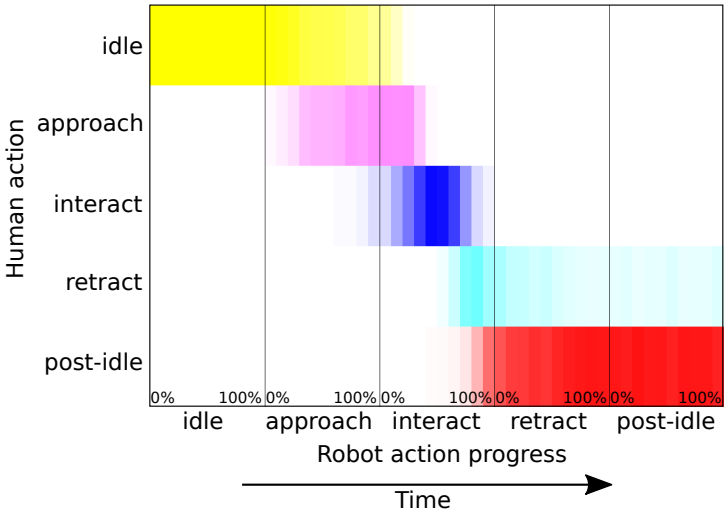


Figure 5.5: Frequency of each human action corresponding to the robot's actions in the training set for nominal R2H handovers. The progress of each robot action from 0% - 100% is discretized into 10 bins.

(b) **MSTCN-B**: the original sequence length is used, but the network predicts the robot action in addition to predicting the human action and overall outcome. The input features have a dimension of $F \in \mathbb{R}^{T \times 2080}$.

The classification loss for the outcome is summed over all frames t and all stages s of the multi-stage network, as defined in Eq. 5.2.

$$\mathcal{L}_{cls} = -\sum_{s} \frac{1}{T} \sum_{t} \log(\hat{o}_c^{s,t}) \tag{5.2}$$

The segmentation loss (Eq. 5.5) is a sum of the classification loss of the robot or human action (Eq. 5.3), and a truncated MSE as a smoothing loss (Eq. 5.4) which reduces over-segmentation [6] by discouraging changes in log probabilities between consecutive frames. The smoothing loss is weighted by $\lambda = 0.15$ and we use $\tau = 4$ as in [6]. The human action and robot action segmentation loss terms, $\mathcal{L}_{seg}^{H}$ and $\mathcal{L}_{seg}^{R}$, are also summed over all stages s of the multi-stage network. $\hat{y}_c^{s,t}$ is the probability of class c at time t and stage s, where c corresponds to the ground-truth human or robot action, respectively.

$$\mathcal{L}_{segcls} = -\sum_{s} \frac{1}{T} \sum_{t} \log(\hat{y}_c^{s,t}) \tag{5.3}$$

$$\mathcal{L}_{smooth} = \sum_{s} \frac{1}{TC} \sum_{t,c} \min(|\log(\hat{y}_c^{s,t-1}) - \log(\hat{y}_c^{s,t})|, \tau)^2 \tag{5.4}$$

$$\mathcal{L}_{seg} = \mathcal{L}_{segcls} + \lambda \mathcal{L}_{smooth} \tag{5.5}$$

5.2.2.3 Metrics

All networks are trained five times, and we report the mean of the classification accuracy for the outcome and the segmentation metrics, which are frame-wise accuracy and segmental F_1-score [159] at overlapping thresholds of 10%, 25% and 50%.

5.2.3 Results

Table 5.3: Main results for the HFD dataset showing the impact of modalities, loss terms and network

		Modality			Loss function			
ID	**Model**	**V**	**F-T**	**G**	$\mathcal{L}_{\mathbf{cls}}$	$\mathcal{L}^{\mathbf{H}}_{\mathbf{seg}}$	$\mathcal{L}^{\mathbf{R}}_{\mathbf{seg}}$	**Accuracy**
1	I3D-A	✓			✓			**64.8**
2	I3D-A		✓		✓			29.4
3	I3D-A			✓	✓			38.2
4	I3D-A	✓	✓	✓	✓			65.4
5	I3D-B	✓	✓	✓	✓			66.5
6	I3D-C	✓	✓	✓	✓			64.1
7	I3D-D	✓	✓	✓	✓			**67.9**
8	I3D-D	✓			✓			64.8
9	I3D-D	✓	✓		✓			**69.1**
10	I3D-D	✓		✓	✓			67.3
12	I3D-D	✓	✓	✓	✓			67.9
13	MSTCN-A	✓			✓			57.9
14	MSTCN-A	✓			✓	✓		63.8
15	MSTCN-B	✓			✓			56.6
16	MSTCN-B	✓			✓	✓		62.7
17	MSTCN-B	✓			✓	✓	✓	**67.8**
18	MSTCN-A	✓	✓		✓	✓		67.3
19	MSTCN-A	✓		✓	✓	✓		65.6
20	MSTCN-A	✓	✓	✓	✓	✓		**71.4**
21	MSTCN-B	✓	✓		✓	✓		69.0
22	MSTCN-B	✓		✓	✓	✓		66.4
23	MSTCN-B	✓	✓	✓	✓	✓		**71.1**
24	MSTCN-B	✓	✓	✓	✓	✓	✓	70.7

V: Video/I3D features, F-T: force-torque, G: gripper

The main results are shown in Table 5.3, in which the networks are trained using different combinations of input modalities and loss functions. For I3D-A with individual modalities (1-3), the video modality performs better than F-T or gripper alone. In terms of fusion strategies (4-7), I3D-D, in which F-T and gripper features are combined using intermediate fusion with both

RGB and flow features, performs best. It also outperforms the uni-modal variants. Intermediate fusion of RGB, F-T, and gripper features (I3D-B) also shows improved performance compared to I3D-A and I3D-C. When we analyze the impact of each modality for I3D-D (8-12), we observe that the combination of video and F-T performs best.

We investigate multitask learning with MSTCN by considering only video as input (13-17). We observe a clear improvement when $\mathcal{L}_{\mathbf{seg}}^{\mathbf{H}}$ is added as a second task. Overall, MSTCN-B with all three tasks performs best, also performing better than the video-only I3D-A model (1).

In the case of multimodal inputs (18-24), both MSTCN variants show the best performance with all three modalities, with a marginal drop in performance when using the robot action segmentation loss term for MSTCN-B (24). The latter is not surprising since the additional modalities provide information about the robot action such that predicting the robot action is not a challenging task anymore. We also note that the video + F-T combination (18, 21) performs better compared to video + gripper (19, 22), as with I3D-D (9, 10). Sample outputs from I3D-D and both MSTCN variants for a trial in which the object was not released can be seen Fig. 5.6, and additional videos can be seen by following the link in the QR code.

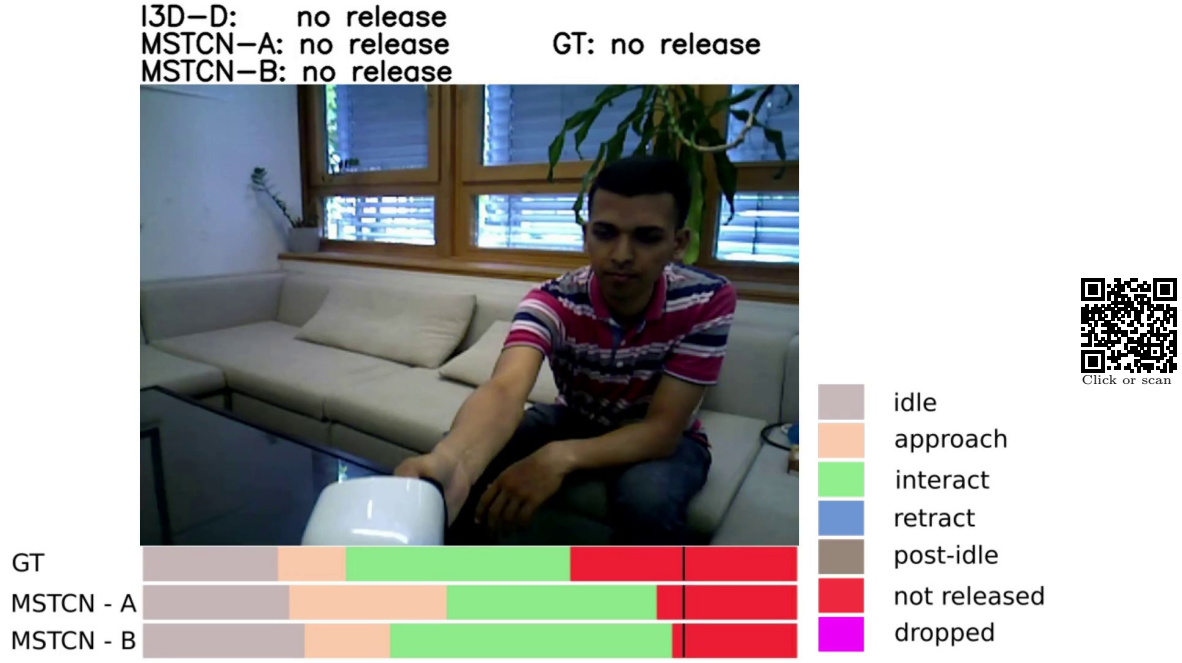


Figure 5.6: Sample outcomes from the I3D-D, MSTCN-A and MSTCN-B networks for the HFD dataset. The ground truth (GT) human action segmentation and predicted segmentations are also visualized.

5.2.3.1 Action Segmentation

Table 5.4 shows the human action segmentation metrics. We also include a baseline which is not learned, but is derived from the correlation between human and robot actions (Fig. 5.5),

Table 5.4: Segmentation Metrics

Model	**F_1@{10,25,50}**			**Frame-wise accuracy**
Baseline				
Correlation	58.1	48.6	30.8	44.5
I3D features only				
MSTCN-A	74.7	68.0	51.1	70.5
MSTCN-B	67.9	62.7	48.2	69.6
I3D features, F-T and gripper				
MSTCN-A	80.3	74.1	57.1	75.9
MSTCN-B	70.8	65.2	49.7	73.2
I3D features, F-T and gripper (only $\mathcal{L}_{\mathbf{seg}}^{\mathbf{H}}$)				
MSTCN-A	74.7	70.7	56.7	76.6
MSTCN-B	73.5	69.3	56.6	77.0

by selecting the most likely human action given the current robot action and progress from the relative occurrences in the training set. We observe that MSTCN-A is able to better segment human actions, with increased performance for both variants using multimodal inputs.

5.2.3.2 Robot Generalization

Table 5.5: Accuracy for robot generalization (Train→Test)

Model	**HSR→HSR**	**HSR→Kinova**	**Kinova→HSR**	**Kinova→Kinova**
I3D-D	51.1	43.0	55.1	62.2
MSTCN-A	70.1	56.3	47.6	48.9
MSTCN-B	70.5	49.3	44.7	56.3

We train the networks using only data from one robot, and evaluate on both to investigate how well the networks are able to generalize to a new robot. Since F-T and gripper data are represented similarly for both platforms, only the viewpoint, appearance and motions of the robot are the factors which differ between the two. The results are shown in Table 5.5. The performances in all cases are lower than when trained with the full dataset, but we note that the MSTCN variants are able to generalize better when trained on the larger HSR dataset, whereas I3D-D generalizes better when trained on the smaller Kinova dataset.

5.2.3.3 Effect of Training on Tasks Separately

All previous experiments jointly trained on R2H and H2R. When the networks are trained on each task separately (see Table 5.6) the results are comparable, but slightly lower on average compared to training on both tasks together.

Table 5.6: Accuracy for separately trained R2H and H2R networks

Model	R2H	H2R
I3D-D	67.2	61.5
MSTCN-A	69.0	65.2
MSTCN-B	68.5	75.3

5.2.3.4 Causal Convolutions

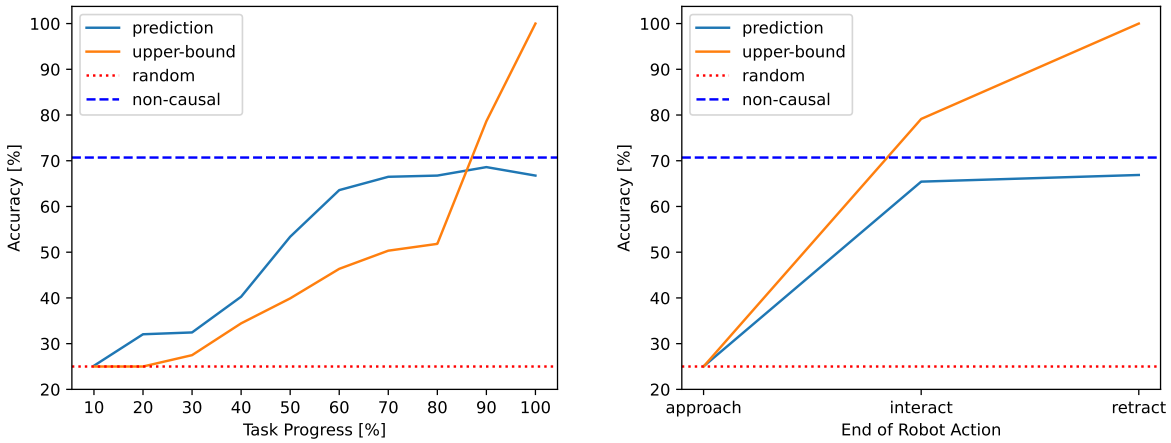
Table 5.7: Accuracy for MSTCN-B with causal convolutions

Video	F-T	Gripper	Accuracy
✓			61.1
✓	✓		62.6
✓		✓	62.6
✓	✓	✓	66.9

The MS-TCN model uses non-causal convolutions, which means that the full sequence of inputs is required to predict the actions and outcome. By using causal convolutions (as illustrated in Fig. 2.8 in Section 2.5), the outcome can be predicted based on the available inputs up to the current time. In Table 5.7, we show the results of training the MSTCN-B model using causal convolutions with all three loss terms. The results are based on the final outcome predicted when all inputs are available.[1] Although all inputs are available for predicting the final outcome in both the causal and non-causal cases, we see a drop of 4-8% in the accuracy when using causal convolutions. In addition to the final outcome, in Fig. 5.7 we show the predicted outcome at different stages during the execution, both as a percentage of the task progress and at the end of each of the `approach`, `interact` and `retract` robot actions. We compute and visualize an upper-bound accuracy at each stage, which is based on the earliest time when the outcome can be definitively determined by an external observer. For a nominal trial, the earliest time is considered to be the end of the trial. For the outcomes `no approach` and `no grasp`, the earliest observation time is defined as the start of the `retract` action of the robot. For the outcomes `drop` and `no release`, the observation time is defined as the start of the `Dropped` and `Not Released` actions of the human. If an outcome cannot be determined

[1] We use a 2-stage model with 10 layers in each stage, and none of the inputs exceed 1024 frames, thus the last output neuron has access to all inputs.

at a given stage, a random outcome is selected. The non-causal result shown in the figure corresponds to row 24 in Table 5.3. We find that for the majority of the correct predictions, the outcome is determined before the end of the `interact` action (i.e. between 60-90% of the task progress). Until about 80% of the task progress, we also note that the predicted accuracy is greater than the upper-bound accuracy, suggesting that early indications of a failure are observable and learned by the network (such as if a person does not begin approaching the person immediately).



(a) Task Progress vs Accuracy

(b) Robot Action vs Accuracy

Figure 5.7: The prediction accuracy at different time points corresponding to a) the task progress percentage and b) the end of the three robot actions. The upper-bound accuracy, which is based on the earliest time when the outcome can be determined, the random accuracy, and a reference accuracy for the non-causal model are also plotted.

5.2.4 Discussion

Overall, the results show that while video is an essential modality for detecting these failures, the addition of F-T in particular, improves the performance. Adding the robot action segmentation loss for MSTCN-B improves the performance in the unimodal setting, but not in the multimodal setting. I3D-D, MSTCN-A and MSTCN-B all perform well for the failure detection task, whereas MSTCN-A shows a better performance overall with better segmentation accuracy. In terms of interpretability of the results, the predicted human actions with the MSTCN networks can be used to analyze the reasons for the failures and when they occur. The causal MSTCN network performs worse than the non-causal version and the results indicate that failure anticipation is an interesting direction for future work. Although the best performing model is likely to detect a majority of the failures, further improvement is needed to achieve the reliability required for deployment on actual robots.

5.3 Grasp Failure Detection

In this section, we describe our experiments with the Visual Tactile dataset [8], which consists of pick-and-place trials by a robot arm equipped with an Intel Eagle Shoal hand. As described in Section 3.3, the VT dataset consists of videos, tactile data and finger motor position data during the execution of a pick-and-place task on a tabletop. The outcome of a trial is either *success* or *failure*. We first describe additional annotations and pre-processing of the dataset and then present our approach for failure detection and failure localization.

5.3.1 Data Processing and Representation

In this section, we describe additional annotations we contributed to the dataset, the representation of the tactile and joint position data and their alignment with the video, pre-processing and representation of the video, and dataset splits.

5.3.1.1 Annotations

The task is executed as a sequence of actions $A_R = \{$`pre-grasp`, `grasp`, `lift`, `release`$\}$, during which the robot arm approaches, grasps, lifts and finally places the object back on the table. The original dataset contains labels for the outcome of each trial, and annotations for the start of each action of the task, which are provided as indices of the tactile/joint data. In order to facilitate *failure localization*, we annotate the start and end timestamps when the failure occurs for each trial with a failed outcome. Thus, frames which fall within the range of the start and end timestamps are labeled as `failure`, and the remaining frames are labelled as `success`.

The tactile and joint position data streams are not aligned with the video data and are recorded at different frequencies. The data from the tactile sensor and joint motors are recorded for 400 steps for 24 seconds at a frequency of 16.67 Hz, whereas the video is recorded at 18 Hz. In order to align all sensor modalities, the tactile and joint data are first upsampled to match the 18 Hz frame rate of the video. We then annotate the offset between the two sensor streams by visually identifying the video frame when the fingers start grasping the object, and compute the difference to the already annotated start of the `grasp` action (which is the corresponding frame in the tactile/joint data when the grasp begins). Thus, the offset in frames is computed as:

$$\text{offset}_v = t_{gv} - t_g * \frac{18}{16.67} \tag{5.6}$$

where t_{gv} is our annotation of the video frame when the fingers start grasping the object and t_g is the provided label for step number in the tactile and joint data when the grasp begins. The video is shifted by offset_v to align it with the other two sensor streams.

