POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Tactile Forward Model for Robotized Selective Harvesting of High Value Crops

Tesi di Laurea Magistrale in
Automation and Control Engineering - Ingegneria dell'Automazione

Author: **Gabriele Gandolfi**

Student ID: 968890
Advisor: Prof. Paolo Rocco
Co-advisors: Prof. Amir M. Ghalamzan
Academic Year: 2021-22

"Just keep moving forward."

Abstract

The presence of robots in agriculture has grown considerably in recent years. This trend stems from the concrete need to increase agricultural food production in the face of a growing world population and a steadily decreasing labour force in the fields, as well as the fact that the decreasing amount of land used for cultivation in industrialised countries requires more efficient sowing and harvesting operations. At the same time, it allows new technologies to be tested in an unstructured and therefore challenging environment. In this context, soft fruits and in particular strawberries represent high-value crops that require high human labour input and are consequently expensive and at risk of production decline if labour becomes scarce. Current implementations of robotic systems for strawberry harvesting have come up against the difficulties of an environment governed by the chaotic sense of plant growth where multiple factors can prevent successful harvesting. In this work the problem of Physical Robot Interaction (PRI) is addressed during robot cluster manipulation in strawberries harvesting. The approach uses a data-driven forward model based on tactile predictions to inform the controller about potential future movements of the object being pushed, strawberry stem, using a robot tactile finger. The model is a special type of neural network (NN) that, exploiting state-of-the-art Video Prediction methods, is built to predict future tactile sensations from information obtained by a new image-based tactile sensor developed specifically for this work. The model is integrated into a Deep Functional Predictive Control (d-FPC) system to control the displacement of the stem on the tactile finger during pushes. Pushing an object with a robot finger along a desired trajectory in 3D is a highly nonlinear and complex physical robot interaction, especially when the object is not stably grasped. The proposed approach controls the stem movements on the tactile finger in a prediction horizon. The effectiveness of the proposed approach is demonstrated in a series of tests including pushes of single and multiple stems. The results obtained assume that this approach is generalisable and scalable to other contexts requiring the manipulation in 3D space of delicate objects.

Keywords: Agricultural Robotics, Tactile prediction, Forward models, Cluster manipulation, Functional predictive control, Manipulation via tactile feedback

Abstract in lingua italiana

L'impiego di robot in agricoltura è aumentato notevolmente negli ultimi anni. Questa tendenza deriva dalla concreta necessità di aumentare la produzione di alimenti agricoli a fronte di una popolazione mondiale in continua crescita e di una forza lavoro nei campi in costante calo, nonché dal fatto che la diminuzione delle superfici coltivate nei Paesi industrializzati richiede operazioni di semina e raccolta più efficienti. Allo stesso tempo, consente di testare nuove tecnologie in un ambiente non strutturato e quindi più impegnativo. In questo contesto, la raccolta della frutta, e in particolare delle fragole, rappresenta un'operazione che richiede un elevato apporto di manodopera umana, è di conseguenza costosa e a rischio di declino della produzione se la manodopera scarseggia. Le attuali implementazioni di sistemi robotici per la raccolta delle fragole si sono scontrate con le difficoltà presenti in un ambiente non strutturato, condizionato dalla caotica configurazione assunta dalle piante e dove molteplici fattori possono impedire il successo dell'operazione di raccolta. Questo lavoro affronta il problema delle interazioni del manipolatore robotico con potenziali ostacoli durante la raccolta delle fragole. L'approccio proposto utilizza un modello di previsione data-driven basato su previsioni tattili per informare il controllore sui potenziali movimenti futuri dell'oggetto da spostare, lo stelo della fragola, utilizzando un sensore tattile montato su un dito dell'end-effector del braccio robotico. Il modello è una particolare rete neurale artificiale che, sfruttando i più avanzati metodi di predizione video, prevede le future sensazioni tattili a partire dalle informazioni ottenute da un nuovo sensore tattile basato su immagini, sviluppato appositamente per questo lavoro. Il modello è integrato in un sistema di controllo predittivo (d-FPC) per controllare lo spostamento dello stelo sul sensore tattile durante la manipolazione. Spingere un oggetto con l'end-effector del robot lungo una traiettoria desiderata nello spazio tridimensionale è un'interazione fisica non lineare e complessa, soprattutto quando l'oggetto non è afferrato in modo stabile. L'approccio proposto controlla i movimenti dello stelo sul sensore tattile entro un orizzonte di previsione. L'efficacia della strategia di controllo è dimostrata in una serie di test che includono spinte di steli singoli e multipli (grappolo). I risultati ottenuti presuppongono che questo approccio sia generalizzabile e applicabile ad altri contesti che richiedono la manipolazione nello spazio tridimensionale di oggetti delicati.

Parole chiave: Robot agricoli, Predizione tattile, Modelli predittivi, Manipolazione di frutta a grappoli, Controllo predittivo, Manipolazione con feedback tattili

Contents

1 | Introduction

1.1. Robots in Agricolture

Agricultural activities produce the food necessary for our survival and today make a modest contribution to the world's Gross Domestic Product (GDP) (5%) [56]. Although they still employ the majority of the planet's working population (around 40%), the number of people working in the fields is nowadays decreasing and more concentrated in few parts of the world (Fig.1.1 and Fig.1.2)

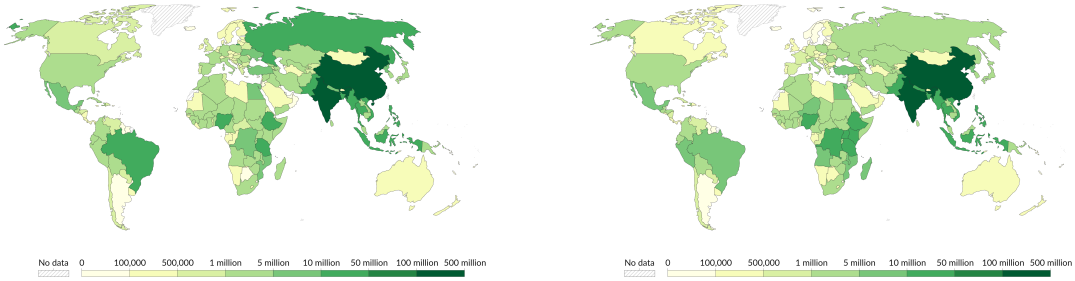


Figure 1.1: Number of people employed in agriculture across the world in 1991 (Left) and in 2019 (Right) [46].

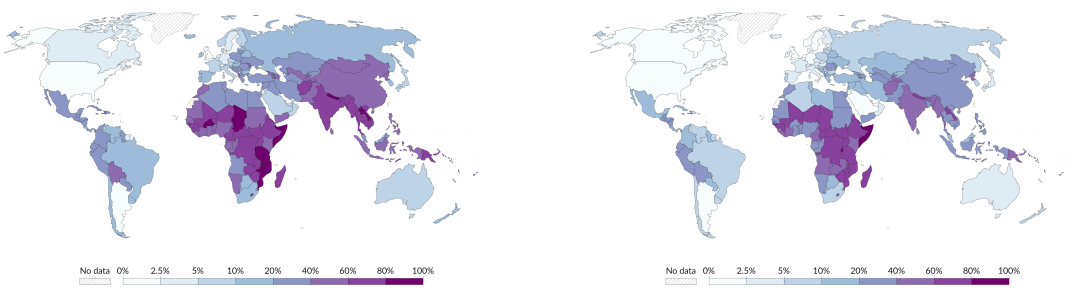


Figure 1.2: Share employed in agriculture across the world in 1991 (Left) and in 2019 (Right). In more industrialised countries, these rates are often less than 5% [46].

Over the last fifty years population has grown considerably and although it has been stabilising recently, continents such as Africa and Asia will still see a large population

expansion. This has led an increase in demand for agricultural products, which simultaneously clashed with the decrease in labour force employed in the fields.

Among the causes that are accelerating this trend are urbanisation and ageing. Urbanisation is now a constant phenomenon which, despite particular cases (such as during the Covid-19 epidemic where on the contrary people preferred to move to the countryside) leads people to abandon rural areas and move to urban centres. Since it is often young people who move to urban centres, the phenomenon of urbanism is linked to that of ageing, as those who remain in the countryside employed in agricultural activities are left without generational change [11] (Fig.1.3).

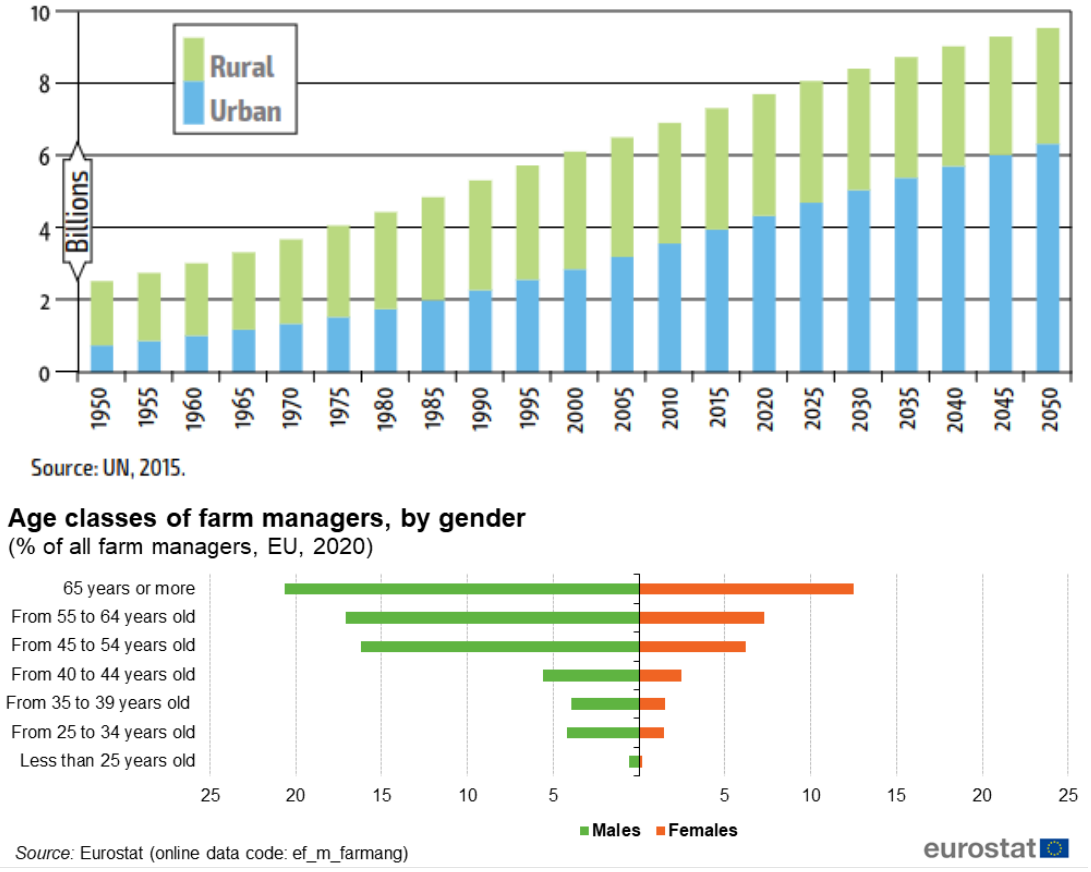


Figure 1.3: Two trends that are accelerating the lack of labour force in agricolture: urbanisation and ageing.