5.3.1.2 Tactile and Joint Position Data

Prior work which uses tactile sensors for learning have represented the data as images, since the physical form of the sensor is a grid [162,163]. Thus CNNs pre-trained on image data have been fine-tuned for tactile sensor data to perform tasks such as recognizing objects [162, 163], slip

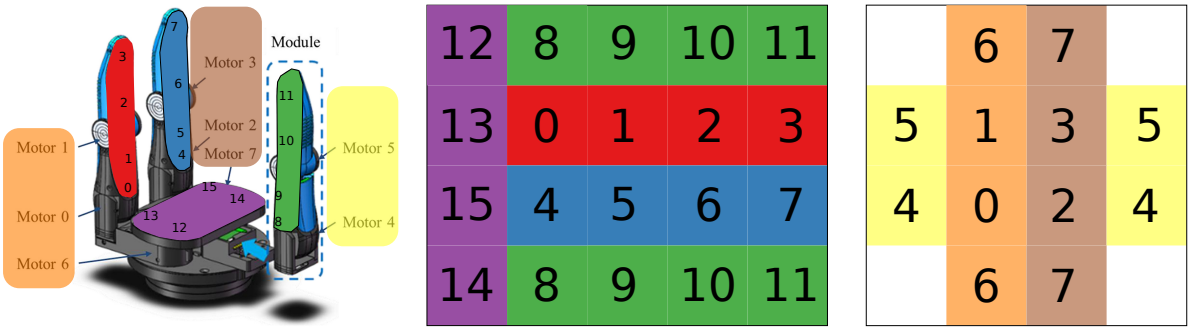


Figure 5.8: **Left:** Shoal hand (adapted from [8]) **Middle:** Arrangement of tactile sensor data in a 2D-grid to capture spatial relationships between sensors **Right:** Arrangement of joint motor positions in a 2D-grid to capture spatial relationships between the motors.

Click or scan

Figure 5.9: The temporal boundaries of the robot's actions in the VT dataset are annotated. We additionally annotate the start and end of the failure segments and represent the tactile sensor data and finger joint positions single channel images.

detection [158, 164], and feature extraction for imitation learning of fine pinch-grasps [165]. In the VT dataset, as seen in Fig 5.8, the tactile sensors do not form a grid; therefore, the sensor readings are represented as 1D vectors, $t_{vec} \in \mathbb{R}^{16}$. Nevertheless, each sensor does have a fixed spatial relationship with its neighbours, which allows us to construct a grid that respects those spatial relationships. For example, sensor 0 has the sensors 1, 4 and 13 as immediate neighbours. In Fig. 5.8, we illustrate the grid with which we represent all tactile sensors. The sensors on the palm form the first column, whereas the sensors on the three fingers are represented as rows, with the thumb (sensors 8-11) represented twice in the image to account for its proximity to the other two fingers. This results in a single channel 5×4 image, $t_{image} \in \mathbb{R}^{1\times5\times4}$, in which the pixel intensities correspond to the normalized force in milli-Newton. Each channel is normalized to the range $[0, 1]$ by dividing by the maximum sensor value for each channel in the training set.

Similarly, the joint position data, $p_{vec} \in \mathbb{R}^8$, can also arranged in a grid, with motors from each finger represented as columns, and the positions of the thumb being duplicated to account for its proximity to both other fingers. Positions for motor 6 and 7 are also duplicated such that they are placed next to both other motors of their respective fingers. Thus, this forms a single channel 4×4 image, $p_{image} \in \mathbb{R}^{1\times4\times4}$, with pixel intensities corresponding to the max-normalized joint position in degrees, and the four corners being padded with zeros. A visualization of the image representations, robot actions and annotated failure segments can be seen in Fig. 5.9.

5.3.1.3 Video

The task execution is captured by two cameras at 18 Hz and are about 20 seconds long. RGB and depth video are captured by the front camera, whereas only RGB video is captured by the left camera. We use the 3-channel RGB video from the front camera in our work. For the frame-wise failure classification approach described in Section 5.3.2.3, RGB features for each frame are extracted from a pre-trained I3D model.

5.3.1.4 Dataset Split

The trials are split based on the object being manipulated, referred to as TTobject in [144]. Videos of eight objects are used for training (1299 trials), and two for testing (358 trials), resulting in a train-test ratio of 78:22.

5.3.2 Approach

We experiment with one approach that performs failure detection and two approaches that perform failure localization.

5.3.2.1 Failure Detection

For failure detection, we use a variant of the I3D-B model described in Section 5.2.2.1. The optical flow branch of the model is omitted, and tactile and joint position data are used instead of the F-T and gripper position data. We sample 64 equidistant frames from the full execution.

As in Section 5.2.2.1, the RGB frames are cropped and resized to 224×224 resulting in an input $\tilde{I} \in \mathbb{R}^{3\times64\times224\times224}$. The image representation and 1D vector representations of the tactile and joint position data are used, such that the inputs are tactile images $\tilde{t}_{image} \in \mathbb{R}^{1\times64\times5\times4}$, joint position images $\tilde{p}_{image} \in \mathbb{R}^{1\times64\times4\times4}$, tactile 1D vectors $\tilde{t}_{vec} \in \mathbb{R}^{1\times64\times16}$ and joint position 1D vectors $\tilde{p}_{vec} \in \mathbb{R}^{1\times64\times8}$. For the $\tilde{p}_{image}$ and $\tilde{t}_{image}$, a 3D convolutional block is used; the block is comprised of two 3D convolutional layers, ReLU activation layers, a 3D max pooling layer, a FC layer and a dropout layer, as shown in Fig. 5.10. For $\tilde{t}_{vec}$ and $\tilde{p}_{vec}$, we use the same 1D convolutional block as in Section 5.2.2.1.

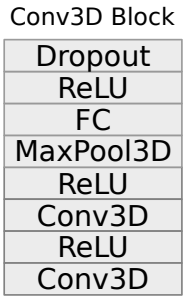


Figure 5.10: The 3D convolutional block used to encode tactile and joint position images for the I3D-B model.

5.3.2.2 Clip-Wise Failure Localization

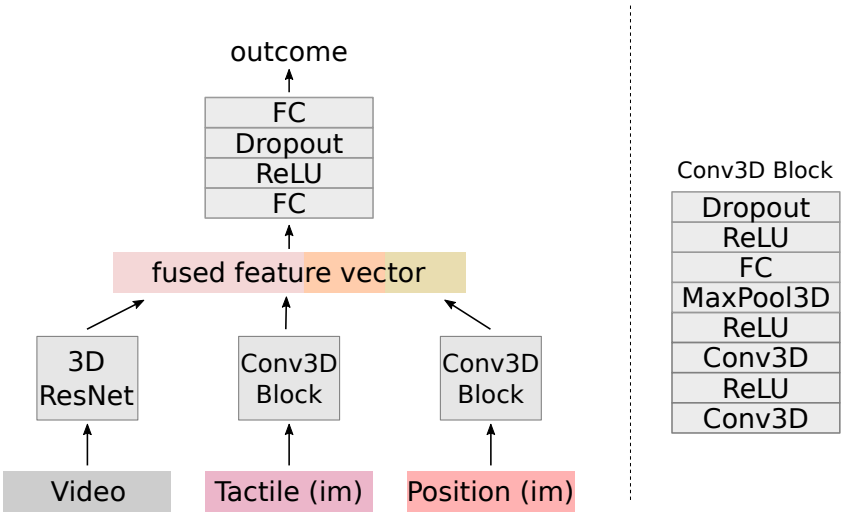


Figure 5.11: The $3DMNN_{int}$ model is used to classify 1-second clips as `success` or `failure`. The 3D convolutional block used for tactile and joint position images is shown on the right.

In [144], we classified 1-second (18 frames) clips from the video as `success` or `failure` using the $3D\text{-}MNN_{int}$ model illustrated in Fig. 5.11. The inputs are 18-frame clips, such that the video

from each trial contains multiple clips. The model uses a 3D ResNet [166,167] pre-trained on the Kinetics dataset for the video clips, and a 3D convolutional block for the other two modalities, which are represented as images. The RGB frames are resized to 112×112, resulting in input clips $\tilde{I} \in \mathbb{R}^{3\times18\times112\times112}$. The tactile and joint position inputs are $\tilde{t}_{image} \in \mathbb{R}^{1\times18\times5\times4}$ and $\tilde{p}_{image} \in \mathbb{R}^{1\times18\times4\times4}$ respectively.

We originally reported the clip-wise accuracy for failure detection in [144]. Here, in order to compute the frame-wise accuracy, we apply the predicted outcome of the 1-second clips to all frames within the clip and compare it to the frame-wise labels[2].

5.3.2.3 Frame-Wise Failure Localization

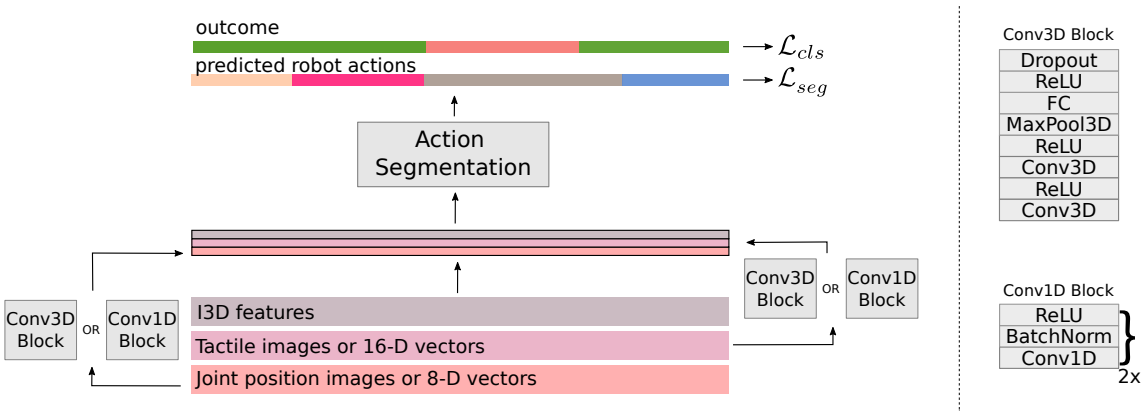


Figure 5.12: The temporal action segmentation model jointly performs robot action segmentation and failure localization.

Since the clip-wise classification described in Section 5.3.2.2 performs failure localization in a course-grained manner, for a fine-grained frame-wise failure localization we use the MS-TCN model, similar to Section 5.2.2.2. The model, illustrated in Fig. 5.12, jointly segments the video into the `pre-grasp`, `grasp`, `lift` and `release` robot actions and classifies each frame as `success` or `failure`. The input to the model is a sequence of features corresponding to each frame. Similar to the failure detection I3D model, we use trainable 3D convolutional blocks and 1D convolutional blocks to extract features from the image-based and 1D vector representations of the tactile and joint position data. For the image-based representation, a 64-frame clip from the past is passed through the convolutional blocks to output 16-dimensional features for each frame. For the 1D vector representation, the full sequence of tactile and position data are passed through 1D convolutional blocks, which output a sequence of 16-dimensional feature vectors as well. For the video, we use the sequence of 1024-dimensional feature vectors extracted from the RGB I3D network. Thus, when using all three modalities, the input features have a dimension of $F \in \mathbb{R}^{T\times1056}$.

[2]In [144], the clip-wise precision, recall and F_1 score were reported by considering the `success` class as the positive class. Here we consider the `failure` class as the positive class.

We use the same segmentation losses as in Section 5.2.2.2 for both the robot action segmentation ($\mathcal{L}_{seg}^{R}$) and failure localization ($\mathcal{L}_{cls}$), except that we use the binary cross entropy loss function instead of cross entropy for failure localization.

5.4 Results

All models are trained 5 times and we report the mean of precision, recall and F_1 score for failure detection and frame-wise failure localization.

5.4.1 Failure Detection

Table 5.8: Failure detection results for the VT dataset in which the tactile and joint position data are represented as images or 1D vectors.

Modality			Image			1D		
V	**T**	**P**	**Precision**	**Recall**	**F_1**	**Precision**	**Recall**	**F_1**
✓			96.9	96.5	96.5	-	-	-
	✓		90.6	90.4	90.4	93.9	93.8	93.8
		✓	93.0	92.6	92.6	94.5	94.1	94.1
✓	✓		98.3	98.2	98.2	97.7	97.5	97.5
✓		✓	98.3	98.2	98.2	99.2	99.2	**99.2**
	✓	✓	93.8	93.4	93.4	95.4	95.2	95.2
✓	✓	✓	99.1	99.1	**99.0**	97.8	97.5	97.4

V: Video, T: Tactile, P: Joint Position

In Table 5.8, we report the failure detection results using various combinations of modalities and with both types of representations for the tactile and joint data. We find with just video data, the model achieves an F_1 score of 95.6, and the tactile and joint position data alone also have F_1 scores above 90. As expected, the multi-modal variants perform better than the individual modalities alone, with the model using all modalities achieving an F_1 score of 99.0.

Overall, when using the 1D representations for tactile and joint data, the performance improves, except in the cases where a combination of tactile and video data are used, with the video and joint data model performing best with an F_1 score of 99.2. We speculate that although the image-based representation makes the spatial relationships between the sensors explicit, the redundant data and additional trainable parameters lead to a slight drop in performance. Additionally, using a kernel size of 3 on the small images (5×4 and 4×4) potentially leads the network to attempt to learn non-existent spatial correlations.

5.4.2 Clip-Wise Failure Localization

In Table 5.9, the failure localization results using the clip-wise classification model 3D-MNN$_{\text{int}}$ are reported. The frame-wise precision, recall and F_1 score are computed, as described earlier,

Table 5.9: Failure localization results on the VT dataset using a clip-based classifier.

Video	Tactile	Joint Position	Precision	Recall	$\mathbf{F_1}$
✓			52.2	6.2	11.1
	✓		36.5	43.5	39.7
		✓	20.8	78.5	24.9
✓	✓		36.7	48.5	41.7
✓		✓	44.7	36.5	40.2
	✓	✓	40.6	47.2	**43.5**
✓	✓	✓	40.8	40.7	40.7

by applying the predicted outcome for each clip to the individual frames in the clip. As expected, the performance is quite poor, since the clip-wise model has only a (temporally) local view of the execution. Therefore, for example, it would be difficult to determine if an object is on the table because it wasn't grasped yet, or was the result of a failed grasp, or had already been placed back on the table. Nevertheless, we see that using tactile or joint position data alone performs better than video alone, and the multimodal models perform better, with the combination of tactile and joint data having the highest F_1 score of 43.5. This is likely due to the fact that both sensors change from their default state when the hand is interacting with the object or meant to be interacting with the object. In particular, contacts with the object can be easily determined with the tactile sensor, and the state of the joints can be used to determine whether the hand is open, partially closed (suggesting that an object has been grasped) or fully closed (suggesting that an object is meant to be grasped, but is not).

5.4.3 Frame-Wise Failure Localization

Table 5.10 lists the results using the MS-TCN model. Looking at the performance for different modalities, we again observe that the video-only version performs the worst, and the unimodal tactile and position data have similar performance as the multimodal variants. We see a minor improvement in results when using the image-based representations for tactile and joint data.

We find that adding the robot action segmentation loss $\mathcal{L}^R_{seg}$ (1-7) leads to a slight drop in the performance. The high accuracy of action segmentation shown in Table 5.11 suggests that the task is not challenging, and thus adding the auxiliary loss term distracts from the main objective of failure localization. We also note that for the multimodal variants, using video in the case of the single loss function leads to a minor improvement in performance (for example, row 9 and 11), and a minor decrease in performance in the case of two loss functions (for example, row 2 and 4). This also suggests that using video features in combination with the action segmentation loss adds noise to the learning process and distracts from the main objective.

Table 5.10: Failure localization results on the VT dataset using a temporal action segmentation model.

	Modality			**Loss Function**		**Image**			**1D**		
ID	**V**	**T**	**P**	$\mathcal{L}_{\mathbf{cls}}$	$\mathcal{L}_{\mathbf{seg}}^{\mathbf{R}}$	**Precision**	**Recall**	$\mathbf{F_1}$	**Precision**	**Recall**	$\mathbf{F_1}$
1	✓			✓	✓	83.3	66.5	73.9	-	-	-
2		✓		✓	✓	81.1	75.1	78.0	81.9	73.2	**77.3**
3			✓	✓	✓	82.6	73.0	77.5	82.4	72.7	77.2
4	✓	✓		✓	✓	82.9	72.4	77.3	82.7	72.5	**77.3**
5	✓		✓	✓	✓	85.7	69.7	76.9	85.2	69.1	76.3
6		✓	✓	✓	✓	81.4	75.6	**78.4**	81.6	73.1	77.1
7	✓	✓	✓	✓	✓	84.6	71.8	77.7	83.8	71.6	77.2
8	✓			✓		82.3	68.8	74.9	-	-	-
9		✓		✓		81.2	75.0	78.0	81.7	73.2	77.2
10			✓	✓		82.0	76.5	79.1	82.8	71.3	76.6
11	✓	✓		✓		83.3	74.6	78.7	80.5	75.1	77.7
12	✓		✓	✓		83.3	75.6	**79.3**	84.5	74.0	78.9
13		✓	✓	✓		80.8	75.6	78.1	81.4	73.4	77.2
14	✓	✓	✓	✓		83.3	75.6	79.2	82.9	75.9	**79.2**

V: Video, T: Tactile, P: Joint Position

Table 5.11: Frame-wise accuracy for robot action segmentation on the VT dataset

Modality			**Frame-wise accuracy**
Video	**Tactile**	**Joint Position**	
✓			98.1
	✓		98.3
		✓	**99.3**
✓	✓		98.3
✓		✓	99.0
	✓	✓	**99.3**
✓	✓	✓	99.1

5.4.4 Discussion

The failure detection performance suggests that the failed outcomes in this dataset are easy to discriminate, even when only using video data. The image-based representation of the tactile and joint position data do not bring much benefit, but do require more trainable parameters. For the more challenging task of failure localization, aggregating results from a clip-wise failure detector leads to poor performance as expected (an F_1 score of about 40.0), although the tactile and joint data perform significantly better than video alone. Using the full context of the execution with the MS-TCN model results in an improvement of the F_1 score to around 79.0, but adding the auxiliary action segmentation loss term leads to a slight drop in performance since the task is not challenging.

5.5 Summary

In this chapter we presented approaches for failure detection, classification and localization using multimodal data. First, we described our development of the HFD dataset, which includes human-induced failures during R2H and H2R handovers. Both the HFD dataset and the VT dataset contain video data in addition to sensor data that record robot joint state data and contact and interaction forces with the environment. We explore two main methods for failure detection and localization using existing base networks that have been developed for video learning tasks. Firstly, the I3D network is used to encode video data (both RGB and optical flow), and we use 1D or 3D convolutional networks to encode the other modalities. More specifically, 1D convolutional networks are used to encode the force-torque sensor and gripper state in the HFD dataset, and both 3D and 1D convolutional networks are used to encode the tactile and joint position data in the VT dataset, depending on whether they are represented as images or 1D vectors. Secondly, we use the MS-TCN network for failure classification in the HFD dataset by using it to learn to jointly classify the outcome, and segment the human's and robot's actions. We also use the MS-TCN network for the VT dataset for failure localization, in which each frame is classified as `success` or `failure`. A course-grained failure localization is also performed for the VT dataset using a clip-wise failure detection using 3D ResNets.

For both datasets, we experiment with all combinations of modalities, using both late and intermediate fusion schemes. In the case of the HFD dataset, we find that video is an essential modality, but using multimodal data, in particular the force-torque data, improves performance. The joint classification of the outcome and segmentation of the human actions also improves performance. The best performing model achieves an accuracy of 71.4%, thus there is scope for additional improvements. For the VT dataset, failure detection using the I3D and 1D/3D convolutional networks leads to an F_1-score above 90% in all cases; all three modalities individually are able to discriminate well between nominal and failed trials but the multimodal version using all three modalities (when using image-based representations) and the version using just video and joint position (when using 1D representations) perform the best. The clip-wise failure detection results are aggregated across the full video to produce failure localization results, but performs poorly, with the best performing multimodal model (using tactile and joint position data only) achieving an F_1-score of 43.5. This is expected since the

individual clips do not have the full context of the execution. We also find that for the clip-wise detector, the tactile sensor data is the most important modality, achieving an F_1-score of 39.7 when used alone. Failure localization using the MS-TCN network, that operates on features from the full video, is able to perform significantly better, with the best performing models achieving an F_1-score above 79.0. Also in this case, the tactile and joint position data alone perform better than using video features. Additionally, using the auxiliary action segmentation loss leads to a small drop in performance, presumably because the task of action segmentation is not sufficiently challenging.

The results confirm the conclusions in existing literature that multimodal data improves the performance. The importance of video data depends on the task and characteristics of the dataset. Using networks that subsample data points from the full execution or networks that require features from every frame are both viable options for failure detection and localization, although the latter comes with additional costs at inference time. In the case of the HFD dataset, training the network for the auxiliary task of human action segmentation led to an improvement in performance, particularly since it was directly related to the failure classification task.

The methods in this chapter detect failures after the completion of the execution. It would be more practical for the robot to detect failures immediately after they have occurred (or anticipate them to avoid them altogether). Our investigation of the causal MS-TCN model for detecting failures during execution shows a drop in accuracy of about 4-8%; thus research targeting online failure detection is a good candidate for future work.

CHAPTER 6

Using Task Knowledge to Enhance Failure Detection

The contents of this chapter are partially based on the following publication:

S. Thoduka, S. Houben, J. Gall, and P. G. Plöger, "Enhancing Video-Based Robot Failure Detection Using Task Knowledge" in 2025 European Conference on Mobile Robots (ECMR), Padova, Italy, 2025, ppp. 1–6, doi: https://doi.org/10.1109/ECMR65884.2025.11162998. [168]

The contributions of the authors are as follows:

- **Santosh Thoduka** Developed the main approach, enhanced the ARMBench dataset with additional annotations, ran the experiments, and wrote most of the paper.
- **Sebastian Houben** Contributed to writing, and discussions about presentation of results.
- **Jürgen Gall** Contributed to overall supervision, discussions, and writing.
- **Paul G. Plöger** Contributed to the overall supervision, discussions, and writing.

6.1 Introduction

In this chapter, we explore strategies to improve failure classification performance by incorporating readily available robot task knowledge. In Chapter 4, we used knowledge of the robot's own motions to mask regions of the image and to model expected apparent motion of the scene. In Chapter 5, our main focus was on multi-modal learning, but we also used the temporal boundaries of the robot's actions as an auxiliary task for the temporal action segmentation network. In this chapter, we propose to use the temporal boundaries of the robot's actions to guide frame selection for video classification models and to use the locations of task-relevant objects in the scene to pre-process the input frames.

As discussed in Section 1.2, failure or anomaly detection can be performed at the level of individual images or the complete video of a task execution. When individual images are used

as input to a failure classification model, the context of the task and previous observations are often ignored, but they have the advantage that failures can be detected as soon as they occur. As discussed in Chapter 5, for video classification models such as I3D, a fixed number of frames have to be sampled from the video, which encompasses more context compared to using individual images. The temporal action segmentation approach discussed in Chapter 5.2.2.2 operates on every frame of the video, and also considers previous and future observations through non-causal temporal convolutions. However, it is more computationally expensive since features need to be extracted for every frame. Our focus in this chapter is on video classification models and different strategies on how to sample frames from the video and pre-process them.

For some types of failures, it is essential to capture the moment when the failure happens (for example, a dropped object), whereas other types of failures (such as failing to open a drawer) can be detected by simply observing the pre- and post-states of the action. In the latter case, more contextual frames are necessary, whereas in the former case, the frames that capture the moment when the failure occurs are most important. This variety in failure types motivates detection strategies that are adapted to the task and expected failure types.

In existing vision-based approaches, task-specific knowledge is integrated via hard-coded heuristics, such as cropping frames to a fixed region of interest (ROI) [3,9], via natural language descriptions of the tasks for vision-language models (VLMs) [125], or via learning to predict the robot's actions as an auxiliary task (Section 5.2.2.2). For instance, the inputs to VLM-based failure detectors include a natural language description of the task [42,125] and an expectation of the nominal state [42, 119, 123]. As discussed in Chapter 5, several approaches make use of task-relevant multimodal data [2, 9, 169], by combining visual data with tactile, auditory and proprioceptive sensor data. Park et al. [102] use a task progress-based prior for deciding on an anomaly detection threshold during a robot-assisted feeding task. The robot state and current action being performed have also been used as inputs to networks performing future video prediction [170], and imitation learning [134]. As discussed in Section 3.2.1, Hegemann et al. [18] make explicit use of task knowledge by modeling tasks using a set of Markov Chains corresponding to actions performed in the task. The state transition matrices are learned using a training set of nominal executions, and failures are detected by computing the likelihood of the state trajectories.

For video action recognition models, the common practice for selecting frames is to sample a clip with a fixed stride and duration from the video during training, and to average the softmax scores from multiple clips at test time for predictions [48,171]. Some approaches employ more sophisticated strategies for selecting frames, such as motion-guided sampling [172, 173], learning-based approaches that score the importance of frames [174, 175], and feature-based similarity measures to reject redundant frames [176,177]. The rate at which frames are sampled is also an important factor, as shown by Feichtenhofer et al. [171]. The authors propose a model with *Slow* and *Fast* pathways, which applies different sampling rates to the same video for each pathway with lateral connections between the pathways. This is done both at training and inference time to effectively capture both semantic features (such as object categories, colour, etc.) using the *Slow* pathway, and temporal features (such as fast moving objects) using the *Fast* pathway. Epstein et al. [90] use frame rate prediction as a self-supervision signal for video

representation learning, applied to unintentional action detection in videos. Varying the frame rate has been used as a data augmentation method for time-series classification [178], speech-related tasks [179, 180], and for videos [181]. More specifically, Zou et al. [181] use *TDrop*, which randomly drops frames and effectively varies the frame rate for those regions. In our augmentation method, presented in Section 6.2.1.4, we instead sample larger regions of the video at different frame rates, both based on the underlying temporal boundaries of the robot's actions and randomly, which is most similar to *window warping* applied in [178].

In this chapter, we use the temporal boundaries of the robot's actions and the location of task-relevant objects to guide frame selection and pre-processing, shown in Fig. 6.1, which we extend into a training and test-time data augmentation technique. We additionally revise and augment the ARMBench [3] dataset, including fixing incorrect labels, and annotating the temporal boundaries of the robot's actions and bounding boxes for task-relevant objects.

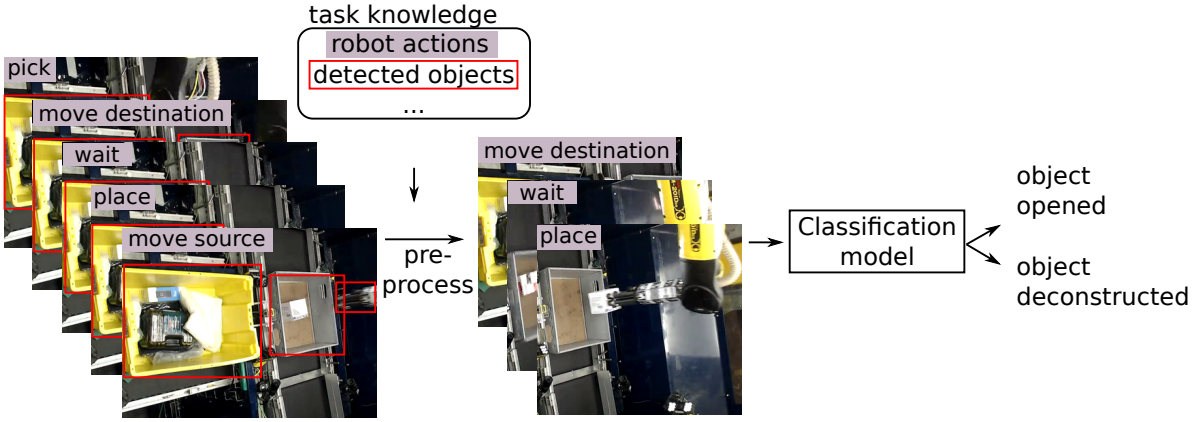


Figure 6.1: Task knowledge such as the temporal boundaries of the robot's actions and detected task-relevant objects are used in the video pre-processing step to improve failure classification.

6.1.1 ARMBench Video Defect Dataset

As described in Section 3.3, the ARMBench Video Defect dataset [3] consists of 20k video clips of a robot performing pick-and-place operations in a warehouse. The robot picks up various objects from a static source container and places them in a destination container which is stopped on a conveyor belt. Two types of object defects are caused during the pick-and-place task: *a)* `open`: the manipulated object is opened (e.g. a box cover is lifted) *b)* `deconstruction`: the manipulated object is deconstructed into multiple pieces (e.g. the contents of a box are spilled).

The characteristics of the dataset pose some challenges for video-based failure classification:

1. **Duration and resolution**: The duration of the samples ranges from 2 – 74 seconds (recorded at 30 Hz), and the video frames (with a resolution of 1280 × 560) capture two distinct regions where the robot performs its actions (i.e. the regions of the source and destination containers). Video classification models typically require clips of a fixed

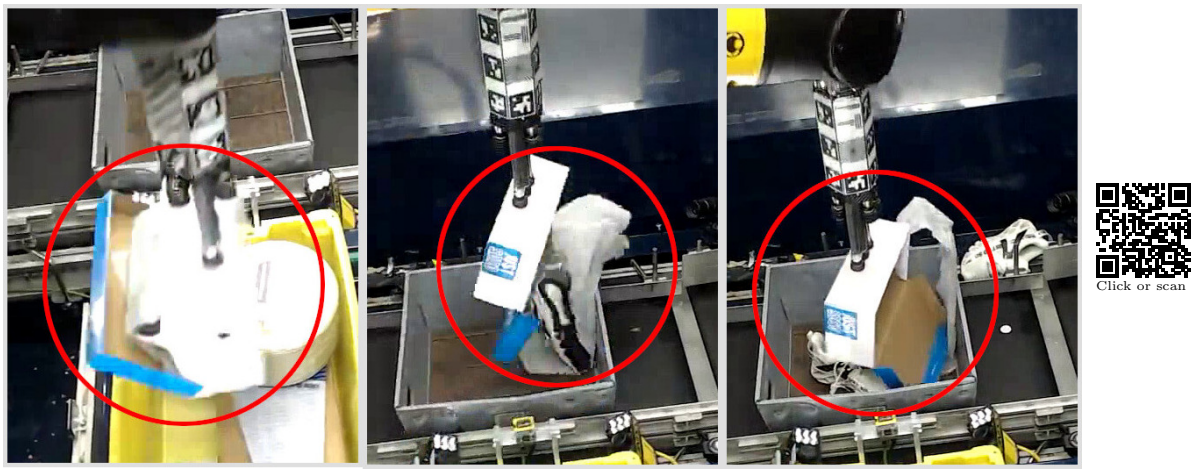


Figure 6.2: Example of a cascading failure: in the course of a `deconstruction` failure, an object first opens (left), then deconstructs (center), and finally remains open (right).

length and square resolution. With the large variance in duration, it is challenging to select a suitable frame sampling method, since the coverage of the videos will vary significantly based on their durations. For example, for the sample with a duration of 74 seconds, the robot waits a long time above the conveyor belt waiting for the correct destination container to arrive. During that time, no significant events occur. By sampling equidistant frames from the entire video, a majority of the frames would be sampled from the `Wait` action, whereas the frames when the failure occurs may be omitted.

2. **Partial and cascading failures**: For some objects, the `deconstruction` failure has three stages: *a)* the object opens, *b)* the object deconstructs, and *c)* the object remains open after deconstruction. An example is shown in Fig. 6.2. Since an `open` object could evolve into a `deconstruction` failure (or be the result of a `deconstruction`), it is essential to capture the *right* frames and to have sufficient contextual frames to classify the failure. Additionally, for some `open` failures, a view of the open object is only visible for a short duration, which again stresses the importance of the frame selection strategy.

Table 6.1: ARMBench dataset stats

Version	Split	Nominal	Deconstruction	Open	Total
Original	Train	5,838	1,520	1,399	8,757
	Test	10,000	645	606	11,251
Corrected	Train	6,370	1,460	927	8,757
	Test	10,175	630	446	11,251

We observed that approximately 1,100 samples are mislabeled in the original dataset, which we correct. In particular, a large number of `nominal` samples were labelled as `open` leading to

a large false positive rate for the open class in the original paper. The data distributions before and after correcting the mislabeled samples are shown in Table 6.1. We keep the same train-test split as in the original dataset [3].

In order to facilitate the action-based frame sampling and cropping strategies described in Section 6.2, we annotate the temporal boundaries of the robot's actions and the bounding boxes of relevant objects in the scene. In a prototypical execution of the task, the robot performs the following actions in sequence: $A = \{$Pick, MoveDestination, Wait, Place, MoveSource$\}$, as shown in Fig. 6.3. In practice, the temporal boundaries of the actions are known to the robot, for example, based on the current state of a state machine. Here, we label the temporal boundaries using a trained MS-TCN [6] model. We extract the I3D RGB features for the videos and manually annotate the robot action boundaries for 500 samples, which we use to train the MS-TCN model. The boundaries for the remaining videos are extracted using the trained model, and we manually correct over-segmentations and incorrect action classifications for a minority of the videos.

We also extract bounding boxes for the end-effector, and source and destination containers after training a YOLO-based object detector [182] using an initial set of 250 labelled images. The object detector was further trained with a dataset of 5000 manually and automatically labelled images of the end-effector.

For videos that contain failures, we annotate the timestamp at which the failure is first visible. For deconstruction samples in which the object remains open for a noticeable duration before deconstructing, we annotate two timestamps, namely, the time when the object opens, and when it finally deconstructs. The original dataset also contained failure timestamp annotations, but were found to be largely incorrect. An overview of the annotations added to the dataset can be seen in Fig. 6.3.

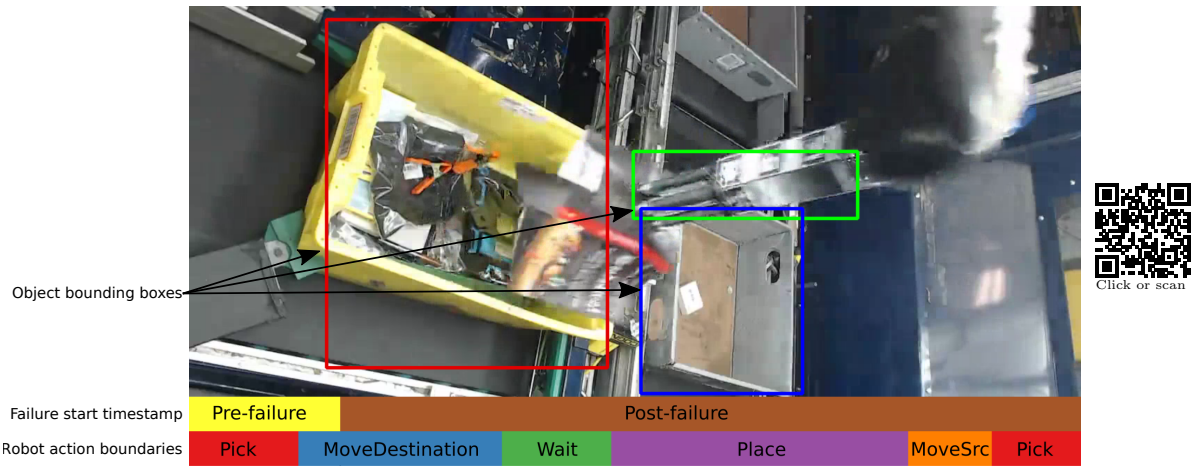


Figure 6.3: The ARMBench dataset is enhanced with additional annotations for the robot action boundaries, bounding boxes for the end-effector and containers, and the timestamp when the failure is first visible.

6.2 Method

The robot performs a high-level task that is composed of a sequence of N actions $A = (a_n)_{n=1}^{N}$, which is recorded by a camera as a sequence of RGB frames, with each action represented by l_n frames: $(I_i^{a_n})_{i=1}^{l_n}$. The complete task consists of the frames $\mathcal{I}_A = (I_1^{a_1}, \ldots, I_{l_1}^{a_1}, \ldots, I_1^{a_N}, \ldots I_{l_N}^{a_N})$. Given the frames $\mathcal{I}_A$, the task of a model f_θ, parameterized by θ, is to perform a binary or multiclass classification to determine the outcome of the task, such that $f_\theta(P(\mathcal{I}_A)) \in \{\texttt{nominal}, \texttt{deconstruction}, \texttt{open}\}$, where P is a pre-processing function that may sub-sample, resize and crop the frames in $\mathcal{I}_A$.

As in the original ARMBench paper, we use the MViT-B model [53] to classify the outcome of the task, given an input sequence of 32 frames with a resolution of 224×224. The final layer is replaced by a FC layer with two outputs, one for each defect type, and the network is trained using a binary cross entropy loss. Our main approach is based on only modifying the frame selection and pre-processing strategy, while using the identical model and hyperparameters as in the original paper.