As the world's population has increased, the per capita availability of arable land has halved almost everywhere, while agricultural production has more than doubled.

The increase in agricultural production is due to the increase in land yield, i.e. of production per hectare, while higher productivity has been possible thanks to the increasing mechanisation of agriculture. In recent years agricultural robotics has been a popular

subject from an academic as well as a commercial point of view [54]. This is because agricultural robotics addresses critical issues such as seasonal shortages in manual labor, e.g., during harvest, as well as the increasing concern regarding environmentally friendly practices.
Harvesting is performed several times during production of a high-value crop and is a candidate operation for automation. High-value crops are generally nonstaple crops such as fruit, vegetables, ornamentals, condiments, and spices [4]. A major reason for a crop to be classified as a high-value crop is the high labor input required. Labor costs in Dutch greenhouse horticulture, for instance, constitute 29% of the production costs [27]. These high costs motivate automation of harvesting, and other motivations involve social, environmental, and food quality aspects [63].
Among the high-value crops, strawberries play a particularly important role due to the high demand from the population since they represent a great addition to an healthy diet [20], in fact they are an excellent source of vitamin C and high in flavonoids (bioactive compounds), fiber, potassium, and several antioxidants (one cup of strawberries contains only 55 calories), and therefore the consequent demand for extensive production. Furthermore, for some countries, most notably near us, Spain and the UK, strawberries are among the most produced and exported soft-fruits, generating good revenues. Of the total amount of strawberries consumed in the UK, the 70% is produced in British fields. It is therefore an important asset and resource, and the risks of decreasing production must be avoided (Fig.1.4).

From the birth of the first major research program in robotic fruit harvesting in the 80s [21], many more realities with the aim of automatise the fruit and vegetables harvesting process tackled the task with different methods and techniques [73]. Companies such as Dogtooth (Fig.1.5), Octinion and Agrobot all have commercially released robotic platforms capable of strawberry picking in polytunnels.

It soon became clear that the main issues to be addressed concerned the uncertainty related to an unstructured environment, resulting in difficulties to generate proper reach-to-pick trajectories, obstructions and the characteristics of the objects to be handled, soft fruits, very different from the hardness of industrial products and more prone to slipping. Strawberries are subject to chaotic configurations from the organic cluster formations, causing obstructions to ripe and harvest ready strawberries, as well as complex non-linear interaction dynamics between the fruits, stems, leaves. One of the undesirable results of these complex non-linear interactions is slip. The causes of slip are multiple and depend on the mechanical and aesthetic characteristics of the surfaces in contact, and on external

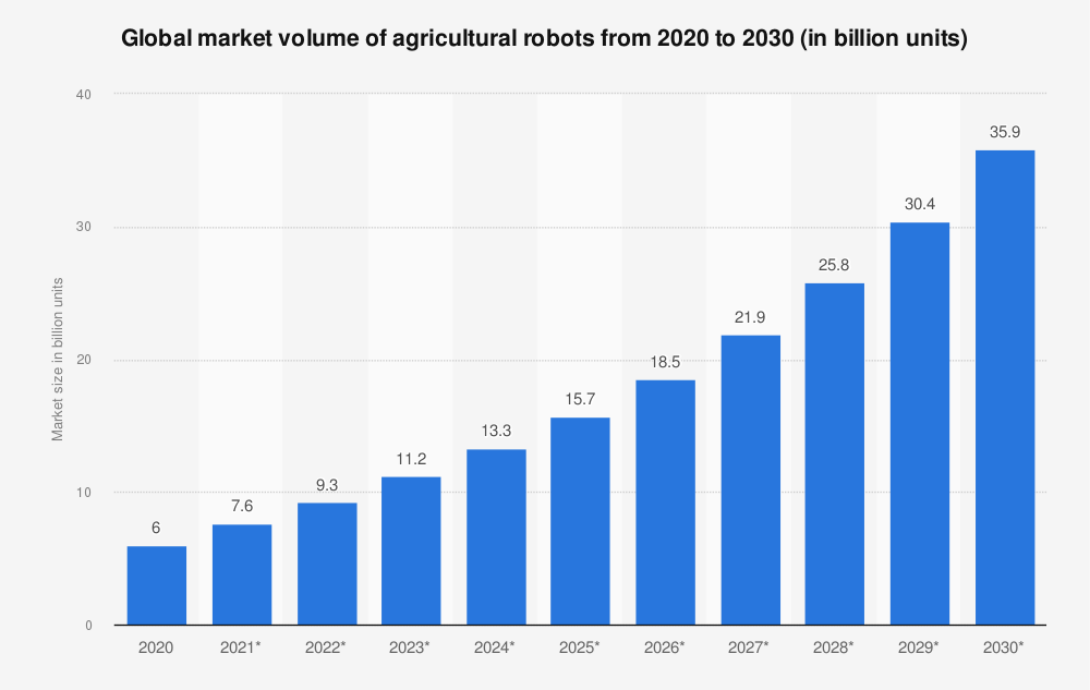


Figure 1.4: Agri-robotics is not a fad [55]. Investment in the sector will increase in the coming years considering the systemic importance of agriculture . According to [17] most of the funds are for harvesting robots and strawberry picking, of all types of soft-fruits, is where robots are most employed.

Figure 1.5: Dogtooth fruit picker robot intent on picking a strawberry in a polytunnel.

elements related to the specific situation. It is a common outcome in cluster manipulation operations and can make it tedious to complete the task quickly.

Other factors such as delicate manipulations are also required to avoid bruising or dam-

aging the fruits, to reduce waste [53]. Many control methods for handling are based solely on increasing grip force, but this is not suitable for handling soft or fragile objects where an increase in force would crush the target and make the effort futile, for a strawberry the average grip force to damage the fruit is 1.5 N [22].

Human intelligence coupled with our dexterity can easily accommodate for this. However, robots require tightly integrated systems which incorporate perception, motion planning and manipulation to be able to operate robustly with high efficiency in these settings [33, 65].

The challenges in the agricultural context provide an opportunity to study and develop alternative methods to be implemented on robots to increase their potential and meet production requirements.
The focus of this work will be to address the strawberry cluster manipulation in an innovative and more human-like way thanks to rich information coming from a custom-made tactile sensor and an approach that falls under the formulation of predictive coding.
The method is effective in circumventing all those interferences that may prevent a successful harvesting and scalable to other applications as it is not case-bound.

1.2. Problem statement and research aim

In this work we introduce a novel approach to address the problem of Physical Robot Interaction (PRI) during robot pushing tasks. In the overall reach-to-pick task, several factors may prevent us from approaching the target fruit or picking a particular fruit selected by a previous algorithm for segmentation. The robotic arm may encounter branches, leaves or other fruits on its way and will have to move them away appropriately to reach its destination. From now on we will refer to this action as push. The push must be guided, controlled and appropriate to prevent displaced elements from slipping off the gripper and returning to the scene, or worse, damaging the target fruit.
Our approach consists in controlling the stem movements during the harvesting task. This is achievable thanks to continuous tactile feedback that we retrieve by a new developed image based tactile sensor. This information are used to make predictions on the future displacement of the stem pushed, we try to minimize the difference between the estimated stem position and a reference, generating a control action to counter the slip on the sensor membrane. This control action correspond to a rotation around the yaw axis of the robot gripper and the result is keeping the stem on a certain position throughout the pushing action.

For the realisation of the Tactile Forward Model we adapted the Computer Vision techniques and the extensive literature on Video Prediction for processing data from an image-based tactile sensor.
The Tactile Forward Model was used to realise a deep Functional Predictive Control (d-FPC), a particular predictive control technique that minimise the number of decision variables and therefore any associated optimisation.
The work presented in this thesis, including model and control strategy development, data collection, training and online testing was carried out in its entirety at the University of Lincoln (UK) [1].

1.3. Thesis achievements

The major contributions and achievements of this thesis are as follows:

- A **tactile prediction method**, rooted in the cognitive predictive paradigm, based on **tactile images** retrieved from **camera-based tactile sensors**;
- An innovative **Tactile Forward Model** built on the principle of tactile prediction. Based on past and present information, the model generates accurate tactile predictions for a future time horizon;
- The **predictive control strategy** integrating the Tactile Forward Model (Fig.1.6). Receiving tactile predictions as input, which provide an estimate of the position of the stem, the control system generates a preventive control action that enables the robotic arm to manipulate the strawberry stems appropriately, counteracting slip instances or loss of contact.