We also report the results using two additional baselines. The first is the VGG-RGB model [9], which consists of a pre-trained VGG-16 model [183] and convolutional LSTM layers, and expects eight input frames, which we sample uniformly from the full video. The second is the MSTCN-B [6] model used in Chapter 5, for which we use I3D features (RGB only) for all frames, and train the network to jointly segment the robot's actions and classify the final outcome (i.e. with the loss functions $\mathcal{L}_{cls}$ and $\mathcal{L}_{seg}^{R}$ as defined in Section 5.2.2.2).

6.2.1 Frame Selection and Pre-processing

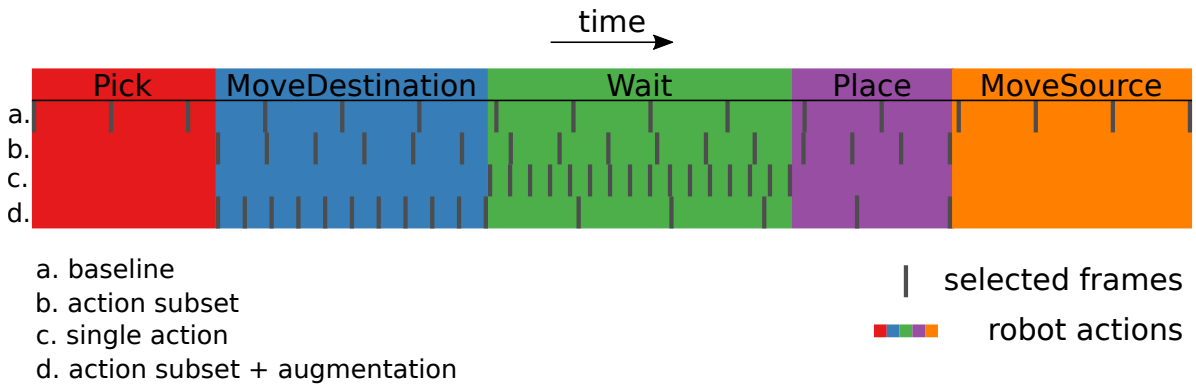


Figure 6.4: An overview of the different frame selection methods used for training the video classification model.

To motivate our exploration of frame selection methods, we performed experiments on ARMBench, in which we deliberately selected frames that were likely to lead to better failure classification performance. For videos with failures, we selected frames in the neighbourhood of the timestamp when the failures are first visible. For nominal videos, we sampled regions of the video where failures were more likely to occur based on statistics derived from the failure videos.

By training the video classification model on these selected frames, the F_1 score improved to 80.0 and 82.1 (without and with ROI cropping described in Section 6.2.1.5) compared to 77.9 and 77.4 for the baseline with uniform frame sampling under otherwise identical settings. This indicates the potential that an educated frame selection strategy holds. These scores can be considered as upper-bounds since they are based on selecting frames using the ground truth timestamps when failures occurred.

As illustrated in Figs. 6.4 and 6.6, we experiment with the frame selection and pre-processing methods described in the following sections. The methods rely on two types of task knowledge, namely, the temporal boundaries of each action, and the location of task-relevant objects, which we expect are already known to the robot at no additional expense.

6.2.1.1 Baseline

For the baseline, we follow the approach in [3], and sub-sample equidistant frames $\tilde{\mathcal{I}}_A \sim \text{Subsample}(\mathcal{I}_A, 32)$ from the full video and crop a fixed region of the image before resizing the frames to 224×224.

6.2.1.2 Action Subset

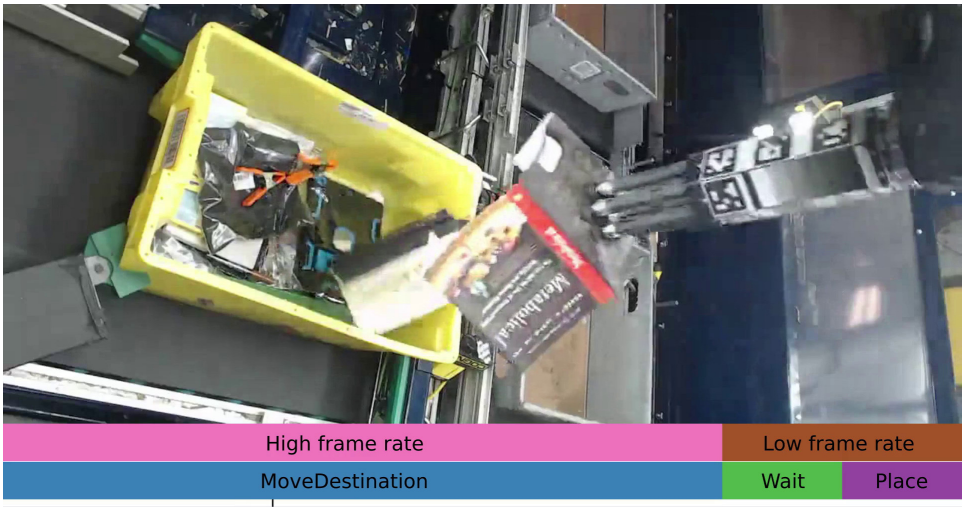
We sub-sample equidistant frames from a subset of the actions $\tilde{\mathcal{I}}_{A'} \sim \text{Subsample}(\mathcal{I}_{A'}, 32)$, where $A' = \{\texttt{MoveDestination}, \texttt{Wait}, \texttt{Place}\}$. We choose these actions since most of the robot-object interaction takes place during these actions, and the failures are most likely to occur during that time.

6.2.1.3 Single Action

We sub-sample equidistant frames from each action separately $\tilde{\mathcal{I}}_{a_m{}'} \sim \text{Subsample}(\mathcal{I}_{a_m{}'}, 32)$, where $a_m{}' \in A'$. This effectively treats each action within the task as a separate sample, so the model only predicts whether a failure has occurred during that action. The labels are adjusted based on the state of the failure at the end of the selected action. For example, if a `deconstruction` occurs during the `Wait` action, and the object is only `open` during the `MoveDestination` action, the labels for the `MoveDestination` and `Wait` samples from the same video are set to `open` and `deconstruction` respectively. Similarly, if a `deconstruction` has already occurred prior to the `Place` action, the label for the `Place` sample is set to `open`. At test time, the predictions for all actions within a task are aggregated, and the final outcome is classified as `open` only if none of the predicted outcomes are `deconstruction`, as shown in Eq. 6.1.

$$\text{outcome} = \begin{cases} \text{nominal}, & \text{if } \forall m \; f_\theta(\tilde{\mathcal{I}}_{a_m{}'}) = \text{nominal}, \\ \text{deconstruction}, & \text{if } \exists m \; f_\theta(\tilde{\mathcal{I}}_{a_m{}'}) = \text{deconstruction}, \\ \text{open}, & \text{if } \Big(\exists m \; f_\theta(\tilde{\mathcal{I}}_{a_m{}'}) = \text{open}\Big) \wedge \Big(\neg\exists m \; f_\theta(\tilde{\mathcal{I}}_{a_m{}'}) = \text{deconstruction}\Big). \end{cases} \tag{6.1}$$

6.2.1.4 Variable Frame Rate Data Augmentation



Click or scan

Figure 6.5: The variable frame rate data augmentation method samples frames from different actions at different frame rates.

With the uniform sampling approach for video classification models described in Secs. 6.2.1.1 and 6.2.1.2, longer running actions dominate the sampled frames. While this ensures a good coverage of the entire task, there is a lower representation of the shorter running actions, which may provide valuable context. In order to balance the requirement for sufficient coverage and context, we propose a train and test-time augmentation method inspired by the SlowFast model [171], and similar to the time-series augmentation method proposed by Guennec et al. [178]. We sub-sample and concatenate equidistant frames from each action separately $\tilde{\mathcal{I}}^{sep}_{A'} = \{\tilde{\mathcal{I}}_{a_{m'}} \sim \text{Subsample}(\mathcal{I}_{a_{m'}}, s_m)\}$, where s_m, the number of frames sampled from each action, can vary. More concretely, for a given sample, we randomly select an action and sample a majority of the 32 frames from that action, and the remaining frames from the other actions. This results in a clip where the selected action has a high frame-rate and the remaining actions have a low frame-rate, which is visualized in Fig. 6.5. At inference time, we evaluate by *a)* only using the uniform sub-sampling method (i.e. $\tilde{\mathcal{I}}_{A'} \sim \text{Subsample}(\mathcal{I}_{A'}, 32)$), and *b)* applying the augmentation method for each action individually, and computing the final result based on the mean logits from each augmentation and the uniformly sampled frames. The test-time augmentation ensures that every action, regardless of its duration, contributes to the final prediction, with each sample focussing on one action, but including frames from the other actions as well. We also experiment with sampling at variable frame rates in a non-action-aligned manner, namely by selecting a random position in the video and sampling frames at a higher rate in its neighbourhood, and slower elsewhere. Unless otherwise specified, we use action-aligned frame sampling. This method also bears some similarity to the pre-processing of inputs for the MSTCN-A model as described in Section 5.2.2.2. In that case, the input feature sequences are resampled based on the robot's action boundaries both at training and inference time such that the feature sequence lengths for all actions are equal.

6.2.1.5 Region of Interest Cropping

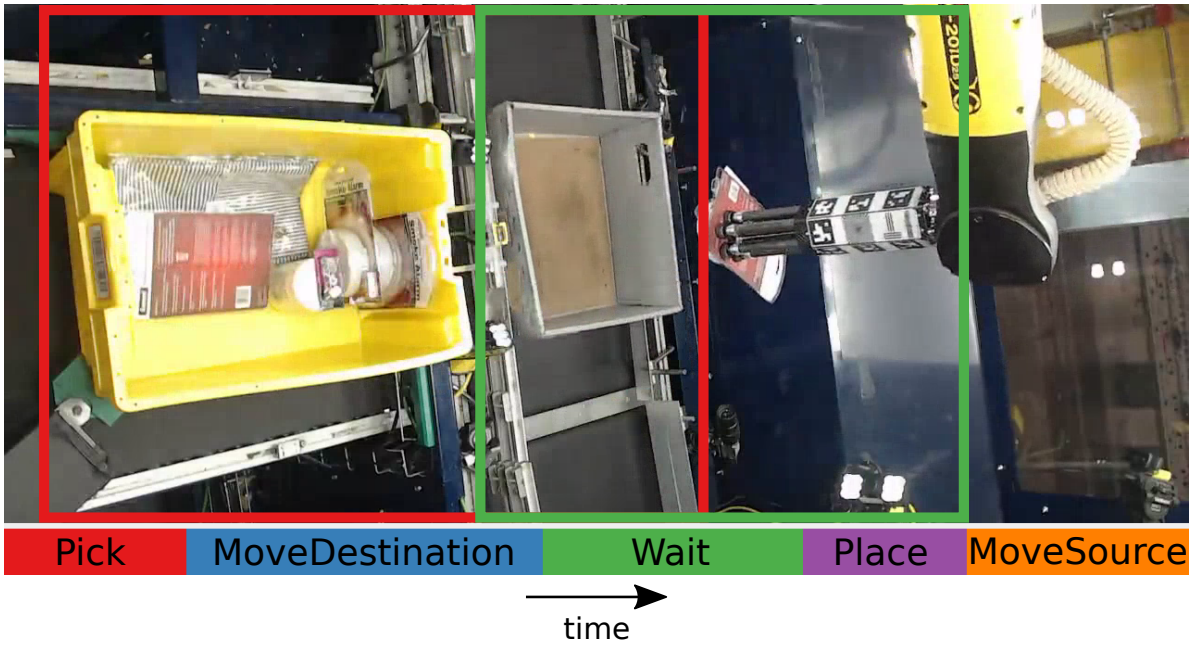


Figure 6.6: The pick-and-place task in the ARMBench dataset is segmented into the actions shown at the bottom. The current frame shows the `Wait` action. The red box on the left indicates the cropped region during the `Pick` and `MoveDestination` actions, and the green box indicates the cropped region for the remaining actions.

The robot performs its activities in two distinct regions of the image, namely picking up from the source container on the left, and waiting and placing the object in the destination container on the right. Depending on the action being performed, different regions of the image are relevant for failure classification. Therefore, we crop the sub-sampled frames based on the action they belong to, by making use of the detected bounding boxes of the containers and end-effector. As illustrated in Figs. 6.6 and 6.7, for `Pick` and `MoveDestination`, we crop the region formed by the union of the bounding boxes (only along the horizontal axis) of the source container and end-effector across all frames within that action, and similarly for `Wait`, `Place` and `MoveSource`, we crop the region formed by the union of the bounding boxes of the destination container and end-effector across all frames within those actions. Thus, the final cropped and resized (to 224×224) frames input to the model are more focussed on the end-effector and object compared to a fixed cropped that includes both regions.

6.2.2 Training and Metrics

The MViT-B model is initialized with weights trained on the Kinetics-600 dataset [48]. Since the ARMBench dataset does not have a validation set, we train all models for a fixed number of epochs. We first train a baseline model for 5 epochs, and use that to initialize all subsequent

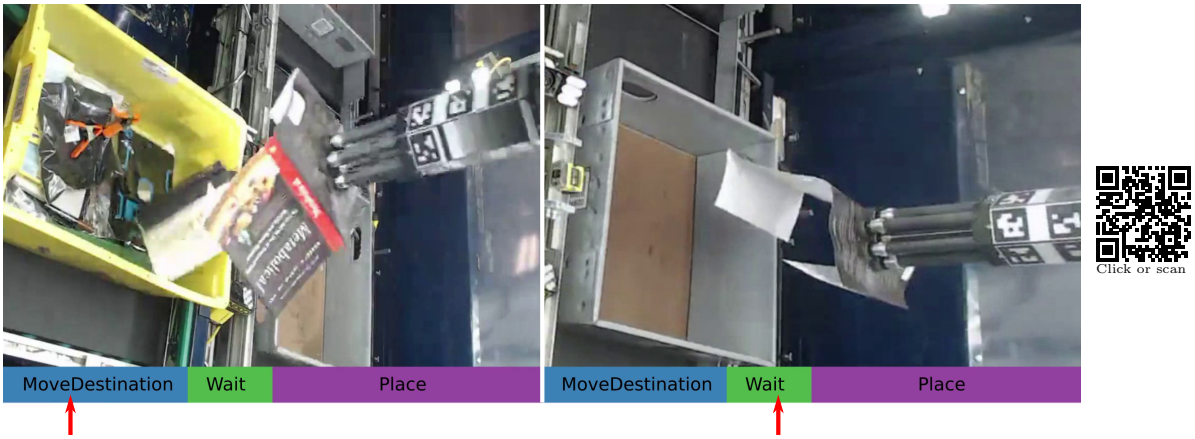


Figure 6.7: The frames are cropped based on the ROI for the current action being performed.

models, which are trained for a further 10 epochs. At training time, we introduce randomness in the selected frames by varying the starting point for uniform sampling. All models are trained 3 times, and the mean results are reported. As described in Section 2.7, we report the F_1-score rather than accuracy due to the large number of nominal samples in the test split as defined by [3]. We also report the recall and false positive rate (FPR) for the individual failure classes.

6.3 Results

The MViT-B model is trained using different sequences of frames and with the ROI cropping strategy described in Section 6.2. The corresponding results, and results from two additional baseline models are presented in Table 6.2. The baseline (1) from [3] uses the MViT-B model with uniform sampling and a fixed cropping strategy. However, the results are based on incorrect labels, and thus cannot be directly compared to our results. The result in (4) corresponds to the same training configuration as (1) with corrected labels. In (2-3), the models are trained using frames sampled from the neighbourhood of the timestamp when the failure are first visible and correspond to a loose upper-bound on the performance assuming an ideal frame selection strategy, as described in Section 6.2.1.

We find that action-based ROI cropping improves performance in all variants (8-17) except the baseline (4-7), particularly improving the recall for open failures. Limiting the frame selection to a subset of the actions (8-11) improves performance compared to the baseline as well. Using frame rate data augmentation during training (12, 14) improves performance slightly and, as predicted, test-time augmentation (13, 15) further improves performance, although this comes at the expense of additional computation at test time, since multiple forward passes are needed. We see that classifying individual actions (16, 17) lowers performance considerably; this is likely because the individual actions do not capture sufficient context necessary to determine what type of failure has occurred. The high FPR for open in the *Single action* case suggests that a wider coverage of the task is necessary to reliably conclude that a failure

Table 6.2: ARMBench Defect Classification Results

ID	Frames (Model)	Aug.	Crop	F_1	Deconstruction		Open	
					Recall	FPR	Recall	FPR
1	Baseline [3]*	-	-	-	79.0	3.0	69.0	23.0
2	Upper bound	-		80.0	87.8	1.0	71.1	1.1
3	(MViT-B)	-	✓	82.1	88.4	0.8	73.2	1.0
4		-		77.9	84.8	1.1	67.9	1.1
5	Baseline	test		78.6	81.9	0.6	58.6	0.4
6	(MViT-B)	-	✓	77.4	83.6	1.0	69.4	1.3
7		test	✓	79.1	80.2	0.6	66.0	0.8
8		-		78.7	85.1	1.0	67.7	0.9
9	Action subset	test		79.6	83.9	0.8	62.9	0.5
10	(MViT-B)	-	✓	79.6	87.3	1.1	71.2	1.1
11		test	✓	81.0	86.3	0.8	68.1	0.7
12		train		78.9	85.8	1.0	68.6	1.0
13	Action subset	train + test		80.5	86.3	0.9	66.6	0.6
14	(MViT-B)	train	✓	80.0	87.0	1.0	69.3	1.0
15		train + test	✓	**81.4**	87.3	0.9	67.9	0.6
16	Single action	-		70.8	74.6	0.8	77.9	3.5
17	(MViT-B)	-	✓	71.6	73.7	0.6	81.0	3.8
18	Eight frames	-		52.3	70.1	1.9	30.8	3.5
19	(VGG-RGB [9])	-	✓	52.2	70.5	2.3	31.5	3.1
20	All (MSTCN-B)	-	-	55.4	72.7	1.0	18.3	1.0

*using MViT-B as reported in [3], with incorrect labels

Table 6.3: Action-aligned vs random augmentation on ARMBench

Frames	Augmentation	Crop	F_1	Deconstruction		Open	
				Recall	FPR	Recall	FPR
Baseline	action-aligned		78.2	85.4	1.0	69.2	1.2
	random		78.2	85.2	1.1	68.3	1.1
	action-aligned	✓	77.9	84.5	1.1	69.2	1.1
	random	✓	77.5	85.2	1.3	69.6	1.2
Action subset	action-aligned		78.9	85.8	1.0	68.6	1.0
	random		78.9	86.2	1.1	68.6	1.0
	action-aligned	✓	80.0	87.0	1.0	69.3	1.0
	random	✓	79.2	86.5	1.1	70.4	1.1

is deconstruction rather than open. The VGG-RGB model (18, 19), which only uses 8 frames as the input, also performs poorly as expected since the input does not capture the full execution adequately. In particular, we note a low recall for open. The joint action segmentation and outcome classification approach using the MS-TCN model (20) has a similar performance, although it has access to features from *all* frames in the video. Although the input covers the full execution of the task, the most likely reason for the poor performance is that the extracted features are not discriminative enough, particularly for the open class. The large domain shift between the ARMBench dataset and the Kinetics dataset (on which the I3D-based feature extractor is trained) is likely the reason for the poor generalization of the features. In Table 6.3, we compare action-aligned augmentation versus augmenting random parts of the video on ARMBench during training, and we find that there is no significant difference.

Table 6.4: F_1 scores for the FAILURE and (Im)PerfectPour datasets

Model	**Augmentation**	**FAILURE**	**(Im)PerfectPour**
FINO-Net-RGB [9]	-	85.8†	74.0*
VGG-RGB [9]	-	**90.1**†	-
ConditionNET [42]	-	88.0*	**97.0***
MViT-B	-	87.0	96.9
MViT-B	train	87.7	**97.1**
MViT-B	train + test	88.6	**97.3**

as reported in * [42] and † [9]

6.3.1 FAILURE and (Im)PerfectPour Datasets

In both the FAILURE [9] and (Im)PerfectPour [42] datasets (see Section 3.3), the tasks are comprised of three actions, namely $A = \{$Approach, Act, Retract$\}$ in the case of FAILURE and $A = \{$Pre, Core, Post$\}$, in the case of (Im)PerfectPour. The Act and Core actions refer to the actual task being performed, such as pouring, placing etc. For the FAILURE dataset, we annotate the temporal boundaries of the three actions, and for (Im)PerfectPour, the temporal boundaries for the actions are already annotated. Thus for these two datasets, we use all actions of the dataset and only experiment with performing the action-aligned variable frame rate augmentation. The MViT-B models are trained 3 times, and the mean results are reported in Table 6.4. For both datasets, we see minor improvements in the performance with the augmentation, and the performance of the MViT-B model is comparable to the VGG-RGB and ConditionNET models.

6.3.2 Handover Failure Detection Dataset

As described in Section 5.2.1.3, the actions performed by the robot in the HFD dataset are $A = \{$Idle, Approach, Interact, Retract, Post-Idle$\}$. For the action subset variant listed in Table 6.5, we sample frames from $A' = \{$Approach, Interact, Retract$\}$. We use the I3D models described in Section 5.2.2.1, and only alter the frame sampling strategy. For I3D-A, we use

Table 6.5: Failure classification accuracy for the HFD dataset

		Accuracy	
Input	**Augmentation**	**I3D-A (V)**	**I3D-D (V/F-T/G)**
Full video	-	64.8	67.9
Full video	train	69.9	68.5
Full video	train + test	**70.9**	69.3
Action subset	-	68.5	71.4
Action subset	train	68.6	72.5
Action subset	train + test	68.9	**75.6**

V: Video, F-T: force-torque, G: gripper

the video-only model (with late-fusion of the RGB and optical flow networks), and for I3D-D, we use all modalities (namely, RGB, optical flow, force-torque sensor, and gripper). For consistency with the results in Section 5.2.3, we train all models 5 times and report the mean accuracy in Table 6.5. We find that selecting frames from a subset of the actions and the augmentation strategy improves performance in all cases, with the multimodal model operating on A' achieving an accuracy of 75.6%.

6.3.3 Visual-Tactile Dataset

As described in Section 5.3.1.1, the actions performed by the robot in the VT dataset are $A = \{$`pre-grasp`, `grasp`, `lift`, `release`$\}$. For the action subset variant listed in Table 6.6, we sample frames from $A' = \{$`grasp`, `lift`, `release`$\}$.

We use the RGB-only and the multimodal (RGB, tactile image and joint position image) I3D-B models as described in Section 5.3.2.1 for failure detection, and only alter the frame sampling strategy. All models are trained 5 times and we report the mean precision, recall and F_1 score in Table 6.6. In general, the performance improvements are similar to those seen in the other datasets, however, since the baseline model already has a high F_1 score, all improvements are minor. Contrary to the previous datasets, we see that for the unimodal model using the full video (1-3), there is, in fact, a drop in performance when applying the frame rate augmentation approach at training time. This leads us to a limitation of the approach, namely, that it can also emphasize irrelevant actions. In the case of the VT dataset, the `pre-grasp` action occurs before any interaction with the object, and provides no relevant information for failure detection; therefore, including training samples in which the `pre-grasp` action is sampled at a high-frame rate potentially leads to a drop in performance. In the multimodal case (7, 8), we do not see a similar drop in performance, presumably because the features from the other two modalities are strong enough to compensate. In fact, using the action subset for the multimodal model (10-12) leads to a slight drop in performance, suggesting that using a subset of the actions may negatively affect the representation of the tactile and joint position data.

Table 6.6: Failure detection performance on the VT dataset

ID	Input	V	T	P	Augmentation	Precision	Recall	F_1
1	Full video	✓			-	96.9	96.5	96.5
2	Full video	✓			train	93.3	90.3	88.7
3	Full video	✓			train+test	94.7	92.7	92.0
4	Action subset	✓			-	99.0	99.0	99.0
5	Action subset	✓			train	99.5	99.5	99.5
6	Action subset	✓			train+test	99.6	99.6	**99.6**
7	Full video	✓	✓	✓	-	99.1	99.1	99.0
8	Full video	✓	✓	✓	train	99.7	99.7	99.7
9	Full video	✓	✓	✓	train+test	99.9	99.9	**99.9**
10	Action subset	✓	✓	✓	-	99.0	99.0	99.0
11	Action subset	✓	✓	✓	train	99.4	99.4	99.4
12	Action subset	✓	✓	✓	train+test	99.2	99.2	99.2

V: Video, T: Tactile, P: Joint Position

6.3.4 Discussion

Overall, we find that focusing on certain actions and regions based on task knowledge improves performance without additional computational expense, but a sufficient coverage of the task execution is necessary to reliably detect failures. The variable frame rate-based data augmentation approach provides a minor improvement as well, with test-time augmentation improving performance further, particularly by reducing the false positive rate in the ARMBench dataset. We see consistent improvements across five datasets, although a limitation of the augmentation method is observed in the VT dataset, namely that applying it to the full video, which may include irrelevant actions, can hurt the performance. For the ARMBench dataset, the remaining false negatives include failures that occur in a short time window, and subtle `open` failures which require a close inspection of the object, suggesting that even denser sampling of frames and closer crops of the object may be required.

6.4 Summary

In this chapter, we highlighted the importance of using readily available task information for improving the performance of video-based failure detection and classification models. The robot's actions and task-relevant objects are used to select and pre-process frames used by the classification model, leading to improved performance without additional computational expense. We focused primarily on the ARMBench dataset, which is the largest video-based real-world robot failure dataset available currently. In particular, we enhanced the dataset by fixing incorrect labels, adding annotations for the temporal boundaries of the robot's actions, labelling bounding boxes for relevant objects in the scene and annotating the timestamp when

the failures are first visible. Based on the additional annotations, we experimented with various frame selection and pre-processing methods, which showed that using the temporal boundaries of the robot's actions to guide frame selection, and using object bounding boxes to crop frames based on the action being performed, both led to improved performance of a failure classification model. The proposed data augmentation approach, which varies the frame rate of sampled frames, leads to additional improvements through train and test-time augmentation as well. We showed that the frame selection and augmentation strategies also leads to improvements in the FAILURE, (Im)PerfectPour, HFD and VT datasets. A limitation of the approach is that test-time augmentation requires multiple forward passes, and hence is more computationally expensive. Additionally, the augmentation approach could emphasize irrelevant actions, leading to a drop in performance.

CHAPTER 7
Conclusions

In this chapter, we summarize the contributions of this thesis and discuss future directions for research in the area of visual failure detection in robotics.

7.1 Summary

In Chapter 1, we identified the key opportunities and challenges associated with vision-based failure detection in robotics that we focus on in this thesis, namely, the development of datasets, and making use of multimodal data and task knowledge to improve failure detection performance. We summarize our contributions in these three areas.

Datasets With the increased performance of deep learning for various vision tasks, data-driven approaches have similarly improved the performance of vision-based failure detection. We contribute to the development of datasets with the Bookshelf dataset in Chapter 4 and the Handover Failure Detection dataset in Chapter 5. Both datasets are task-specific and contain multimodal data. The Bookshelf dataset contains 121 trials of a robot placing a book on a shelf and is intended for anomaly detection and localization, with anomalies such as falling objects and camera occlusions present in the test set. The HFD dataset contains 589 trials and is designed for multimodal failure classification in robot-human handovers, where the failures are induced by the human, such as not releasing the object and dropping objects. We also enhance existing datasets with additional annotations and corrections. We align the visual, tactile and proprioceptive sensor data in the VT dataset through additional annotations in Chapter 5 to facilitate multimodal learning, and fix incorrect labels and annotate the temporal boundaries of the robot's actions and locations of task-relevant objects in the ARMBench dataset to make use of the additional task knowledge for improved failure classification in Chapter 6.

Multimodal Learning In Chapters 4 and 5, we make use of multimodal data for various tasks. In Chapter 4, in addition to a future optical flow prediction network for anomaly detection, the proprioceptive sensor data of the robot is used to model the expected motions in the scene that are related to the robot's own motion. These expectations are compared

to observed motions to detect anomalous motions. We find that masking motions of the robot's body in the optical flow input to the future flow prediction network leads to improved anomaly localization, and using the discrepancies in expected and observed apparent scene motion to due camera motion also leads to improved anomaly localization performance. In Chapter 5, we use video, force-torque sensor and gripper position data from the HFD dataset for failure classification and find that feature-level (intermediate) fusion of all three modalities leads to the best performance, though the combination of video and force-torque sensor data was particularly effective. For the VT dataset, fusing video, tactile and joint position data also resulted in improved failure detection performance over individual modalities, with a fusion of video and position data performing the best when the position data were represented as 1D vectors. For failure localization, video data was found to be less important, since using tactile or joint position data alone already achieved a similar performance as the multimodal variants.

Task Knowledge Task knowledge, which encompasses any task-specific knowledge available to the robot at the time of execution, has been used in Chapters 5 and 6 to improve failure classification performance. In Chapter 5, the proposed temporal action segmentation approach shows an improved performance when it is trained to jointly classify failures and to temporally segment the human's actions. In the unimodal (video-only) case, learning to temporally segment the robot's actions leads to an additional improvement. The human's actions represent *learned* task knowledge, i.e., the model learns to anticipate the behaviour of the human during a handover task from the training data, which in turn improves its failure classification performance. The robot's actions are task knowledge available "for free", since the robot already knows when it starts and ends its own actions. Similarly, in Chapter 6, the temporal boundaries of the robot's actions are used to guide the selection of frames for a video classification model. Firstly, frames are selected only from a subset of actions considered to be relevant for failure classification (determined by a human expert), and the variable frame rate-based data augmentation approach uses the temporal boundaries of the actions to sample frames at different rates from different actions. The locations of task-relevant objects and the temporal boundaries of the actions are also used to crop the frames to different regions of interest based on the action being performed. All three strategies result in improved performance on the ARMBench dataset, and the data augmentation method is shown to improve performance on four additional datasets.

7.2 Future Work

The failure detection performance in the literature and in our work indicates potential for further improvement. For robots to be deployed reliably in real-world settings, a true positive rate $> 95\%$ and false positive rate $< 1\%$ are to be expected [3]. *Online* failure detection would also improve the utility of the models, since detecting failures as soon as they occur would allow the robot to potentially repair the failures. For methods that need to sub-sample videos, there are further opportunities to refine the selection of frames based on multimodal data. For example, one could consider using cues from contact-related sensors (such as tactile and force-torque sensors) to increase sampling rates when the robot is interacting with the environment.

The intelligent frame sampling strategies could also improve the performance of the general-purpose VLMs and VLAs, particularly since they are limited in the number of frames they can process at each step. Currently, most VLMs perform uniform sampling of frames, although they do use other strategies to reduce the number of visual tokens.

A limitation of existing approaches and the methods presented in this thesis is the task-specific nature of the failure and anomaly detectors. The models are trained on datasets that cover a single task, and typically consider a limited set of failure types. While the anomaly detection approach, in principle, removes the limitation on the types of failures, the current approaches still train models on individual tasks. Ideally, a general-purpose model would be able to detect failures in any task by incorporating all available task knowledge and by utilizing multimodal data. Incorporating vision-based learning in symbolic learning approaches such as Hegemann et al. [18] would enable more effective failure detection that makes use of task knowledge, while also improving the interpretability of the model's outputs. Additionally, the development of general-purpose LLMs, VLMs and VLAs and large-scale robot learning datasets such as Open X-Embodiment [184] and DROID [185] present opportunities to further develop task-agnostic failure detection approaches. Indications that VLAs trained on large-scale data are able to detect and reason about failures can already be seen in models such as Gemini Robotics 1.5 [120]. If large-scale datasets were to explicitly include failure data, this would in turn improve the failure detection performance of the general-purpose models. Although the models make use of multimodal data (for example, Gemini Robotics 1.5 makes use of images, proprioception and natural language), only a minority make use of force-torque [186] sensor data and, to the best of our knowledge, none use tactile sensors, which are essential for contact-based tasks.

Producing failure data for training and evaluation is challenging. Capturing naturally occurring failures lead to imbalanced datasets, in which infrequent failures are represented poorly or not at all. On the other hand, inducing failures is inherently biased, since it relies on a human to decide on the types of failures. Simulation-based data generation is a possible solution for video-based failure detection, as in the RoboFail benchmark dataset [25], using simulators and frameworks such as Isaac Sim [187], DISCOVERSE [188] and RoboTHOR [189].

APPENDIX A
Dataset Datasheets

Datasheet for the Bookshelf Dataset

I INTRODUCTION

This datasheet accompanies the Bookshelf Dataset [1][1] and is based on the Datasheet for Datasets paper by Gebru et al. [190].

II MOTIVATION

A For what purpose was the dataset created?

The dataset was created to improve and evaluate multimodal anomaly detection methods in robotics that can make use of video, proprioception and force-torque sensor data.

B Who created the dataset?

The dataset was created by Santosh Thoduka.

C Who funded the creation of the dataset?

The creation of the dataset has been supported by the Bonn-Aachen International Center for Information Technology[2] and a PhD scholarship from the Graduate Institute at Hochschule Bonn-Rhein-Sieg[3].

III COMPOSITION

A What do the instances that comprise the dataset represent?

Each instance consists of a trial of a robot placing a book on a shelf. The trial may result in a successful or failed placement and may contain various anomalies such as falling books, camera occlusions and disturbances to the robot.

B How many instances are there in total?

The dataset consists of 121 trials. The total number of video frames across all trials is 16,558. The samples are split into training, validation and test sets, and Table A.1 lists the number of samples in each set.

C Does the dataset contain all possible instances or is it a sample of instances from a larger set?

The dataset contains all instances that were recorded, with the exception of incorrect or incomplete trials which were discarded during the data collection phase.

D What data does each instance consist of?

The raw data for each instance consists of:

- RGB video from the head-mounted camera (an Asux Xtion Pro)
- Depth video from the head-mounted camera

In addition, the following generated files or annotations are also included:

- label of the outcome of the trial (success, or type of anomaly) and the ranges of frames when the anomalies occur, if any
- resampled wrist-mounted force-torque sensor data on the robot that are aligned with timestamp of the RGB frames (the raw data is not included)
- resampled robot joint positions, velocities and efforts that are aligned with the timestamps of the RGB frames (the raw data is not included)
- rendered images of the robot's 3D model from the perspective of the head-mounted camera as the robot performs the task

A detailed list of the files in a trial can be found in Table A.2.

E Is there a label or target associated with each instance?

Each instance is labeled with the outcome of the trial. All trials in the training and validation sets have the outcome `Success`, whereas the instances with anomalies in the test set have various natural language descriptions of outcomes such as `Object falls to the ground`, `Object does not reach intended position` and `Robot is disturbed`. For instances with anomalies, the ranges of frames during which the anomalies are present are labelled.

F Is any information missing from individual instances?

No.

G Are relationships between individual instances made explicit?

There is no relationship between individual instances.

H Are there recommended data splits?

The dataset is split into a training set, validation set and test set. The training and validation sets only contain nominal trials, and the test set contains both anomalous and nominal trials.

I Are there any errors, sources of noise, or redundancies in the dataset

The annotations for the start and end times of the anomalies in the videos may contain ambiguities, since they were annotated manually by a single annotator.

J Is the dataset self-contained, or does it link to or otherwise rely on external resources?

It is self-contained.

K Does the dataset contain data that might be considered confidential?

No.

L Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety?

No.

M Does the dataset identify any sub-populations?

No.

[1] `https://zenodo.org/records/4578539`

[2] `https://www.b-it-center.de/`

[3] `https://www.h-brs.de/en/graduierteninstitut`

Table A.1: Training, validation and test sets for the Bookshelf dataset

Dataset split	Nominal trials	Anomalous trials	Nominal frames	Anomalous frames
Training	48	0	6,732	0
Validation	6	0	749	0
Test	7	60	7,969	1,108
Total	61	60	15,450	1,108

Table A.2: Description of files in each trial

File	Description
`rgb`	Folder containing RGB frames of the full trial of the robot (length = M frames)
`rendered_body`	Folder containing rendered images of the robot's 3D model from the perspective of the head-mounted camera (length = M frames)
`joint_names.npy`	names of joints of the robot (dim = K)
`joint_state_efforts.npy`	resampled joint efforts which correspond to the timestamps of the frames in `rgb` for all joints listed in `joint_names.npy` (dim = $M \times K$)
`joint_state_positions.npy`	resampled joint positions which correspond to the timestamps of the frames in `rgb` for all joints listed in `joint_names.npy` (dim = $M \times K$)
`joint_state_velocities.npy`	resampled joint velocities which correspond to the timestamps of the frames in `rgb` for all joints listed in `joint_names.npy` (dim = $M \times K$)
`camera_matrix.npy`	resampled homogeneous transformation of the head-mounted camera positions with respect to the robot's base, corresponding to the timestamps of the frames in `rgb` (dim = $M \times 4 \times 4$)
`wrench.npy`	Resampled wrench (force X, force Y, force Z, torque X, torque Y, torque Z) measured at the wrist, corresponding to the timestamps of the frames in `rgb` (dim = $M \times 6$)
`annotation_info.json`	dictionary with outcome, and optionally, the ranges of frames when anomalies are present

N Is it possible to identify individuals, either directly or indirectly from the dataset?