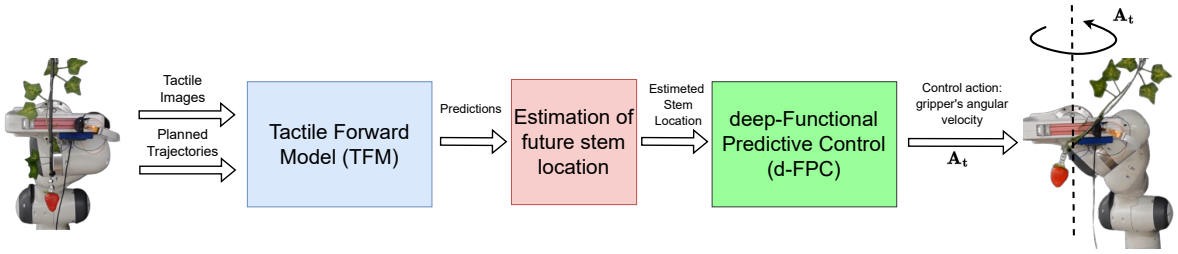


Figure 1.6: Simplified pipeline of the proposed approach for strawberries manipulation.

1.4. Reading guide

This work is structured as follows.

- **Chapter 2**: **Related Works**: Here works that have been an inspiration to this one are briefly summarised, in particular: Video Prediction techniques and architectures important to build the knowledge behind our new proposed model; other approaches to address PRI of unstructured objects configuration and different control strategies for cluster manipulation in agriculture.

- **Chapter 3**: **Tactile Prediction**: The chapter presents the novel idea of our approach and the heart of the strategy, the tactile prediction. First, we will take a glance at the realm of video prediction, showing how we can capture time dependencies from a flux of data; Secondly, we will explain how we can adopt the same strategy for data retrieved from a tactile sensor.

- **Chapter 4**: **Deep Functional Predictive Control (d-FPC)**: We introduce the d-FPC, the predictive controller that integrates the TFM explained in the previous chapter.

- **Chapter 5**: **Experimental Setup**: In this chapter details are provided concerning the tactile sensor used and the dataset collected to train the predictive model, including the manipulation tasks perfomed during the collection and details of the training and the offline testing.

- **Chapter 6**: **Results and Discussion**: Here the results of the online testing are showed. We test our proposed controller in two configurations: single strawberry pushes, cluster of strawberries pushes.

- **Chapter 7**: **Conclusions and future developments**: Conclusions are drawn and evaluated. Future applications and improvements are discussed.

2 | Related Works

The fruit harvesting process consists of several challenging tasks for the robots involved. We have to address the unpredictability and the uncertainty of the environment where natural elements as fruits, stems and branches are not placed in a determined way but follow the naive sense of growth of plants. These unstructured configurations generate non-idealities that the robot will have to face, in particular, occlusions due to branches and leaves obscuring the sight, obstructions to the robot movements to reach the target, caused by branches standing in the way, that generate physical robot interactions (**PRI**) and physical phenomena that can occur during reach-to-pick operations or the final picking task such as slip. The other main challenge to be addressed is the intrinsic characteristic of the target, soft fruits, totally different with respect to the hardness of common industrial components and easily at risk of being damaged. Many control approaches to manipulate object rely on adjusting gripping force to avoid slip [29, 60], but in this context an exaggerated force exerted by the robot would end up compromising the integrity of the fruit or the entire plant.

Here, an overview of the current state-of-the-art (**SOTA**) is given, in particular concerning: strawberries harvesting robotic systems, control strategies and motion planning to perform cluster manipulation and video prediction architectures that inspired the realisation of the novel Tactile Forward Model.

2.1. Robotics in Soft-Fruit Harvesting

The unstructured environment in which robots involved in agriculture have to work poses a considerable challenge for engineers compared to similar tasks but on assembly lines or other structured workplaces.
Robotics researchers and Startup companies alike are starting to develop robotic systems for use in strawberry agricultural settings which are often imprecise and subject to chaotic configurations with complex interaction dynamics.

Octinion has the Strawberry Picker 2.0 which uses a soft-gripper to pick the strawberries [13]. However, their robots are unable to pick strawberries obstructed within clusters as they can only pick strawberries that drop down from the polytunnel trusses due to their ripe weight. A similar approach is taken by [67] in their Robotic Strawberry Harvester (Fig.2.1).

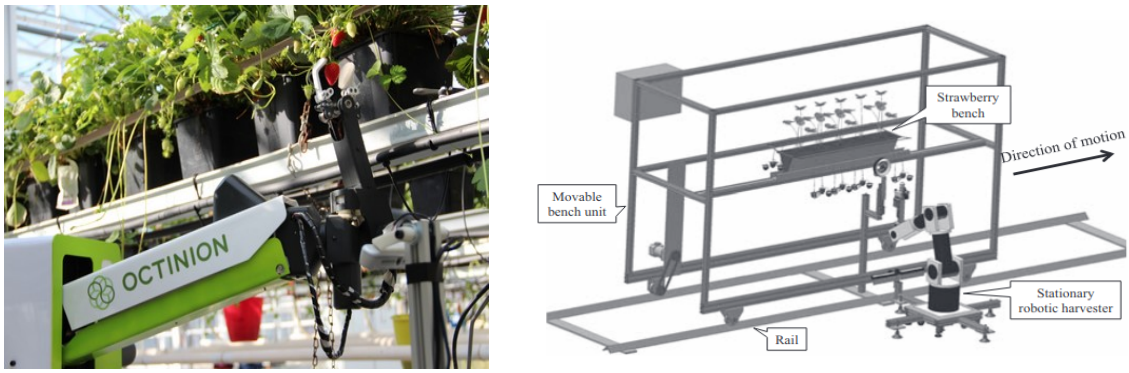


Figure 2.1: Octinion robotic arm reaching strawberry from below (left) and the schematic diagram of the analogue approach used by Yamamoto et al. in their Robotic Strawberry Harvester (right).

More recent development in strawberry harvesting robots in polytunnels include [43] and [57] presenting new perception methods for ready-to-pick strawberries selection and trajectory planning to reach them. In the event of obstructions, making it impossible to reach the strawberry and complete the harvest, it should be noted that the systems mentioned above not deal with such events. To address these situations, [65] developed a three fingered end-effector controlled by cables and servo motors that works as cutter but can handle obstructions (Fig.2.2). However, if the collection does not take place from below and if the environment is sufficiently unstructured, in several situations, the harvesting stage performed with a cutter end-effector may only be possible after an appropriate handling manoeuvre in order to make the target strawberry reachable.

Cluster manipulation in fruit harvesting is a challenging task from both motion planning and motion control perspectives [33, 73]. One of the challenges is avoiding slip of a grasped object, which can be addressed through closed-loop robot trajectory adaptation [38].
The majority of academic robotics research related to soft-fruits and strawberries use trajectory planning methods that either ignore the obstructions caused by fruit clustering completely or by performing obstacle avoidance.

If we broaden our view to the harvesting of different products, [9] proposed an obstacle avoidance enabled trajectory planner for tomatoes harvesting or [61] who proposed

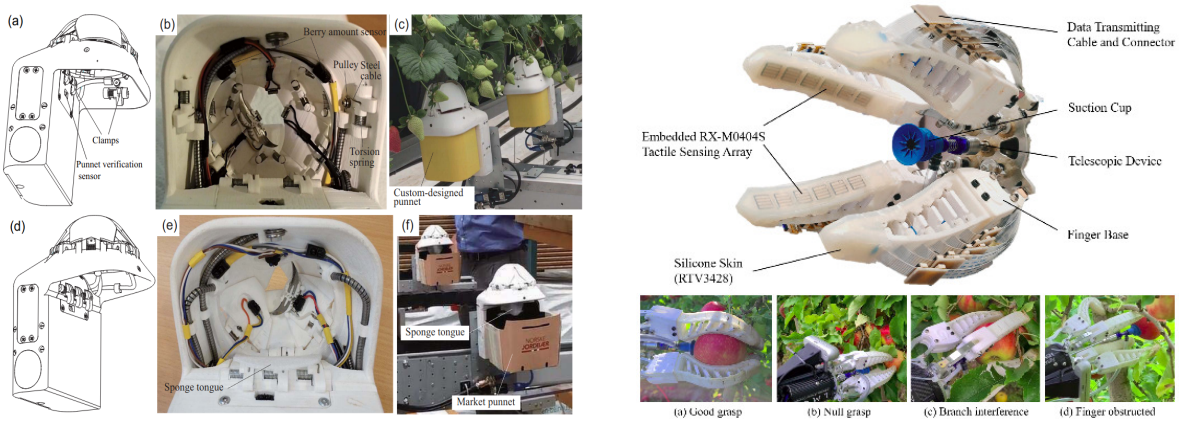


Figure 2.2: Robot end-effectors built to avoid interferences from natural elements in harvesting operations. The Punnet Clump Gripper [65] (left) capture the strawberry to be picked in scenarios where to robot approaches the fruit from below. The sophisticated tactile enabled robotic gripper developed by Zhou et al. [74] to prevent branches contact damages in apples harvesting.

a smooth polynomial-based trajectory planner by applying constraints on velocities and jerks.

Certainly more inherent to our situation, where obstacles are not easily avoidable through a particular trajectory, the works of [74] and [50] are more interesting. The first one introduced a tactile-enabled robotic grasping method that allows robot to sense interference from branches avoiding fruit damages, the latter, using appropriately a dual-arm harvester robot, addresses in human-like behaviour the manipulation and harvest of aubergine plants.

Other works that address the problem of vegetation obscuring the view and approaches to prevent crop damage include [5] with an eye-in-hand sensing to observe the target fruit from several directions in cases of dense vegetation and plan an ideal trajectory and [75], research of which showed how common automated harvesting approaches generate trajectories that bring cherries to collide and damage.

2.2. Modelling interactions with the environment

In presence of PRI, due to contact with branches, leaves or any other elements, to understand how the impact will affect, complicating, our task, we have to represent and address the interaction.