No.

O Does the dataset contain data that might be considered sensitive in any way?

No.

IV COLLECTION PROCESS

A How was the data associated with each instance acquired? Was the data directly observable, reported by subjects, or indirectly inferred/derived from other data?

The raw data was recorded in the form of Robot Operating System (ROS) bag files while the robot was commanded to execute the task of placing the book on the shelf. The video, depth, and sensor data were subsequently extracted from the bag files.

B What mechanisms or procedures were used to collect the data?

All data were recorded on the Toyota HSR robot from its on-board sensors.

Robot execution and data recording were conducted as follows:

1. the robot was placed in front of a shelf and a book was placed in the robot's gripper
2. data recording was started
3. the robot executed a pre-defined trajectory, based on a target pose, to approach the shelf, release the book and retract
4. for certain anomalies, such as camera occlusions and external disturbances to the robot, the anomalies were introduced while the robot executed the trajectory
5. data recording was stopped

An earlier version of the `metrics_refbox`[4] tool was used for coordinating execution and recording.

C Over what timeframe was the data collected?

The data was collected during three sessions between May 2020 and January 2021.

D Were any ethical review processes conducted?

No.

[4] `https://github.com/HEART-MET/metrics_refbox`

V PREPROCESSING/CLEANING/LABELING

A Was any preprocessing/cleaning/labeling of the data done?

Yes. The final outcome of the trial was labeled during the data collection process.

After data collection, the following were also performed:

1. video, depth and and sensor data (force-torque and joint data) were extracted from the ROS bag files
2. the raw sensor data were resampled to match the frequency of the video stream
3. images of the robot's 3D model were rendered from the perspective of the head-mounted camera by setting the simulated robot's joint configuration based on the recorded joint states for every time step using pyrender[5]

B Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data?

Yes. However, the raw data in the form of ROS bag files are not part of the dataset.

C Is the software that was used to preprocess/clean/label the data available?

No.

VI USES

A Has the dataset been used for any tasks already?

The dataset has been used for visual anomaly detection in the work published at the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) in the paper titled "Using Visual Anomaly Detection for Task Execution Monitoring" [1].

B Is there a repository that links to any or all papers or systems that use the dataset?

Yes. The webpage `https://sthoduka.github.io/motion_anomaly_detection` corresponds to the paper mentioned above.

VII DISTRIBUTION

A Will the dataset be distributed to third parties outside of the entity on behalf of which the dataset was created?

Yes.

B How will the dataset be distributed?

The dataset has been distributed on Zenodo[6].

C When will the dataset be distributed?

The dataset has been available on Zenodo since March 2021.

D Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)?

Yes. The dataset is distributed under the Creative Commons Attribution license 4.0.

E Have any third parties imposed IP-based or other restrictions on the data associated with the instances?

No.

F Do any export controls or other regulatory restrictions apply to the dataset or to individual instances?

No.

VIII MAINTENANCE

A Who will be supporting/hosting/maintaining the dataset?

The dataset will be maintained by Santosh Thoduka.

B How can the owner/curator/manager of the dataset be contacted?

The owner can be contacted by creating an issue on Github: `https://github.com/sthoduka/motion_anomaly_detection/issues`

C Is there an erratum?

No.

D Will the dataset be updated?

Yes. Any updates (such as fixing labeling or data errors, new instances, deleting instances) will be uploaded to Zenodo as a new version of the dataset.

E If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances?

No.

F Will older versions of the dataset continue to be supported/hosted/maintained?

Yes.

G If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so?

There is no mechanism currently, but interested persons can contact the maintainer for more details.

[5] `https://github.com/mmatl/pyrender`

[6] `https://zenodo.org/records/4578539`

Datasheet for the Handover Failure Detection Dataset

I INTRODUCTION

This datasheet accompanies the Handover Failure Detection Dataset [2][7] and is based on the Datasheet for Datasets paper by Gebru et al. [190].

II MOTIVATION

A *For what purpose was the dataset created?*

The motivation for creating the dataset is to improve the monitoring capabilities of robots during human-to-robot (H2R) and robot-to-human (R2H) handovers, thus enabling them to react to failure conditions appropriately. Current research focuses on preventing handover failures, whereas this dataset focuses on the detection of unpreventable failures which may be caused by the human participant.

B *Who created the dataset?*

The dataset was created at Hochschule Bonn-Rhein-Sieg, Germany, in the context of the HEART-MET competition for healthcare robots[8] within the METRICS project.

C *Who funded the creation of the dataset?*

The dataset was created during the METRICS project, which was funded by the European Union Horizon 2020 research and innovation program under grant agreement No. 871252.

III COMPOSITION

A *What do the instances that comprise the dataset represent?*

Each instance consists of one trial of a robot-to-human handover or a human-to-robot handover of an object. The trial may consist of a successful handover or a failed handover.

B *How many instances are there in total?*

Table A.3 summarizes the total instances in the dataset. Two robots were used to collect the dataset, namely, the Toyota Human Support Robot (HSR) and the Kinova Gen3 arm. The columns indicate the different outcomes of the trials, which relate to the action of the human participant (for example, `no approach` means the person did not approach the robot).

A total of 17 participants are part of the dataset. A total of 22 object classes are used for the handovers; these are tomato paste tube, cardboard box (e.g. for crackers), kitchen sponge, banana, bottle, spectacles, plate, torch, bowl, toothbrush, book, towel, bottle of eye drops, pill box, cup, cup noodles, ball, double-sided tape, glue stick, jug, pringles can and toothpaste.

[7] `https://zenodo.org/records/10708763`
[8] `https://metricsproject.eu/healthcare/`

Table A.3: Dataset statistics (reproduced from [2])

	R2H					H2R				
	success	no approach	no grasp	drop	**total**	success	no approach	no release	drop	**total**
Toyota HSR	68	50	49	58	225	51	46	57	66	220
Kinova Gen3	18	17	17	20	72	19	18	17	18	72
Total	**86**	**67**	**68**	**76**	**297**	**70**	**64**	**74**	**84**	**292**

C *Does the dataset contain all possible instances or is it a sample of instances from a larger set?*

The dataset contains all instances that were recorded, with the exception of incorrect or incomplete trials which were discarded during the data collection phase.

D *What data does each instance consist of?*

The raw data for each instance consists of:

- RGB video from the point of view of the robot
- wrist-mounted force-torque sensor on the robot (Toyota HSR), or simulated force-torque at the wrist based on sensed robot joint torques (Kinova Gen3)
- robot joint positions, velocities and efforts

In addition, the following generated files or annotations are also included:

- resampled force-torque and joint data to match the frequency of the video stream
- optical flow images generated from the RGB videos
- I3D [48] features for both RGB and optical flow
- labeling of the start and end of the robot's actions (approach, interact, retract) based on the joint data
- labeling of the start and end of the human's actions, and incorrect behaviours (idle, approach, interact, retract, post-idle, not released, dropped)
- metadata regarding the robot used, and the task (R2H or H2R) performed in that instance

A detailed list of the files in a trial can be found in Table A.4, and descriptions of class IDs can be found in Table A.5.

E *Is there a label or target associated with each instance?*

Each instance is labeled with the outcome of the trial (as listed in Table A.3). In addition, as mentioned previously, the start and end times of the robot's and human's actions are also labeled.

Table A.4: Description of files in each trial

File	Description
head_cam.mp4	RGB video (length = M frames)
head_cam_ts.npy	Numpy file with timestamps associated with each frame of the RGB video (dim = M)
flow	Folder containing optical flow images. Each pair (frameAAAA_x.jpg and frameAAAA_y.jpg) correspond to the horizontal and vertical components of the flow respectively. frame0001_x.jpg refers to the horizontal optical flow from the first to the second frame of the RGB video. Thus, in each trial, there is one less pair of optical flow frames compared to the frames in the RGB video. (i.e. $2(M-1)$ images)
i3d_kinetics_flow.npy	I3D features corresponding to the optical flow images (dim = M x 1024)
i3d_kinetics_rgb.npy	I3D features corresponding to the RGB frames (dim = M x 1024)
i3d_kinetics_rgb_augmented.npy	additional I3D features available only in the training set corresponding to augmented RGB frames (dim = M x 5 x 1024)
human_activity.npy	Numpy file containing the human action performed for each frame of the RGB video; class IDs are described in Table A.5 (dim = M)
robot_actions.npy	Numpy file containing the robot action performed for each frame of the RGB video; class IDs are described in Table A.5 (dim = M)
joint_names.npy	names of joints of the robot (dim = K)
joint_state_ts.npy	Numpy file with timestamps associated with each joint state reading (joint positions, velocities and efforts) (dim = N, where $N > M$)
joint_efforts.npy	efforts of joints listed in joint_names.npy (dim = N x K)
joint_positions.npy	positions of joints listed in joint_names.npy (dim = N x K)
joint_velocities.npy	velocities of joints listed in joint_names.npy (dim = N x K)
joint_pos_resampled.npy	resampled joint positions which correspond to timestamps in head_cam_ts.npy (dim = M x K)
wrench_ts.npy	Numpy file with timestamps associated with each wrench (force-torque) reading (dim = P, where $P > M$)
wrench.npy	Wrench (force X, force Y, force Z, torque X, torque Y, torque Z) measured at the wrist (dim = P x 6)
wrench_resampled.npy	resampled wrench which corresponds to timestamps in head_cam_ts.npy (dim = M x 6)
task_info.json	dictionary with task type (R2H or H2R) and robot type (Toyota HSR or Kinova Gen3) for the trial
trialBBBB.json	stored in the labels folder, this file has a dictionary containing the task type, robot type and overall outcome for the trial; class IDs are described in Table A.5

Table A.5: Class IDs and descriptions

Class ID	R2H outcomes	H2R outcomes	human actions	robot actions
0	success	success	idle	idle
1	no approach	no approach	approach	approach
2	no grasp	no release	interact	interact
3	drop	drop	retract	retract
4	-	-	post-idle	post-idle
5	-	-	not released	-
6	-	-	dropped	-

F Is any information missing from individual instances?

No.

G Are relationships between individual instances made explicit?

Each instance includes metadata about the robot and task (H2R or R2H); thus instances can be related by the robot or task.

H Are there recommended data splits?

The dataset is split into a training set, validation set and test set based on the subjects present in the video.

I Are there any errors, sources of noise, or redundancies in the dataset

The annotations for the start and end times human's actions in the video may contain ambiguities. All annotations were performed by a single annotator; nevertheless, there might be ambiguities regarding the exact temporal boundaries of the actions. For example, the interact phase of the handover is determined visually (start and end of the physical contact between the human, object and robot) and using the force-torque data. Different annotators may decide on different boundaries for this, based on their assessment of the time of contact, etc.

J Is the dataset self-contained, or does it link to or otherwise rely on external resources?

It is self-contained.

K Does the dataset contain data that might be considered confidential?

No.

L Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety?

No.

M Does the dataset identify any sub-populations?

No.

N Is it possible to identify individuals, either directly or indirectly from the dataset?

The RGB video in each instance captures a view of the subject interacting with the robot. As such, their

face is visible in the video, and could be used to identify them. No other identifying information is included in the dataset.

O Does the dataset contain data that might be considered sensitive in any way?

No.

IV COLLECTION PROCESS

A How was the data associated with each instance acquired? Was the data directly observable, reported by subjects, or indirectly inferred/derived from other data?

The raw data was recorded in the form of Robot Operating System (ROS) bag files. The video and sensor data were subsequently extracted from the bag files.

B What mechanisms or procedures were used to collect the data?

For the Toyota HSR, all data were recorded on the robot from its on-board sensors. For the Kinova Gen3, two arms were used. One arm performed the handover task, while the second arm was placed in a fixed configuration such that its arm-mounted camera could view the scene. Both arms were connected to a single laptop via Ethernet cables, where the recording took place. Thus robot joint data was recorded from the first arm, and video data from the second.

Robot execution and data recording were conducted as follows:

1. the subject is instructed about their desired behaviour (for example, they should not release the object after the robot grasps it)
2. data recording is started
3. the robot approaches the subject with its arm
4. if a threshold is reached on the force-torque sensor, the robot closes (or opens) its gripper
5. the robot retracts its arm
6. data recording is stopped

The `metrics_refbox`[9] tool was used for coordinating execution and recording.

C Who was involved in the data collection process and how were they compensated?

Robotics students and researchers were the subjects in the trials and they were not compensated.

D Over what timeframe was the data collected?

For an individual subject, the data collection process took a maximum of 3 hours. Overall, the data was collected in a total of 3 weeks in 2020 and 2022. Annotation of the human activities was performed over several weeks.

E Were any ethical review processes conducted?

No.

F Did you collect the data from the individuals in question directly, or obtain it via third parties or other sources?

The data were collected directly.

G Were the individuals in question notified about the data collection?

Yes.

H Did the individuals in question consent to the collection and use of their data?

Yes.

I If consent was obtained, were the consenting individuals provided with a mechanism to revoke their consent in the future or for certain uses?

The consent form provided the individuals with the option to revoke permission for use of their data. However, they were also informed that once the dataset was made public, it could not be guaranteed that all instances of their data would be deleted.

J Has an analysis of the potential impact of the dataset and its use on data subjects been conducted?

No.

V PREPROCESSING/CLEANING/LABELING

A Was any preprocessing/cleaning/labeling of the data done?

Yes. The final outcome of the trial was labeled during the data collection process.

After data collection, the following were also performed:

1. video and sensor data (force-torque and joint data) were extracted from the ROS bag files
2. the raw sensor data were resampled to match the frequency of the video stream (both raw and resampled data are part of the dataset)
3. optical flow images were generated from the RGB videos
4. I3D [48] features for both RGB and optical flow were extracted
5. the start and end of the robot's actions (approach, interact, retract) for each instance was extracted based on the joint data
6. the start and end of the human's actions, and incorrect behaviours (idle, approach, interact, retract, post-idle, not released, dropped) were annotated manually

B Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data?

Yes. However, the raw data in the form of ROS bag files are not part of the dataset.

C Is the software that was used to preprocess/clean/label the data available?

No.

[9] `https://github.com/HEART-MET/metrics_refbox`

VI USES

A Has the dataset been used for any tasks already?

The dataset has been used to evaluate two baselines for handover failure detection. This work has been published at the IEEE 2024 International Conference on Robotics and Automation in the paper titled "A Multimodal Handover Failure Detection Dataset and Baselines" [2].

B Is there a repository that links to any or all papers or systems that use the dataset?

Yes. The webpage[10] corresponds to the paper mentioned above.

C What (other) tasks could the dataset be used for?

The dataset could be used for other handover-related tasks, such as anticipating the person's actions, tracking their hand, etc.

VII DISTRIBUTION

A Will the dataset be distributed to third parties outside of the entity on behalf of which the dataset was created?

Yes.

B How will the dataset be distributed?

The dataset has been distributed on Zenodo[11].

C When will the dataset be distributed?

The dataset has been available on Zenodo since Feburary 2024.

D Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)?

Yes. The dataset is distributed under the Creative Commons Attribution license 4.0.

E Have any third parties imposed IP-based or other restrictions on the data associated with the instances?

No.

F Do any export controls or other regulatory restrictions apply to the dataset or to individual instances?

No.

VIII MAINTENANCE

A Who will be supporting/hosting/maintaining the dataset?

The dataset will be maintained by Santosh Thoduka.

B How can the owner/curator/manager of the dataset be contacted?

The owner can be contacted by creating an issue on Github: `https://github.com/sthoduka/handover_failure_detection/issues`

C Is there an erratum?

No.

D Will the dataset be updated?

Yes. Any updates (such as fixing labeling or data errors, new instances, deleting instances) will be uploaded to Zenodo as a new version of the dataset.

E If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances?

No.

F Will older versions of the dataset continue to be supported/hosted/maintained?

Yes.

G If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so?

There is no mechanism currently, but interested persons can contact the maintainer for more details.

[10] `https://sthoduka.github.io/handover_failure_detection`

[11] `https://zenodo.org/records/10708763`

APPENDIX B
Data and Software

We list the various datasets and software produced as part of this work. Videos referenced in the figures can also be found at https://sthoduka.github.io/robot_failure_detection/.

B.1 Datasets

Bookshelf dataset
https://zenodo.org/records/4578539

Handover Failure Detection dataset
https://zenodo.org/records/10708763

Visual-Tactile dataset - timestamp annotations
https://github.com/priteshgohil/Multimodal-Machine-Learning/tree/master/dataset

ARMBench dataset - corrected and additional annotations
https://zenodo.org/records/15873769

B.2 Software

Paper: Using Visual Anomaly Detection for Task Execution Monitoring
https://github.com/sthoduka/motion_anomaly_detection

Paper: A Multimodal Handover Failure Detection Dataset and Baselines
https://github.com/sthoduka/handover_failure_detection

Paper: Enhancing Video-Based Robot Failure Detection Using Task Knowledge
https://github.com/sthoduka/using_task_knowledge

Temporal segments annotation tool
https://github.com/sthoduka/temporal_segments_annotator

Bibliography

[1] S. Thoduka, J. Gall, and P. G. Plöger, "Using Visual Anomaly Detection for Task Execution Monitoring," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4604–4610.

[2] S. Thoduka, N. Hochgeschwender, J. Gall, and P. G. Plöger, "A Multimodal Handover Failure Detection Dataset and Baselines," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 17 013–17 019.

[3] C. Mitash, F. Wang, S. Lu, V. Terhuja, T. Garaas, F. Polido, and M. Nambi, "ARMBench: An Object-centric Benchmark Dataset for Robotic Manipulation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9132–9139.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[6] Y. A. Farha and J. Gall, "MS-TCN: Multi-Stage Temporal Convolutional Network for Action Segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3575–3584.

[7] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," in *Arxiv*, 2016. [Online]. Available: https://arxiv.org/abs/1609.03499

[8] T. Wang, C. Yang, F. Kirchner, P. Du, F. Sun, and B. Fang, "Multimodal grasp data set: A novel visual–tactile data set for robotic manipulation," *International Journal of Advanced Robotic Systems*, vol. 16, no. 1, p. 1729881418821571, 2019.