[68] reviewed common modelling methods for deformable objects manipulation highlighting possible weak points. One idea that is fairly well researched in the literature is that

of representing interactions with the environment through analytical models.
Deformable object manipulation, such as cloth, has been modelled using simplified mass-spring models [8], but is inaccurate for large deformations. Position-based dynamics modelling works by manipulating vertex positions of objects meshes drawn by particles representation [69], it's fast and stable but it's visual fidelity only. Continuum mechanics approaches, based on FEM (finite element method), [70] are physically accurate but too complex and expensive to compute. Approaches based on 3D mesh generation [3] have complex dynamics and usually are just locally valid, while heuristic feature spaces have been used for flexible cable manipulation with dual robot arms [76]. The other branch of research in this field is focused on data-driven methods [16], [28, 71] showed the effectiveness of a data-driven method using multiple perception modalities, mainly visual and tactile, however, they are highly task-dependent and need to learn large datasets.

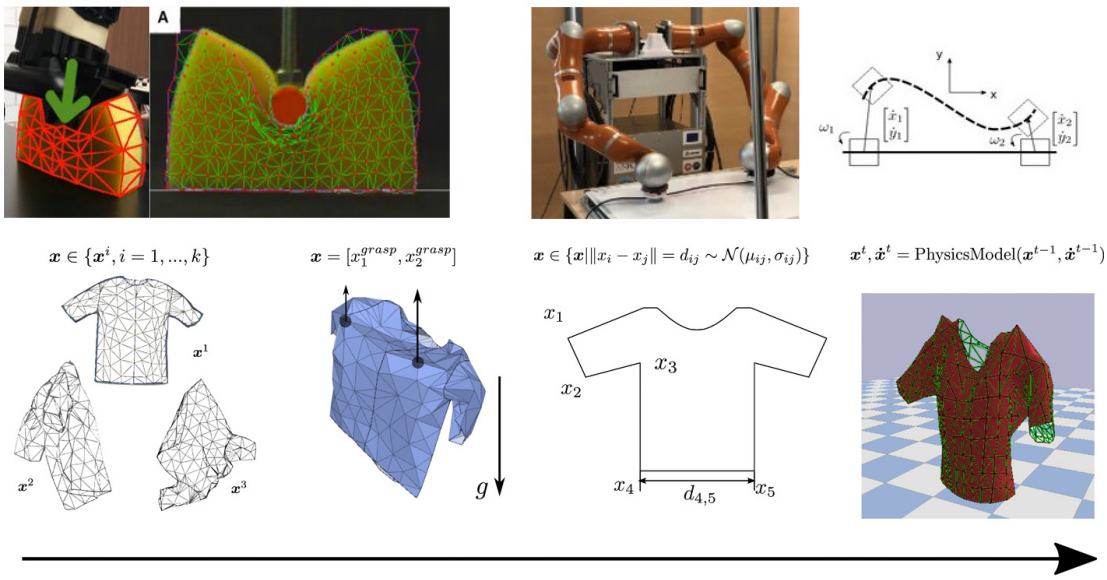


Figure 2.3: Modelling robot interactions with the environment is extremely challenging and case-specific.

In general, analytical modelling methods are limited to specific object sets and are not scalable to larger object and action sets (Fig.2.3). In contrast, our proposed approach uses a time-series model for action-conditioned tactile prediction for pushing control which can be applied in unstructured settings without the knowledge about the model of the individual objects.

2.3. Relying on tactile feedbacks

In the presence of visual occlusions, cameras are no longer reliable for guiding the robotic arm towards the target, and this is where the fundamental importance of equipping the robotic system with tactile sensors becomes apparent (Fig.2.4).

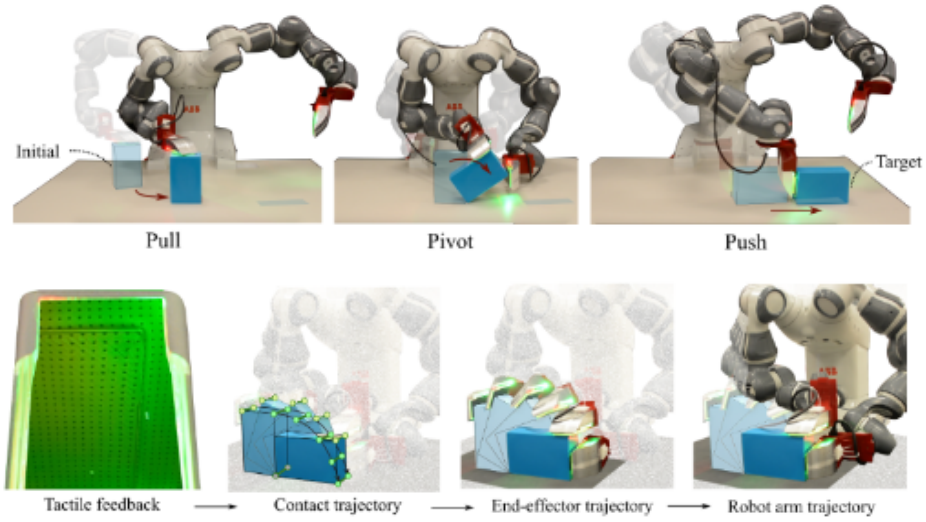


Figure 2.4: Even for common tasks like pushing an object on the surface of a table, relying on tactile sensation can help to complete the task in the best way [24].

Tactile feedback is mostly used for grasp control in robotic object clutter manipulation [66] and detecting a grip on the fruit, detaching and dropping into a basket in harvesting settings [74]. Another line of research uses tactile sensors for ripeness estimation [12, 49], although these studies show the incredible potential of equipping robotic manipulators with a sense of touch, these are operations that can be traced back to harvesting stages where the fruit has already been picked or is otherwise easily grasped. Another interesting field of research on tactile sensation in agriculture is represented by slip detection during fruit picking [15, 58]. However, the use of tactile sensors has been limited to grip control and has not been applied for any cluster manipulation.
[59] exploited tactile sensing for continuous manipulation in the task of pushing objects across a table surface in a controlled manner. This was possible by processing the rich information coming from the GelSight image based tactile sensor (Fig.2.5).

2.3.1. Video and Tactile prediction

Tactile prediction models are used for controlling manipulation tasks, from the simple task of rolling a marble on a table [59] to the complex task of slip control [38]. The core of such controllers is a forward model that can generate predicted tactile readings (we call them tactile images). For instance, action-conditioned tactile predictive models are utilised with a taxel-based tactile sensor in pick and place tasks [32], demonstrating the approach performs well only for flat surface objects.

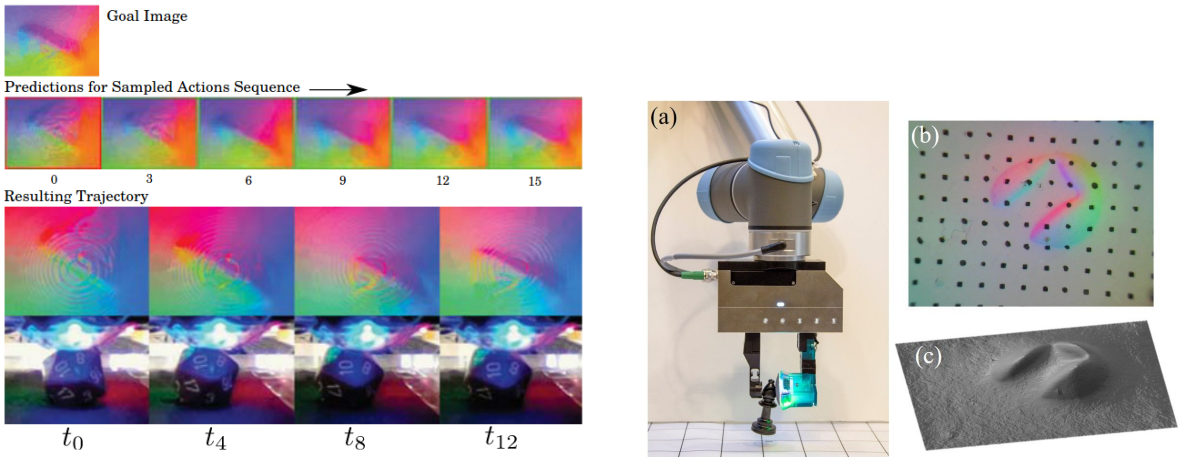


Figure 2.5: An example of successful execution of die rolling task presented in Tian at al. work using the GelSight tactile sensor to retrieve the hidden sight of the object (the robot is pushing from above).

However, in our purposes we have to address pushing in 3D space and with complex shape objects (strawberries and stems), which introduce more variables and difficulties. It is also indispensable to have a sensor that is not only reliable and capable of providing first-class information, but is also able to make its way in such an intricate environment. As a starting point for the realisation of our Tactile Forward Model, we consulted the current video prediction SOTA [41], in particular the models implemented by [18, 32]. [32] proposed an action conditioned forward model used to process tactile information coming from the taxel-based uSkin Xela sensor. The model's simple architecture also makes it ideal for online applications, where computational time must be as low as possible. [18] developed a model to achieve high-quality predictions without overcomplicating the architecture along with good image preprocessing techniques. All of the above methods uses common neural networks layers for image processing as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and, for learning time dependencies, Long Short-Term Memory (LSTM) (Fig.2.6).
It is worth mentioning other works that have contributed to the development of tactile

information processing in PRI contexts and works that introduced approaches to manage the stochasticity associated with the prediction horizon. [72] uses a deep learning framework to classify tactile signals to enhance the adaptability of the robotic grasp system and avoid failures. Towards more accurate video predictions, [14] has brought to light an architecture to address stochasticity to extend the quality of predictions to longer horizons.

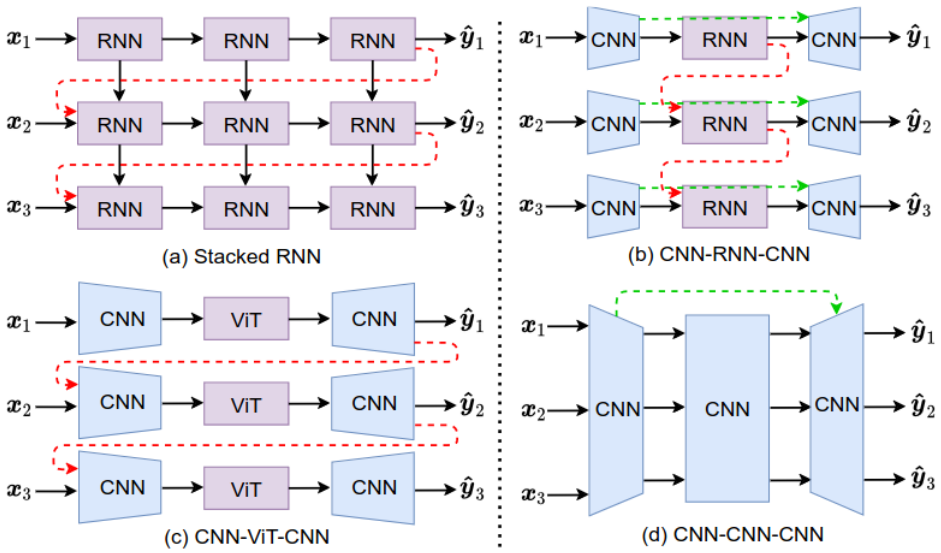


Figure 2.6: Possible predictive models architectures , extract from [18].

2.4. Predictive control

Using the Tactile Forward Model, we developed a deep Functional Predictive Control (d-FPC) [47] which enables the robot to control the strawberry pushing actions. Deep models have been extensively used for learning lower dimensional state spaces for Model Predictive Control (MPC) [31]. These methods have also been used for learning visual dynamic models for control [36]. In a simplified task of rolling a dice, the tactile prediction was used in an MPC controller [59]. In this work, a Proportional-Derivative (PD) control was implemented over the error in the prediction horizon to control the contact state of a flexible object on a robot hand. Unlike previous work that used trajectory adaptation to minimise the likelihood of predicted binary slip signal in a prediction horizon [38], our model learns the complex contact behaviour and generates actions to control the movements of the stem on the tactile finger to keep it stable.

3 | Tactile Predictive Models

3.1. Video Prediction

The ability to **predict**, **anticipate** and reason about **future outcomes** is a key component of intelligent decision-making systems.

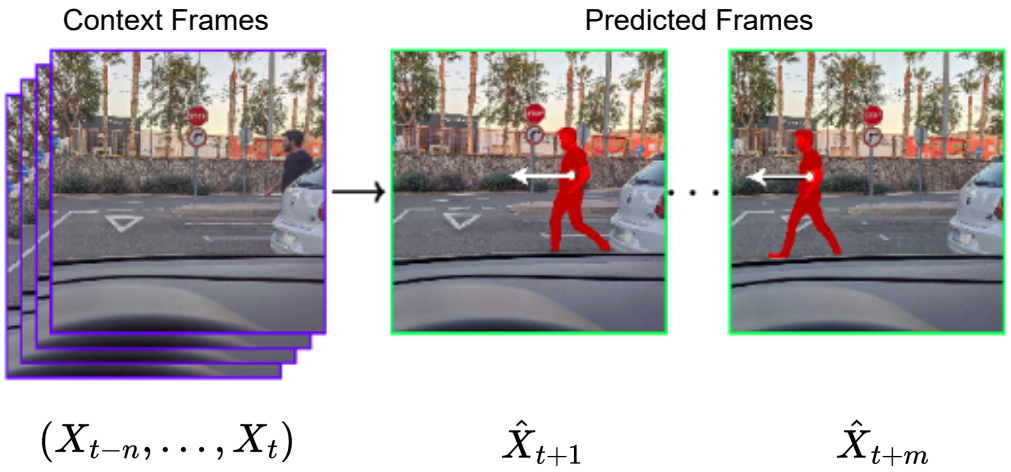


Figure 3.1: An everyday example, from [41], of how we perform video prediction to determine future outcomes of what happens around us and act accordingly. A pedestrian appeared from behind the white car with the intention of crossing the street. The driver of the car must make a call: hit the emergency braking routine or not. This all comes down to predict the next frames $(\hat{X}_{t+1}, ..., \hat{X}_{t+m})$ given a sequence of context frames $(X_{t-n}, ..., X_t)$, where n and m denote the number of context and predicted frames, respectively. From these predictions at a representation level (RGB, high-level semantics, etc.) a decision-making system would make the car avoid the collision.

In everyday situations, we are called upon to formulate expectations about future outcomes of events in order to act in advance in the best possible way. [41] propose an excellent example in Fig.3.1. Will the car hit the pedestrian? When we look at the figure, that might be one of the questions that pops into our heads. Answering this question might be challenging, but if we carefully examine the picture sequence, we might find

some small hints that can assist us in making future predictions., e.g. the person's body indicates that he is moving quickly enough to avoid the car's trajectory. This example is just one situation among many others in which predicting future frames in video is useful. In general terms, the prediction and anticipation of future events is a key component of intelligent decision-making systems.

As humans, we heavily rely on this skill to interact with the environment and each others in order to have a slight sense of control over events and avoid unintended outcomes. We solve this problem quite easily and effortlessly, however, from a machine's point of view it is extremely challenging.
The idea is to model the animals predicting capability based on expecting a certain outcomes after retrieving a sufficient number of previous observations.
Deep learning-based models fit perfectly into the learning by prediction paradigm, enabling the extraction of meaningful spatio-temporal correlations from video data in a self-supervised fashion.
This idea has biological roots, and also draws inspiration from the **predictive coding paradigm** borrowed from the cognitive neuroscience field. From a neuroscience perspective, the human brain builds complex mental representations of the physical and causal rules that govern the world.

3.1.1. The roots are in neuroscience

The first formulation of a predictive theory in neuroscience dates back to 1860 with Hermann von Helmholtz's concept of unconscious inference as part of a theory of visual perception [42]. This theory implies that human vision is incomplete and that details are inferred by the unconscious mind to create a complete picture. Some of the assumptions that the brain makes from the eye's perception are motion and depth perception. The understanding of perception as the interaction between sensory stimuli (bottom-up: from sensory perception) and conceptual knowledge (top-down: from the intellect) continued to be established by Jerome Bruner who, starting in the 1940s, studied the ways in which needs, motivations and expectations influence perception [10].
In the late 1990s, the idea of top-down and bottom-up processing was translated into a computational model of vision by Rao and Ballard in their work [44]. Their paper demonstrated that there could be a generative model of a scene (top-down processing), which would receive feedback via error signals (how much the visual input varied from the prediction), which would subsequently lead to updating the prediction (Fig.3.2).

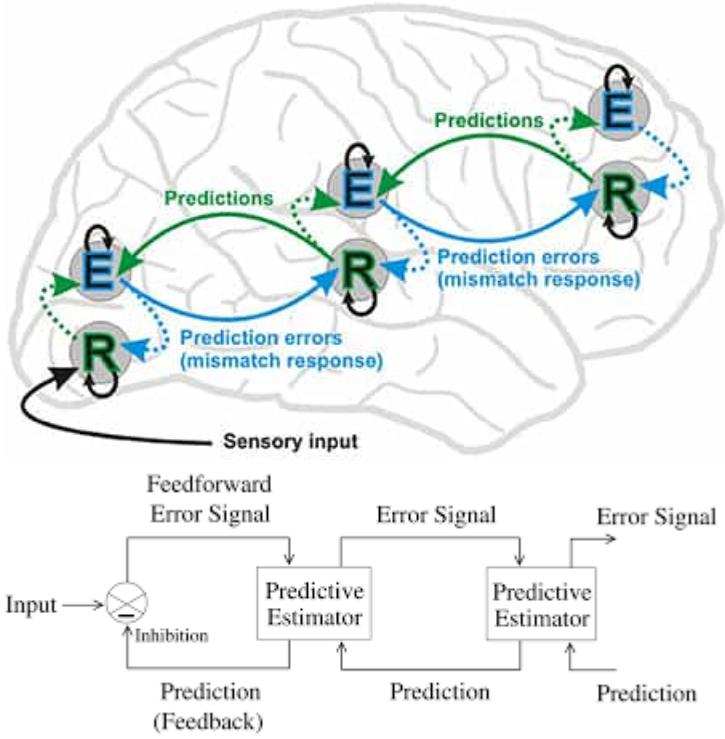


Figure 3.2: The general architecture behind hierarchical predictive coding as formulated by Rao and Ballard in their 1990s paper [44]. At each hierarchical level, feedback pathways carry predictions of neural activity at the lower level, whereas feed forward pathways carry residual errors between the predictions and actual neural activity. These errors are used by the predictive estimator (PE) at each level to correct its current estimate of the input signal and generate the next prediction.

Subsequent formalisations within machine learning and information theory then led to specific proposals for computational architectures.
The development in machine learning and increasingly powerful neural network architectures have enabled the implementation of the learning by prediction paradigm in deep-learning based models, ideally suited for the extraction of significant features and their spatio-temporal correlations in the learning process. Video prediction task closely captures the fundamentals of the predictive coding paradigm and it is considered the intermediate step between raw video data and decision making. Its potential to extract meaningful representations of the underlying patterns in video data makes the video prediction task a promising avenue for self-supervised representation learning.

3.1.2. Formulation

Analytically, we can say that, given $\mathbf{X}_t \in \mathcal{R}^{\omega \times h \times c}$, the t-th frame in the video sequence $\mathbf{X} = (X_{t-n}, ..., X_{t-1}, X_t)$ composed of n frames, where ω, h and c denote width, height and number of channels, respectively, of the frame (e.g. $c = 3$ for an RGB image, $c = 4$ for an RGBD image, where D stands for *Depth*), the target is to predict the next frames $\hat{\mathbf{X}} = (\hat{X}_{t+1}, \hat{X}_{t+2}, ..., \hat{X}_{t+m})$ from the input $\mathbf{X}$.

The task of video prediction is therefore defined as: given a sequence of past video frames, providing the context, the objective is to predict the subsequent future frames - generation of continuing video given a sequence of previous frames.