[9] A. Inceoglu, E. E. Aksoy, A. C. Ak, and S. Sariel, "FINO-Net: A Deep Multimodal Sensor Fusion Framework for Manipulation Failure Detection," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 6841–6847.

[10] M. Levenson, "Driverless Car Gets Stuck in Wet Concrete in San Francisco," *The New York Times*. [Online]. Available: https://www.nytimes.com/2023/08/17/us/driverless-car-accident-sf.html

[11] L. Tisserand, B. Stephenson, H. Baldauf-Quilliatre, M. Lefort, and F. Armetta, "Unraveling the Thread: Understanding and Addressing Sequential Failures in Human-Robot Interaction," *Frontiers in Robotics and AI*, vol. 11, p. 1359782, 2024.

[12] M. Vasic and A. Billard, "Safety Issues in Human-Robot Interactions," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 197–204.

[13] M. Salem, G. Lakatos, F. Amirabdollahian, and K. Dautenhahn, "Would You Trust a (Faulty) Robot? Effects of Error, Task Type and Personality on Human-Robot Cooperation and Trust," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, 2015, pp. 141–148.

[14] M. Diehl and K. Ramirez-Amaro, "A Causal-Based Approach to Explain, Predict and Prevent Failures in Robotic Tasks," *Robotics and Autonomous Systems*, vol. 162, p. 104376, 2023.

[15] T. Ji, A. N. Sivakumar, G. Chowdhary, and K. Driggs-Campbell, "Proactive Anomaly Detection for Robot Navigation with Multi-Sensor Fusion," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4975–4982, 2022.

[16] A. G. Eguiluz, I. Rañó, S. A. Coleman, and T. M. McGinnity, "Reliable Object Handover Through Tactile Force Sensing and Effort Control in the Shadow Robot Hand," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 372–377.

[17] N. Akhtar, A. Kuestenmacher, P. G. Plöger, and G. Lakemeyer, "Simulation-based Approach for Avoiding External Faults," in *2013 16th International Conference on Advanced Robotics (ICAR)*. IEEE, 2013, pp. 1–8.

[18] P. Hegemann, T. Zechmeister, M. Grotz, K. Hitzler, and T. Asfour, "Learning Symbolic Failure Detection for Grasping and Mobile Manipulation Tasks," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 4302–4309.

[19] D. Park, H. Kim, and C. C. Kemp, "Multimodal Anomaly Detection for Assistive Robots," *Autonomous Robots*, vol. 43, no. 3, pp. 611–629, 2019.

[20] J. T. Brockmann, M. Rudolph, B. Rosenhahn, and B. Wandt, "The voraus-AD Dataset for Anomaly Detection in Robot Applications," *IEEE Transactions on Robotics*, vol. 40, pp. 438–451, 2024.

[21] E. Khalastchi and M. Kalech, "On Fault Detection and Diagnosis in Robotic Systems," *ACM Computing Surveys (CSUR)*, vol. 51, no. 1, pp. 1–24, 2018.

[22] G. Coruhlu, E. Erdem, and V. Patoglu, "Explainable Robotic Plan Execution Monitoring Under Partial Observability," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2495–2515, 2021.

[23] A. Mitrevski, P. G. Plöger, and G. Lakemeyer, "Robot Action Diagnosis and Experience Correction by Falsifying Parameterised Execution Models," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 11 025–11 031.

[24] P. Khanna, E. Yadollahi, M. Björkman, I. Leite, and C. Smith, "Effects of Explanation Strategies to Resolve Failures in Human-Robot Collaboration," in *2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2023, pp. 1829–1836.

[25] Z. Liu, A. Bahety, and S. Song, "REFLECT: Summarizing Robot Experiences for Failure Explanation and Correction," in *Proceedings of The 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Tan, M. Toussaint, and K. Darvish, Eds., vol. 229. PMLR, 06–09 Nov 2023, pp. 3468–3484. [Online]. Available: https://proceedings.mlr.press/v229/liu23g.html

[26] M. Diehl and K. Ramirez-Amaro, "Why Did I Fail? A Causal-Based Method to Find Explanations for Robot Failures," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 8925–8932, 2022.

[27] S. Luo, H. Wu, S. Duan, Y. Lin, and J. Rojas, "Endowing Robots with Longer-term Autonomy by Recovering from External Disturbances in Manipulation Through Grounded Anomaly Classification and Recovery Policies," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 3, p. 51, 2021.

[28] C.-M. Hung, L. Sun, Y. Wu, I. Havoutis, and I. Posner, "Introspective Visuomotor Control: Exploiting Uncertainty in Deep Visuomotor Control for Failure Recovery," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6293–6299.

[29] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. Srinivasan, M. Hwang, J. E. Gonzalez, J. Ibarz, C. Finn, and K. Goldberg, "Recovery RL: Safe Reinforcement Learning with Learned Recovery Zones," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4915–4922, 2021.

[30] A. Kuestenmacher, "Improving the Reliability of Service Robots in the Presence of External Faults," PhD dissertation, RWTH Aachen University, 2019.

[31] M. Basiri, E. Piazza, M. Matteucci, and P. Lima, "Benchmarking Functionalities of Domestic Service Robots Through Scientific Competitions," *KI-Künstliche Intelligenz*, vol. 33, no. 4, pp. 357–367, 2019.

[32] P. So, A. Sarabakha, F. Wu, U. Culha, F. J. Abu-Dakka, and S. Haddadin, "Digital Robot Judge: Building a Task-Centric Performance Database of Real-World Manipulation With Electronic Task Boards," *IEEE Robotics & Automation Magazine*, vol. 31, no. 4, pp. 32–44, 2024.

[33] S. Schneider, F. Hegger, N. Hochgeschwender, R. Dwiputra, A. Moriarty, J. Berghofer, and G. K. Kraetzschmar, "Design and Development of a Benchmarking Testbed for the Factory of the Future," in *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2015, pp. 1–7.

[34] K. Kimble, K. Van Wyk, J. Falco, E. Messina, Y. Sun, M. Shibata, W. Uemura, and Y. Yokokohji, "Benchmarking Protocols for Evaluating Small Parts Robotic Assembly Systems," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 883–889, 2020.

[35] S. Thoduka, D. Nair, P. Caleb-Solly, M. Dragone, F. Cavallo, and N. Hochgeschwender, "Trust in Robot Benchmarking and Benchmarking for Trustworthy Robots," in *Producing Artificial Intelligent Systems: The Roles of Benchmarking, Standardisation and Certification*. Springer, 2024, pp. 31–51.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[37] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-Level Drone Racing Using Deep Reinforcement Learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[38] F. Iodice, E. De Momi, and A. Ajoudani, "HRI30: An Action Recognition Dataset for Industrial Human-Robot Interaction," in *2022 26th International Conference on Pattern Recognition (ICPR)*. IEEE, 2022, pp. 4941–4947.

[39] V. Dutta and T. Zielinska, "Predicting Human Actions Taking into Account Object Affordances," *Journal of Intelligent & Robotic Systems*, vol. 93, pp. 745–761, 2019.

[40] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning Ambidextrous Robot Grasping Policies," *Science Robotics*, vol. 4, no. 26, p. eaau4984, 2019.

[41] M. A. Lee, Y. Zhu, P. Zachares, M. Tan, K. Srinivasan, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making Sense of Vision and Touch: Learning Multimodal Representations for Contact-Rich Tasks," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 582–596, 2020.

[42] D. Sliwowski and D. Lee, "ConditionNET: Learning Preconditions and Effects for Execution Monitoring," *IEEE Robotics and Automation Letters*, vol. 10, no. 2, pp. 1337–1344, 2025.

[43] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *IJCAI'81: 7th International Joint Conference on Artificial Intelligence*, vol. 2, 1981, pp. 674–679.

[44] B. K. Horn and B. G. Schunck, "Determining Optical Flow," *Artificial Intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[45] G. Farnebäck, "Two-Frame Motion Estimation Based on Polynomial Expansion," in *Image Analysis: 13th Scandinavian Conference, SCIA 2003 Halmstad, Sweden, June 29–July 2, 2003 Proceedings 13*. Springer, 2003, pp. 363–370.

[46] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2462–2470.

[47] C. Zach, T. Pock, and H. Bischof, "A Duality Based Approach for Realtime TV-L1 Optical Flow," in *Joint Pattern Recognition Symposium*. Springer, 2007, pp. 214–223.

[48] J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.

[49] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *International Conference on Learning Representations (ICLR)*, 2014.

[50] K. Simonyan and A. Zisserman, "Two-Stream Convolutional Networks for Action Recognition in Videos," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[51] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 248–255.

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All You Need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6000–6010.

[53] H. Fan, B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer, "Multiscale Vision Transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6824–6835.

[54] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, "ViViT: A Video Vision Transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6836–6846.

[55] G. Bertasius, H. Wang, and L. Torresani, "Is Space-Time Attention All You Need for Video Understanding?" in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and

T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 813–824. [Online]. Available: https://proceedings.mlr.press/v139/bertasius21a.html

[56] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, and H. Hu, "Video Swin Transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3202–3211.

[57] G. Ding, F. Sener, and A. Yao, "Temporal Action Segmentation: An Analysis of Modern Techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[58] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "MVTec AD–A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9592–9600.

[59] Y. Zou, J. Jeong, L. Pemula, D. Zhang, and O. Dabeer, "SPot-the-Difference Self-supervised Pre-training for Anomaly Detection and Segmentation," in *European Conference on Computer Vision*. Springer, 2022, pp. 392–408.

[60] P. Mishra, R. Verk, D. Fornasier, C. Piciarelli, and G. L. Foresti, "VT-ADL: A Vision Transformer Network for Image Anomaly Detection and Localization," in *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*. IEEE, 2021, pp. 01–06.

[61] A. Arodi, M. Luck, J.-L. Bedwani, A. Zaimi, G. Li, N. Pouliot, J. Beaudry, and G. M. Caron, "CableInspect-AD: An Expert-Annotated Anomaly Detection Dataset," *Advances in Neural Information Processing Systems*, vol. 37, pp. 64 703–64 716, 2024.

[62] V. Zavrtanik, M. Kristan, and D. Skočaj, "Reconstruction by Inpainting for Visual Anomaly Detection," *Pattern Recognition*, vol. 112, p. 107706, 2021.

[63] Z. You, L. Cui, Y. Shen, K. Yang, X. Lu, Y. Zheng, and X. Le, "A Unified Model for Multi-class Anomaly Detection," *Advances in Neural Information Processing Systems*, vol. 35, pp. 4571–4584, 2022.

[64] J. Yi and S. Yoon, "Patch SVDD: Patch-level SVDD for Anomaly Detection and Segmentation," in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, November 2020.

[65] K. Sohn, C.-L. Li, J. Yoon, M. Jin, and T. Pfister, "Learning and Evaluating Representations for Deep One-Class Classification," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=HCSgyPUfeDj

[66] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "Uninformed Students: Student-Teacher Anomaly Detection with Discriminative Latent Embeddings," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4183–4192.

[67] G. Wang, S. Han, E. Ding, and D. Huang, "Student-Teacher Feature Pyramid Matching for Anomaly Detection," in *32nd British Machine Vision Conference 2021, BMVC 2021, Online, November 22-25*. BMVA Press, 2021, p. 306. [Online]. Available: https://doi.org/10.5244/C.35.349

[68] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, and P. Gehler, "Towards Total Recall in Industrial Anomaly Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14 318–14 328.

[69] N. Li, K. Jiang, Z. Ma, X. Wei, X. Hong, and Y. Gong, "Anomaly Detection via Self-Organizing Map," in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 974–978.

[70] D. Kim, C. Park, S. Cho, and S. Lee, "FAPM: Fast Adaptive Patch Memory for Real-Time Industrial Anomaly Detection," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[71] M. Rudolph, B. Wandt, and B. Rosenhahn, "Same Same but DifferNet: Semi-Supervised Defect Detection With Normalizing Flows," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2021, pp. 1907–1916.

[72] D. Gudovskiy, S. Ishizaka, and K. Kozuka, "CFLOW-AD: Real-Time Unsupervised Anomaly Detection With Localization via Conditional Normalizing Flows," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2022, pp. 98–107.

[73] W. Luo, W. Liu, and S. Gao, "A Revisit of Sparse Coding Based Anomaly Detection in Stacked RNN Framework," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 341–349.

[74] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos, "Anomaly Detection in Crowded Scenes," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 1975–1981.

[75] C. Lu, J. Shi, and J. Jia, "Abnormal Event Detection at 150 FPS in MATLAB," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2720–2727.

[76] B. Ramachandra and M. Jones, "Street Scene: A New Dataset and Evaluation Protocol for Video Anomaly Detection," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 2569–2578.

[77] W. Sultani, C. Chen, and M. Shah, "Real-world Anomaly Detection in Surveillance Videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6479–6488.

[78] S. Haresh, S. Kumar, M. Z. Zia, and Q.-H. Tran, "Towards Anomaly Detection in Dashcam Videos," in *IEEE Intelligent Vehicles Symposium*, 2020.

[79] D. Bogdoll, I. Hamdard, L. Roessler, F. Geisler, M. Bayram, F. Wang, J. Imhof, M. de Campos, A. Tabarov, Y. Yang, M. Gontscharow, H. Gottschalk, and J. M. Zoellner, "AnoVox: A Benchmark for Multimodal Anomaly Detection in Autonomous Driving," in *ECCV 2024 Workshop on Multimodal Perception and Comprehension of Corner Cases in Autonomous Driving*, 2024. [Online]. Available: https://openreview.net/forum?id=Xzb1QlJ98b

[80] Y. Zhao, B. Deng, C. Shen, Y. Liu, H. Lu, and X.-S. Hua, "Spatio-Temporal Autoencoder for Video Anomaly Detection," in *Proceedings of the 25th ACM International Conference on Multimedia*, 2017, pp. 1933–1941.

[81] D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. v. d. Hengel, "Memorizing Normality to Detect Anomaly: Memory-Augmented Deep Autoencoder for Unsupervised Anomaly Detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1705–1714.

[82] Y. Lu, K. M. Kumar, S. Shahabeddin Nabavi, and Y. Wang, "Future Frame Prediction Using Convolutional VRNN for Anomaly Detection," in *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2019, pp. 1–8.

[83] D. Xu, E. Ricci, Y. Yan, J. Song, and N. Sebe, "Learning Deep Representations of Appearance and Motion for Anomalous Event Detection," in *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, September 2015, pp. 8.1–8.12. [Online]. Available: https://dx.doi.org/10.5244/C.29.8

[84] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli, "Adversarially Learned One-Class Classifier for Novelty Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3379–3388.

[85] R. T. Ionescu, S. Smeureanu, M. Popescu, and B. Alexe, "Detecting Abnormal Events in Video Using Narrowed Normality Clusters," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019, pp. 1951–1960.

[86] H. Park, J. Noh, and B. Ham, "Learning Memory-Guided Normality for Anomaly Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020, pp. 14 372–14 381.

[87] J.-C. Feng, F.-T. Hong, and W.-S. Zheng, "MIST: Multiple Instance Self-Training Framework for Video Anomaly Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 009–14 018.

[88] S. Li, F. Liu, and L. Jiao, "Self-Training Multi-Sequence Learning with Transformer for Weakly Supervised Video Anomaly Detection," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 2, pp. 1395–1403, Jun. 2022. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/20028

[89] Y. Zhu, W. Bao, and Q. Yu, "Towards Open Set Video Anomaly Detection," in *European Conference on Computer Vision*. Springer, 2022, pp. 395–412.

[90] D. Epstein, B. Chen, and C. Vondrick, "Oops! Predicting Unintentional Action in Video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 919–929.

[91] A. Flaborea, G. M. D. di Melendugno, L. Plini, L. Scofano, E. De Matteis, A. Furnari, G. M. Farinella, and F. Galasso, "PREGO: Online Mistake Detection in PRocedural EGOcentric Videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 18 483–18 492.

[92] S.-P. Lee, Z. Lu, Z. Zhang, M. Hoai, and E. Elhamifar, "Error Detection in Egocentric Procedural Task Videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 18 655–18 666.

[93] X. Wang, T. Kwon, M. Rad, B. Pan, I. Chakraborty, S. Andrist, D. Bohus, A. Feniello, B. Tekin, F. V. Frujeri *et al.*, "HoloAssist: an Egocentric Human Interaction Dataset for Interactive AI Assistants in the Real World," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 20 270–20 281.

[94] M. Karg and A. Kirsch, "An Expectations Framework for Domestic Robot Assistants," *Advances in Cognitive Systems*, pp. 77–92, 2013.

[95] A. Gautam, T. Whiting, X. Cao, M. A. Goodrich, and J. W. Crandall, "A Method for Designing Autonomous Robots that Know Their Limits," in *2022 International Conference on Robotics and Automation (ICRA)*, May 2022, pp. 121–127.

[96] X. Cao, J. W. Crandall, E. Pedersen, A. Gautam, and M. A. Goodrich, "Proficiency Self-Assessment without Breaking the Robot: Anomaly Detection using Assumption-Alignment Tracking from Safe Experiments," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 12 714–12 720.

[97] M. Fox, J. Gough, and D. Long, "Detecting Execution Failures Using Learned Action Models," in *Proceedings of The National Conference on Artificial Intelligence*, vol. 22, no. 2. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007, p. 968.

[98] D. Park, Z. Erickson, T. Bhattacharjee, and C. C. Kemp, "Multimodal Execution Monitoring for Anomaly Detection During Robot Manipulation," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 407–414.

[99] A. Inceoglu, G. Ince, Y. Yaslan, and S. Sariel, "Failure detection using proprioceptive, auditory and visual modalities," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2491–2496.

[100] D. Park, H. Kim, Y. Hoshi, Z. Erickson, A. Kapusta, and C. C. Kemp, "A Multimodal Execution Monitor with Anomaly Classification for Robot-Assisted Feeding," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 5406–5413.

[101] D. Altan and S. Sariel, "What Went Wrong? Identification of Everyday Object Manipulation Anomalies," *Intelligent Service Robotics*, vol. 14, no. 2, pp. 215–234, 2021.

[102] D. Park, Y. Hoshi, and C. C. Kemp, "A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-based Variational Autoencoder," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1544–1551, 2018.

[103] D. Altan and S. Sariel, "Clue-AI: A Convolutional Three-stream Anomaly Identification Framework for Robot Manipulation," *IEEE Access*, vol. 11, pp. 48 347–48 357, 2023.