In contrast to video generation, that is mostly unconditioned, video prediction is conditioned on a previously learned representation from a sequence of input frames. Stated this, a prediction is considered good if the frames of which it is composed are plausible and consistent with the input sequence from which the prediction was generated.
Unlike static images, videos provide complex transformations and motion patterns in the time dimension. At a fine granularity, if we concentrate on a small patch at the same spatial position over several consecutive time steps, we could find a wide range of local visually similar deformations due to the temporal coherence. Conversely, by looking at the big picture, when viewed as a whole, consecutive frames would be visually different but semantically coherent. Occlusions, variations in lighting, and camera motion, among other things, are the primary causes of this variability in the visual appearance of a video at various scales. The dynamics in a video sequence can be depicted by representative spatio-temporal correlations that can be extracted by predictive models from this source of temporally ordered visual clues.

Two elements that play a key role as supervisor signals in video prediction tasks are, motion, as demonstrated by Agrawal et al. [2] and explained in detail in 3.2.1, and time dimension [25]. The latter refers to the temporal ordering in videos, also known as the arrow of time, and indicates whether the video is played forwards or backwards. This information encourages predictive models to implicitly or explicitly model the spatio-temporal correlations of a video sequence to understand the dynamics of a scene. The temporal dimension of a video reduces the supervisory effort and makes the prediction task self-supervised.

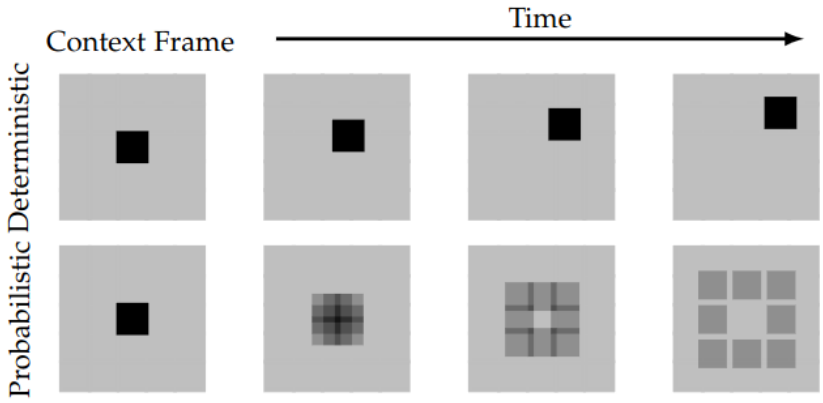


Figure 3.3: At top, a deterministic environment where a geometric object, e.g. a black square, starts moving following a random direction. At bottom, probabilistic outcome. Darker areas correspond to higher probability outcomes. As uncertainty is introduced, probabilities get blurry and averaged. Courtesy: [41].

3.1.3. The future is stochastic and naturally multimodal

By looking at Fig.3.3, we can state that if just one frame is available, even determining the exact future movement of a pixel (a basic shape, a square) in the plane can be extremely challenging. The prediction task is hampered by the lack of contextual information and the existence of numerous equally likely outcomes.
Things change if we start looking at two subsequent frames, under this configuration and assuming a physically perfect environment, the square will be indefinitely moving in the same direction. This represents a deterministic outcome, a common presumption in video prediction tasks. Assuming a deterministic outcome, the prediction space would be constrained to a unique solution. However, this assumption inevitably falls for natural videos, the common observations of our surroundings.
The future is by nature multimodal, since the probability distribution defining all the possible future outcomes in a context has multiple modes, i.e. there are multiple equally probable, valid and legitimate outcomes.
Additionally, on the basis of a deterministic universe, we assume that all possible outcomes are reflected in the input data, in other words the input sequence already allows us to have an idea of all potential futures. These assumptions make the prediction under uncertainty an extremely complex task.
Building a predictive model under deterministic assumptions may be correct, considering that in many everyday contexts, and especially when we have to track and predict the movement of a single object moving independently, given a certain number of past observations we can say with sufficient certainty the direction in which that object will move

in subsequent instants of time. As examples, the motion of car, a pedestrian crossing the street or the bounce of a ball falling perpendicular to the ground, in which case we can say with good confidence that it will come back following the same trajectory, but, with less probability, there are outcomes with bounces in other directions.

When multiple predictions are equally probable, a deterministic model will learn to **average between all the possible outcomes**. This unpredictability is visually represented in the predictions as **blurriness**, especially on large prediction horizons.
This results in a **bound** to the number of future frames we can predict.
The more we move into the future the more the number of possible outcomes increases and so does the average the model will make of this probability distribution. All these future outcomes will be taken into account in the predictions, clearly visible if highly probable or only shaded if the probability of their occurrence is low, but they will still be visible. At some point the prediction will be very fuzzy because of all these possible outcomes that the model takes into account and moving further into the future we will not be able to determine exactly what is happening in the scene and consequently the predictions will be useless.
It tends to be the case that the more subsequent past frames we provide to the model, the more context it will have to understand what is happening in the scene and consequently the predictions will be more accurate and we may go a little further with the prediction horizon.

3.1.4. Neural Networks for Video Prediction

This section will take a look at the common architectures and layers used to perform video prediction. In particular the focus will be on the constructs that have been selected, adapted and implemented for the next stages of this work.

Convolutional Models

Convolutional layers are the basic building blocks of deep learning architectures designed for visual reasoning, since the Convolutional Neural Networks (**CNNs**, Neural Networks that integrate Convolutional layers) efficiently model the spatial structure of images [30].

A Convloutional Neural Network (CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. When compared to other algorithms, a CNN requires much less pre-processing. While filters in archaic methods

are hand-engineered, CNNs can acquire these filters/characteristics with enough training. A convolution is the straightforward process of applying a filter to an input to produce an activation. A feature map, which shows the locations and intensity of a detected feature in an input, an image, is produced by repeatedly applying the same filter to an input.

The architecture of a CNN is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex.
Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area. A CNN is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. Due to the reduction in the number of parameters needed and the reusable nature of the weights, the architecture achieves a superior fitting to the image dataset. To put it another way, the network can be taught to better comprehend the complexity of the image.

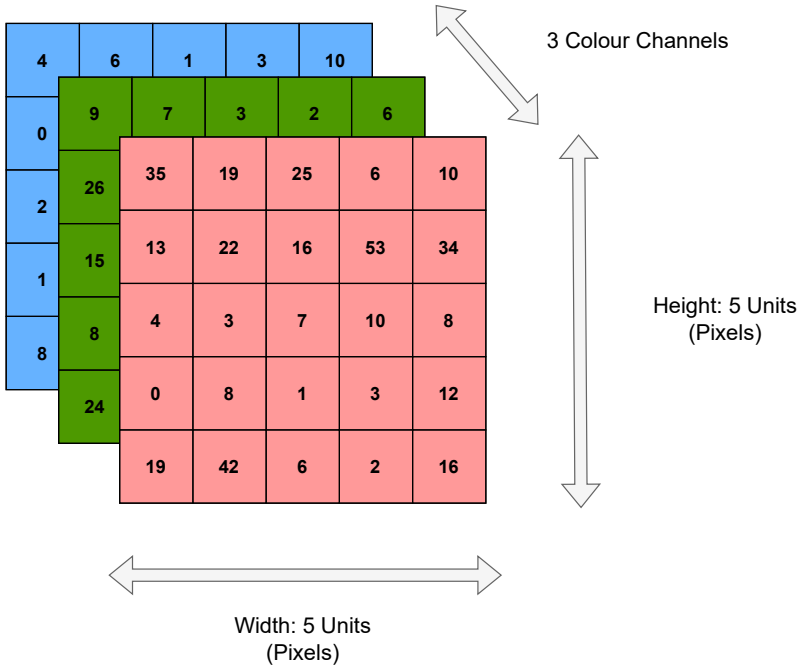


Figure 3.4: 5x5x3 RGB Image.

In the figure (Fig.3.4), we have an RGB image, e.g. 5x5x3, that has been separated by its three color planes — Red, Green, and Blue. As the size increases (e.g. 8K (7680×4320)), the computational complexity becomes too much to handle. The role of ConvNN is to reduce the images into a form that is easier to process, without losing features that are critical for getting a good prediction. This is crucial when creating an architecture that is both scalable to large databases and effective at learning features.

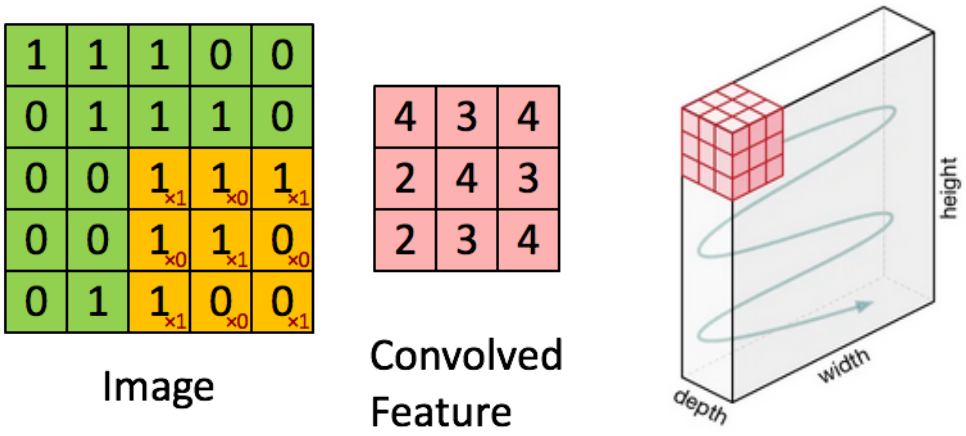


Figure 3.5: The Kernel, a filter used by CNNs to extract the features from an image. The kernel moves over the input data, performs the Hadamard product with the sub-region of input data, and gets the output as the matrix of these products. Courtesy: [48].

In figure 3.5, the green section resembles our 5x5x1 input image, I. The element involved in the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter, K, represented in color yellow. In this example K is a 3x3x1 matrix.

The Kernel shifts 9 times because of Stride Length = 1 (Non-Strided. The Stride indicates how fast the kernel moves along the rows and columns on the input layer e.g. Stride = 2 Fig.3.6), every time performing an elementwise multiplication operation (Hadamard Product) between K and the portion P of the image over which the kernel is hovering.

In the case of images with multiple channels (e.g. RGB), the Kernel has the same depth as that of the input image. Matrix Multiplication is performed between Kn and In stack ([K1, I1]; [K2, I2]; [K3, I3]) and all the results are summed with the bias to give us a squashed one-depth channel Convoluted Feature Output.

The objective of the Convolution Operation is to extract the high-level features from the input image. Usually we use more Convolutional layers; the first one is responsible for capturing the Low-Level features such as edges, color and gradient orientation, other added layers will make the architecture to adapts to the High-Level features for a better understanding of the images.

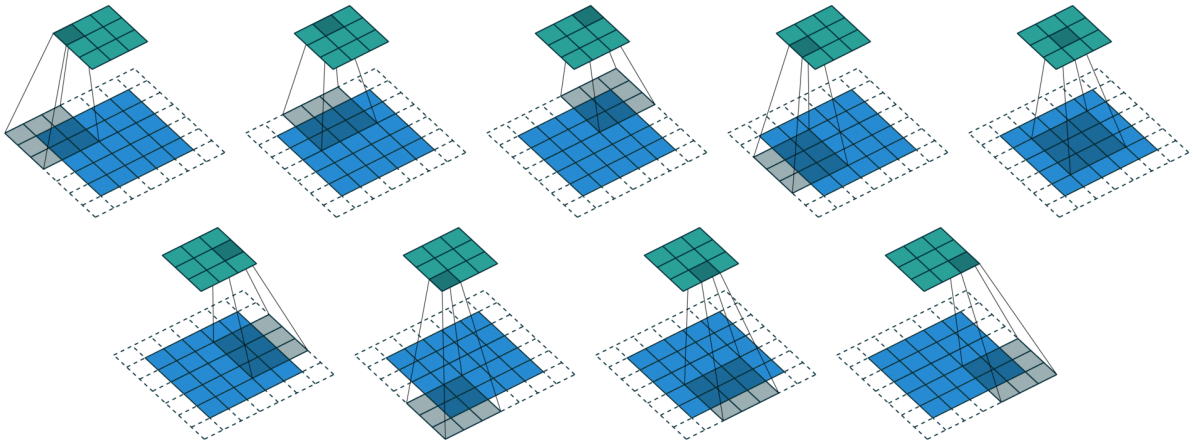


Figure 3.6: Convolution Operation with Stride Length = 2.

Pooling Layer

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training the model.

The output feature maps from Convolutional Layers are sensitive to the location of the features in the input. One method for dealing with this vulnerability is to down sample the feature maps. This makes the resulting downsampled feature maps more resistant to changes in the location of the feature in the image, which is referred to as "local translation invariance".

By summarizing the existence of features in individual feature map patches, pooling layers offer a method for down sampling feature maps. Average pooling and max pooling are two popular pooling techniques that, respectively, summarize a feature's average presence and its most active presence.

Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

After going through the combination of the above processes, we have successfully enabled the model to understand the features. Now, how can we make the model memorise the differences between two or more images, so as to capture that time dependency we talked about when we introduced video prediction?

Convolutional Long Short-Term Memory (ConvLSTM)

In our case, where data are collected as a time series, an interesting approach is to use a model based on **LSTM** (**Long Short Term Memory**) a Recurrent Neural Network architecture. This architecture was first introduce in the work of Hochreiter et al. [23] to solve the long-term dependencies problems of the simple Recurrent Neural Network (RNN) [7].

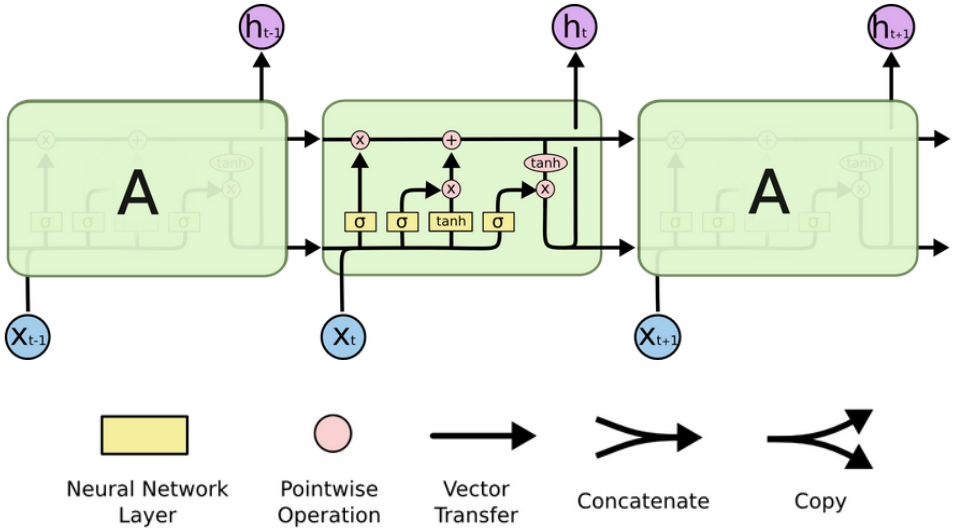


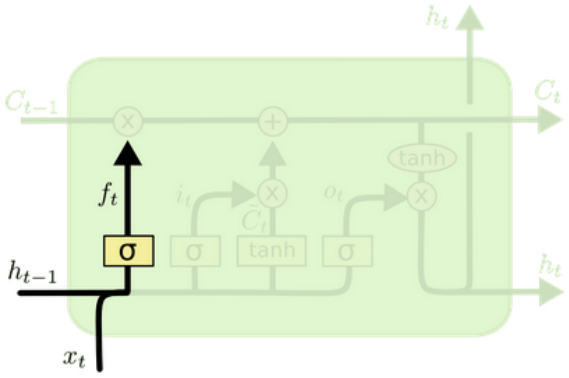
Figure 3.7: An LSTM cell. Courtesy: [39].

Until now, when processing a sequence of data (images) we operate under the assumption that at each iteration the relationship between input and output is independent of the previous output states. Now if we want to extract a time dependancy between two consecutive inputs it is appropriate to drop this assumption. The function value at anyone particular time step can be thought of as directly influenced by the function value at past time steps. There is a temporal dependency between such values. LSTMs are a form of RNN that are excellent at learning such temporal dependencies.

The **cell state**, which permits information to flow from one cell to another, is the fundamental component of LSTMs (Fig.3.7).
This is the LSTM's memory, which can be updated, altered, or forgotten over time. The LSTM components that perform this updating are known as **gates**, and they regulate the information held by the cell. Gates can be thought of as a hybrid of neural network layers and pointwise functions.

Briefly, the main steps performed by an LSTM cell each time new information arrives are as follows:

1 - (Fig.3.8) the amount of data to be deleted from the cell state that is no longer necessary is selected. The forget gate, a component of a neural network with a sigmoid activation function, is in charge of controlling this. The forget gate receives the output of the preceding cell and outputs a value between 0 and 1, indicating how much or little to forget.



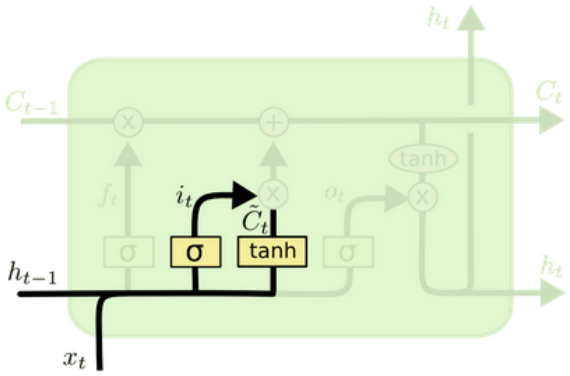
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.1}$$

Figure 3.8: **Step 1**: a sigmoid layer called the "forget gate layer" decides what information we're going to **discard** from the cell state.

2 - (Fig.3.9) Add fresh data and update the cell state. The input gate of a NN layer determines what needs to be updated by adding the output of the prior cell and the current input together. The same concatenation is applied to a tanh layer, which generates an array of potential new values to be added to the state.

3 - (Fig.3.10) To generate a new cell state, modify the old cell state. The previous state is multiplied by the amount established in Step 1 while omitting the earlier-decided to-be-forgotten details. The new candidate numbers discovered in Step 2 are then added. These make up the new cell state, scaled by how much each state number was updated. We can move on to the next cell in the model since this is done for this particular cell.

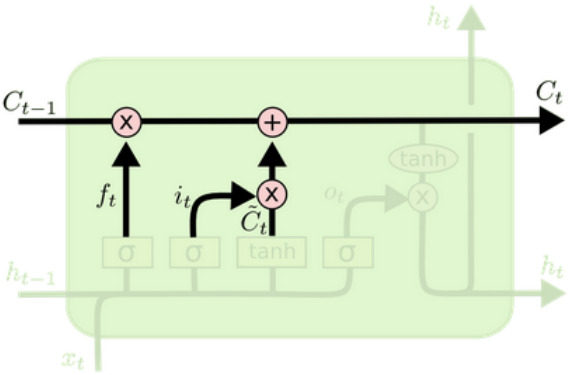
4 - (Fig.3.11) Create the model output using the current input and the prior output. First, a NN layer is passed through the modified cell state. The output of the output/input vector generated by the sigmoid layer is then located, and it is pointwise composed with the altered cell state. We get a suitable output because the cell has complete control over



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.2a}$$

$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_C) \tag{3.2b}$$

Figure 3.9: **Step 2**: a sigmoid layer called the "input gate layer" **decides** what **new information** we're going **to use** to update the cell. Next, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state.

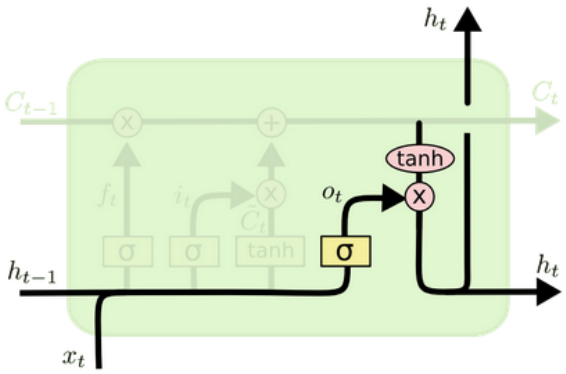


$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{3.3}$$

Figure 3.10: **Step 3**: we **update** the old cell into the new cell based on the old information that we decided to discard (Step 1) and the new information that we decided to store (Step 2).

how its current inputs and state are put together.