[104] V. Arapi, Y. Zhang, G. Averta, M. G. Catalano, D. Rus, C. Della Santina, and M. Bianchi, "To Grasp or Not to Grasp: an End-to-End Deep-Learning Approach for Predicting Grasping Failures in Soft Hands," in *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2020, pp. 653–660.

[105] M. A. Lee, M. Tan, Y. Zhu, and J. Bohg, "Detect, Reject, Correct: Crossmodal Compensation of Corrupted Sensors," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 909–916.

[106] Y. Yoo, C.-Y. Lee, and B.-T. Zhang, "Multimodal Anomaly Detection based on Deep Auto-Encoder for Object Slip Perception of Mobile Manipulation Robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 11 443–11 449.

[107] K. H. Kim, S. Shim, Y. Lim, J. Jeon, J. Choi, B. Kim, and A. S. Yoon, "RaPP: Novelty Detection with Reconstruction along Projection Pathway," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=HkgeGeBYDB

[108] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density Estimation Using Real NVP," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=HkpbnH9lx

[109] C. Xu, T. K. Nguyen, E. Dixon, C. Rodriguez, P. Miller, R. Lee, P. Shah, R. Ambrus, H. Nishimura, and M. Itkina, "Can We Detect Failures Without Failure Data? Uncertainty-Aware Runtime Failure Detection for Imitation Learning Policies," in *Proceedings of Robotics: Science and Systems*, Los Angeles, USA, June 2025.

[110] S. Daftry, S. Zeng, J. A. Bagnell, and M. Hebert, "Introspective Perception: Learning to Predict Failures in Vision Systems," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1743–1750.

[111] H. Liu, S. Dass, R. Martín-Martín, and Y. Zhu, "Model-Based Runtime Monitoring with Interactive Imitation Learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 4154–4161.

[112] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[113] N. Sogi, H. Oyama, T. Shibata, and M. Terao, "Future Predictive Success-or-Failure Classification for Long-Horizon Robotic Tasks," in *2024 International Joint Conference on Neural Networks (IJCNN)*, 2024, pp. 1–8.

[114] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning Latent Dynamics for Planning from Pixels," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 09–15 Jun 2019, pp. 2555–2565.

[115] G. LeMasurier, A. Gautam, Z. Han, J. W. Crandall, and H. A. Yanco, "Reactive or Proactive? How Robots Should Explain Failures," in *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, 2024, pp. 413–422.

[116] Y. Ma, J. Liu, I. Mamaev, and A. Morozov, "Multimodal Failure Prediction for Vision-based Manipulation Tasks with Camera Faults," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 2951–2957.

[117] A. Farid, D. Snyder, A. Ren, and A. Majumdar, "Failure Prediction with Statistical Guarantees for Vision-Based Robot Control," ser. Robotics: Science and Systems. MIT Press Journals, 2022.

[118] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser, "TidyBot: Personalized Robot Assistance with Large Language Models," *Autonomous Robots*, 2023.

[119] J. Duan, W. Yuan, W. Pumacay, Y. R. Wang, K. Ehsani, D. Fox, and R. Krishna, "Manipulate-Anything: Automating Real-World Robots using Vision-Language Models," in *8th Annual Conference on Robot Learning*, 2024. [Online]. Available: https://openreview.net/forum?id=2SYFDG4WRA

[120] A. Abdolmaleki, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, A. Balakrishna, N. Batchelor, A. Bewley, J. Bingham, M. Bloesch *et al.*, "Gemini Robotics 1.5: Pushing the Frontier of Generalist Robots with Advanced Embodied Reasoning, Thinking, and Motion Transfer," *arXiv preprint arXiv:2510.03342*, 2025.

[121] J. F. Mullen Jr, P. Goyal, R. Piramuthu, M. Johnston, D. Manocha, and R. Ghanadan, ""Don't forget to put the milk back!" Dataset for Enabling Embodied Agents to Detect Anomalous Situations," *arXiv preprint arXiv:2404.08827*, 2024.

[122] C. Cornelio and M. Diab, "Recover: A Neuro-Symbolic Framework for Failure Detection and Recovery," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 12 435–12 442.

[123] C. Agia, R. Sinha, J. Yang, Z. Cao, R. Antonova, M. Pavone, and J. Bohg, "Unpacking Failure Modes of Generative Policies: Runtime Monitoring of Consistency and Progress," in *8th Annual Conference on Robot Learning*, 2024. [Online]. Available: https://openreview.net/forum?id=yqLFb0RnDW

[124] Y. Du, K. Konyushkova, M. Denil, A. Raju, J. Landon, F. Hill, N. de Freitas, and S. Cabi, "Vision-Language Models as Success Detectors," in *Proceedings of The 2nd Conference on Lifelong Learning Agents*, ser. Proceedings of Machine Learning Research, S. Chandar, R. Pascanu, H. Sedghi, and D. Precup, Eds., vol. 232. PMLR, 22–25 Aug 2023, pp. 120–136.

[125] J. Duan, W. Pumacay, N. Kumar, Y. R. Wang, S. Tian, W. Yuan, R. Krishna, D. Fox, A. Mandlekar, and Y. Guo, "AHA: A Vision-Language-Model for Detecting and Reasoning Over Failures in Robotic Manipulation," in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=JVkdSi7Ekg

[126] E. Aduh, F. Wang, D. Randle, K. Wang, P. Shah, C. Mitash, and M. Nambi, "Avoiding Object Damage in Robotic Manipulation," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 9518–9525.

[127] Z. Zhou, P. Atreya, Y. L. Tan, K. Pertsch, and S. Levine, "AutoEval: Autonomous Evaluation of Generalist Robot Manipulation Policies in the Real World," in *7th Robot Learning Workshop: Towards Robots with Human-Level Abilities*, 2025. [Online]. Available: https://openreview.net/forum?id=g9k2vlR7VD

[128] L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, D. Salz, M. Neumann, I. Alabdulmohsin, M. Tschannen, E. Bugliarello, T. Unterthiner, D. Keysers, S. Koppula, F. Liu, A. Grycner, A. Gritsenko, N. Houlsby, M. Kumar, K. Rong, J. Eisenschlos, R. Kabra, M. Bauer, M. Bošnjak, X. Chen, M. Minderer, P. Voigtlaender, I. Bica, I. Balazevic, J. Puigcerver, P. Papalampidi, O. Henaff, X. Xiong, R. Soricut, J. Harmsen, and X. Zhai, "PaliGemma: A versatile 3B VLM for transfer," *arXiv preprint arXiv:2407.07726*, 2024.

[129] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "DINOv2: Learning Robust Visual Features without Supervision," *Transactions on Machine Learning Research*, 2024. [Online]. Available: https://hal.science/hal-04376640

[130] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning Transferable Visual Models

From Natural Language Supervision," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 8748–8763.

[131] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in Manipulation Research: The YCB Object and Model Set and Benchmarking Protocols," *IEEE Robotics and Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.

[132] D. Sliwowski, S. Jadav, S. Stanovcic, J. Orbik, J. Heidersberger, and D. Lee, "Demonstrating REASSEMBLE: A Multimodal Dataset for Contact-rich Robotic Assembly and Disassembly," in *Proceedings of Robotics: Science and Systems*, Los Angeles, USA, June 2025.

[133] K. Kimble, J. Albrecht, M. Zimmerman, and J. Falco, "Performance Measures to Benchmark the Grasping, Manipulation, and Assembly of Deformable Objects Typical to Manufacturing Applications," *Frontiers in Robotics and AI*, vol. 9, p. 999348, 2022.

[134] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, "BC-Z: Zero-Shot Task Generalization with Robotic Imitation Learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 991–1002.

[135] "AI2-THOR: An Interactive 3D Environment for Visual AI," *ArXiv*, vol. abs/1712.05474, 2017.

[136] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "VirtualHome: Simulating Household Activities via Programs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8494–8502.

[137] S. Kohl, B. Romera-Paredes, C. Meyer, J. De Fauw, J. R. Ledsam, K. Maier-Hein, S. A. Eslami, D. J. Rezende, and O. Ronneberger, "A Probabilistic U-Net for Segmentation of Ambiguous Images," in *Advances in Neural Information Processing Systems*, 2018, pp. 6965–6975.

[138] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[139] B. S. Reddy and B. N. Chatterji, "An FFT-based Technique for Translation, Rotation, and Scale-Invariant Image Registration," *IEEE Transactions on Image Processing*, vol. 5, no. 8, pp. 1266–1271, 1996.

[140] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *International Conference on Medical Image Computing and Computer Assisted Intervention*. Springer, 2015, pp. 234–241.

[141] D. Jayaraman, F. Ebert, A. A. Efros, and S. Levine, "Time-Agnostic Prediction: Predicting Predictable Video Frames," *International Conference on Learning Representations (ICLR)*, 2019.

[142] B. Ramachandra, M. Jones, and R. R. Vatsavai, "A Survey of Single-Scene Video Anomaly Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 5, pp. 2293–2312, 2020.

[143] W. Liu, W. Luo, D. Lian, and S. Gao, "Future Frame Prediction for Anomaly Detection–A New Baseline," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6536–6545.

[144] P. Gohil, S. Thoduka, and P. G. Plöger, "Sensor Fusion and Multimodal Learning for Robotic Grasp Verification Using Neural Networks," in *2022 26th International Conference on Pattern Recognition (ICPR)*. IEEE, 2022, pp. 5111–5117.

[145] V. Ortenzi, A. Cosgun, T. Pardi, W. P. Chan, E. Croft, and D. Kulić, "Object Handovers: a Review for Robotics," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1855–1873, 2021.

[146] M.-J. Davari, M. Hegedus, K. Gupta, and M. Mehrandezh, "Identifying Multiple Interaction Events from Tactile Data during Robot-Human Object Transfer," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2019, pp. 1–6.

[147] S. Parastegari, E. Noohi, B. Abbasi, and M. Žefran, "A Fail-Safe Object Handover Controller," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2003–2008.

[148] ——, "Failure Recovery in Robot–Human Object Handover," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 660–673, 2018.

[149] P. Rosenberger, A. Cosgun, R. Newbury, J. Kwan, V. Ortenzi, P. Corke, and M. Grafinger, "Object-Independent Human-to-Robot Handovers Using Real Time Robotic Vision," *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 17–23, 2020.

[150] R. Liu, R. Chen, and C. Liu, "Task-Agnostic Adaptation for Safe Human-Robot Handover," *IFAC-PapersOnLine*, vol. 55, no. 41, pp. 175–180, 2022.

[151] Z. Han and H. A. Yanco, "Reasons People Want Explanations After Unrecoverable Pre-Handover Failures," in *ICRA Workshop on Human-Robot Handovers*, 2020. [Online]. Available: https://arxiv.org/abs/2010.02278

[152] C. Meng, T. Zhang, and T. lun Lam, "Fast and Comfortable Interactive Robot-to-Human Object Handover," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 3701–3706.

[153] I. Mamaev, D. Kretsch, H. Alagi, and B. Hein, "Grasp Detection for Robot to Human Handovers Using Capacitive Sensors," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 12 552–12 558.

[154] F. Iori, G. Perovic, F. Cini, A. Mazzeo, E. Falotico, and M. Controzzi, "DMP-Based Reactive Robot-to-Human Handover in Perturbed Scenarios," *International Journal of Social Robotics*, pp. 1–16, 2023.

[155] E. C. Grigore, K. Eder, A. G. Pipe, C. Melhuish, and U. Leonards, "Joint Action Understanding improves Robot-to-Human Object Handover," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 4622–4629.

[156] M. Mohandes, B. Moradi, K. Gupta, and M. Mehrandezh, "Robot to Human Object Handover Using Vision and Joint Torque Sensor Modalities," in *International Conference on Robot Intelligence Technology and Applications*. Springer, 2022, pp. 109–124.

[157] S. Thoduka and N. Hochgeschwender, "Benchmarking Robots by Inducing Failures in Competition Scenarios," in *Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management. AI, Product and Service: 12th International Conference, DHM 2021, Held as Part of the 23rd HCI International Conference, HCII 2021, Virtual Event, July 24–29, 2021, Proceedings, Part II*. Springer, 2021, pp. 263–276.

[158] J. Li, S. Dong, and E. Adelson, "Slip Detection with Combined Tactile and Visual Information," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7772–7777.

[159] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal Convolutional Networks for Action Segmentation and Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 156–165.

[160] S. Li, Y. A. Farha, Y. Liu, M.-M. Cheng, and J. Gall, "MS-TCN++: Multi-Stage Temporal Convolutional Network for Action Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 6, pp. 6647–6658, 2023.

[161] F. Yi, H. Wen, and T. Jiang, "ASFormer: Transformer for Action Segmentation," in *32nd British Machine Vision Conference 2021, BMVC 2021, Online, November 22-25*. BMVA Press, 2021, p. 236.

[162] G. Rouhafzay, A.-M. Cretu, and P. Payeur, "Transfer of Learning from Vision to Touch: A Hybrid Deep Convolutional Neural Network for Visuo-Tactile 3D Object Recognition," *Sensors*, vol. 21, no. 1, p. 113, 2020.

[163] J. M. Gandarias, A. J. García-Cerezo, and J. M. Gómez-de Gabriel, "CNN-Based Methods for Object Recognition With High-Resolution Tactile Sensors," *IEEE Sensors Journal*, vol. 19, no. 16, pp. 6872–6882, 2019.

[164] G. Yan, A. Schmitz, T. P. Tomo, S. Somlor, S. Funabashi, and S. Sugano, "Detection of Slip from Vision and Touch," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3537–3543.

[165] X. Mao, Y. Xu, R. Wen, M. Kasaei, W. Yu, E. Psomopoulou, N. F. Lepora, and Z. Li, "Efficient Tactile Sensing-based Learning from Limited Real-world Demonstrations for Dual-arm Fine Pinch-Grasp Skills," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 5112–5119.

[166] K. Hara, H. Kataoka, and Y. Satoh, "Learning Spatio-Temporal Features with 3D Residual Networks for Action Recognition," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 3154–3160.

[167] ——, "Can Spatiotemporal 3d CNNs Retrace the History of 2D CNNs and ImageNet?" in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6546–6555.

[168] S. Thoduka, S. Houben, J. Gall, and P. G. Plöger, "Enhancing Video-Based Robot Failure Detection Using Task Knowledge," in *2025 European Conference on Mobile Robots (ECMR)*. IEEE, 2025, pp. 1–6.

[169] S. Cui, R. Wang, J. Wei, F. Li, and S. Wang, "Grasp State Assessment of Deformable Objects Using Visual-Tactile Fusion Perception," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 538–544.

[170] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised Learning for Physical Interaction through Video Prediction," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/d9d4f495e875a2e075a1a4a6e1b9770f-Paper.pdf

[171] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "SlowFast Networks for Video Recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6202–6211.

[172] Y. Zhi, Z. Tong, L. Wang, and G. Wu, "MGSampler: An Explainable Sampling Strategy for Video Action Recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 1513–1522.

[173] R. Xian, X. Wang, D. Kothandaraman, and D. Manocha, "PMI Sampler: Patch Similarity Guided Frame Selection for Aerial Action Recognition," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 6982–6991.

[174] S. N. Gowda, M. Rohrbach, and L. Sevilla-Lara, "SMART Frame Selection for Action Recognition," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 2, pp. 1451–1459, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16235

[175] J. Lee, J. Shin, S. W. Ko, S. Ha, and J. Lee, "Scalable Frame Sampling for Video Classification: A Semi-Optimal Policy Approach with Reduced Search Space," in *35th British Machine Vision Conference 2024, BMVC 2024, Glasgow, UK, November 25-28, 2024*. BMVA Press, 2024. [Online]. Available: https://papers.bmvc2024.org/0140.pdf

[176] J. Yoon and M.-K. Choi, "Exploring Video Frame Redundancies for Efficient Data Sampling and Annotation in Instance Segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 3308–3317.

[177] X. Shen, Y. Xiong, C. Zhao, L. Wu, J. Chen, C. Zhu, Z. Liu, F. Xiao, B. Varadarajan, F. Bordes, Z. Liu, H. Xu, B. Soran, R. Krishnamoorthi, M. Elhoseiny, and V. Chandra, "LongVU: Spatiotemporal Adaptive Compression for Long Video-Language Understanding," 2025. [Online]. Available: https://openreview.net/forum?id=G9xhvGPtte

[178] A. Le Guennec, S. Malinowski, and R. Tavenard, "Data Augmentation for Time Series Classification using Convolutional Neural Networks," in *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, Riva Del Garda, Italy, Sep. 2016. [Online]. Available: https://shs.hal.science/halshs-01357973

[179] A. Afshan, J. Guo, S. J. Park, V. Ravi, A. McCree, and A. Alwan, "Variable Frame Rate-Based Data Augmentation to Handle Speaking-Style Variability for Automatic Speaker Verification," in *Interspeech 2020*, 2020, pp. 4318–4322.

[180] V. Ravi, J. Wang, J. Flint, and A. Alwan, "FrAUG: A Frame Rate Based Data Augmentation Method for Depression Detection from Speech Signals," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6267–6271.

[181] Y. Zou, J. Choi, Q. Wang, and J.-B. Huang, "Learning Representational Invariances for Data-Efficient Action Recognition," *Computer Vision and Image Understanding*, vol. 227, p. 103597, 2023.

[182] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[183] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[184] O. X.-E. Collaboration, A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain *et al.*, "Open X-Embodiment: Robotic Learning Datasets and RT-X Models," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6892–6903.

[185] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo,

K. Hsu, J. Hu, D. Jackson, C. Le, Y. Li, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn, "DROID: A Large-Scale In-The-Wild Robot Manipulation Dataset," in *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024.

[186] O. M. Team *et al.*, "Octo: An Open-Source Generalist Robot Policy," in *Robotics: Science and Systems XX (RSS)*, 2024.

[187] NVIDIA, "Isaac Sim." [Online]. Available: https://github.com/isaac-sim/IsaacSim

[188] Y. Jia, G. Wang, Y. Dong, J. Wu, Y. Zeng, H. Lin, Z. Wang, H. Ge, W. Gu, K. Ding *et al.*, "DISCOVERSE: Efficient Robot Simulation in Complex High-Fidelity Environments," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2025, to appear.

[189] "RoboTHOR: An Open Simulation-to-Real Embodied AI Platform," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3164–3174.

[190] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. D. Iii, and K. Crawford, "Datasheets for Datasets," *Communications of the ACM*, vol. 64, no. 12, pp. 86–92, 2021.