In this kind of architecture, the model passes the previous hidden state to the next step of the sequence. Therefore holding information on previous data the network has seen before and using it to make decisions. In other words, the **data order** is extremely important.

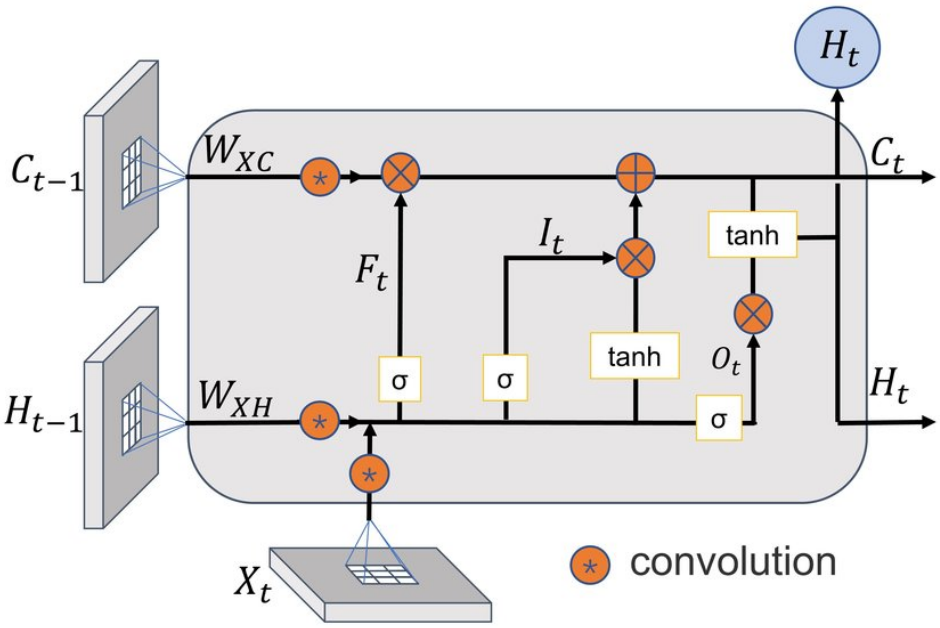


$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.4a}$$
$$h_t = o_t \times tanh(C_t) \tag{3.4b}$$

Figure 3.11: **Step 4**: the model provides a **filtered output**. A sigmoid layer which decides what parts of the cell state we're going to output. Then, a tanh pushes the value between -1 and 1 and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

We have seen before that when working with images, the best approach is a CNN (Convolutional Neural Network) architecture.
The image passes through Convolutional Layers, in which several filters extract important features.
How these two architectures can be connected?

This can easily be achieved by using a convolution operator in the state-to-state and input-to-state transitions, thus, an approach based on **ConvLSTM** (**Convolutional Long Short-Term Memory**) layers [52] (Fig.3.12). It is a Recurrent layer, just like the LSTM, but internal matrix multiplications are exchanged with convolution operations. As a result, the data that flows through the ConvLSTM cells keeps the input dimension (3D in our case: RGB) instead of being just a 1D vector with features, this is the main difference with another type of architecture called CNN-LSTM where a convolutional layer is applied at the beginning to process the data to obtain a 1D array to be fed in a common LSTM.
Before receiving the first inputs, all the states of the ConvLSTM are initialised to zero, a status that corresponds to "total ignorance" of the future.
The PyTorch implementation of a ConvLSTM cell is available in A.



$$i_t = \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \odot C_{t-1} + b_i) \tag{3.5}$$
$$f_t = \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \odot C_{t-1} + b_f) \tag{3.6}$$
$$C_t = f_t \odot C_{t-1} + i_t \odot tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \tag{3.7}$$
$$o_t = \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \odot C_{t-1} + b_o) \tag{3.8}$$
$$H_t = o_t \odot tanh(C_t) \tag{3.9}$$

Figure 3.12: A ConvLSTM cell. This time, in proximity of the state-to-state and input-to-state transitions, there are two convolutional operators to extract image features from new data and from the filtered output of the previous cell. Courtesy: [51].

3.2. Tactile Prediction

Building on the founding principles of Video Prediction, this section presents **Tactile Prediction**.

Touch is an essential tool of survival in humans and primates. It provides them with key information the physical properties of the world and is essential in building physical interaction perception. Human tactile cognition aids in physically interactive tasks such as object pushing, grasping, in-hand manipulation etc. Humans utilise this tactile cognition within predictive cognition to perform daily activities.
In fact, the principles behind predictive coding extend to all sensory organs, and although sight is more intuitive to understand and analyse, in everyday life we rely heavily on touch. We make predictions before grasping and handling objects to do so in the best possible way based on past experiences, or when we need to move objects in a certain way, without them slipping out of our hands or getting back in our way, or when we need to understand in advance the texture and warmth of an object we are going to interact with shortly afterwards. While it is always of fundamental importance, it is worth noting the cases in which predictive coding based on touch becomes vitally important, in particular when we find ourselves in conditions in which we cannot deduce anything from our sight, either because it is in the dark or obscured by dust or sunlight, or for clinical reasons, low vision or blindness.

Without sense of touch, even the easiest actions can be extremely complicated. In 2009, Johansson et al. [26] highlighted this with an interesting example. Imagine picking up a match stick and striking it against a matchbox to light it, a task you have performed with ease many times in your life. But this time, your hand is numb. Johansson et al. performed this experiment, studying the impact of anesthetizing the fingertips of human subjects on their ability to perform this task. The results were striking: human test subjects were unable to even pick up a match stick, much less ignite it. Videos available [45] of this experiment show human subject clumsiness in performing the task that reminds the faltering, lurching struggles of modern robots.
The reason why depriving the sense of touch has such disastrous effects is quickly explained. Touch is distinct among sensory modalities in that it is physically immediate and allows for direct measurement of ongoing contact forces during object interactions, from which friction, compliance, mass, and other physical characteristics of surfaces and objects can be inferred. This knowledge is critical for manipulation tasks like matchstick striking. Visual sensing is a poor substitute: not only is it physically remote, but it is also usually occluded by the actuators at the points of contact (e.g. the hand) [59].

Handling without touch is perhaps comparable to navigation without sight.

3.2.1. Magnetic-based vs image-based tactile sensors

From all tactile sensors developed so far for robotics application we can distinguish two macro-groups: *magnetic-based* and *image-based* tactile sensors (Fig.3.13).
Magnetic-based sensors, or taxel-based sensors, provide high frequency readings at each taxel with tri-axial readings. Considering the most famous and used of these sensors available on the market, the Xela uSKin, it has several magnetic-based cells each measuring normal and shear forces. Its fast readings, essential for control, led this sensor to be successfully used by Mandil et al.[32] for tactile prediction in the specific case of slip avoidance in robot grasping tasks. However, due to the low resolution of the tactile readings, the authors highlighted how their approach is only effective in the manipulation of smooth-surfaced objects and completely detrimental with complex-shape objects.
This is the reason why for this work, where we undoubtedly have to interact with complicated shaped objects such as fruit, stems and branches, we decided to go a step further and use an image-based tactile sensor, custom made as we will see in section 5.1, which, although it comes at a cost in terms of frequencies and abstract readings, provides high resolution data to better understand the characteristics of the objects we interact with, and benefit from the methods developed in computer vision.

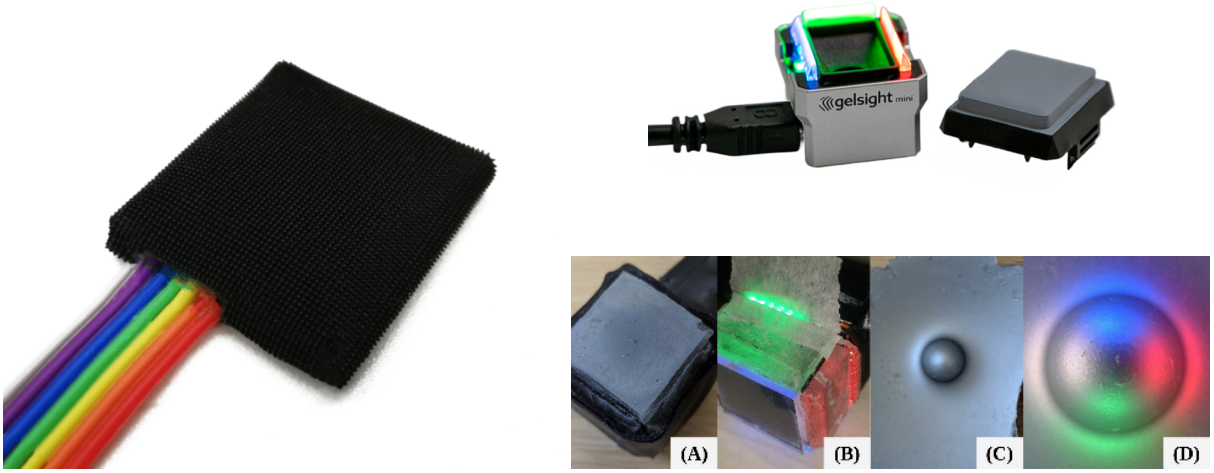


Figure 3.13: On the left, a magnetic-based tactile sensors. The soft membrane retrieves normal and shear forces generated by objects in contact with the sensor in various robotics tasks. On the right, an image-based tactile sensor and its principle of work. A camera captures the deformation of the membrane mounted above. These readings are then processed as normal images.

Action conditioning

Drawing inspiration from what was done by Mandil et al.[32] and the intuition of Agrawal et al.[2] we integrated action conditioning in the development of our predictive model. This expression has the meaning of **conditioning the predictions** of the model by providing as input also the **robot actions**, where, by actions we mean the robot position and orientation in task space, i.e. end-effector position and orientation, during all the pushing action. The fundamental aspect is that we are going to provide actions concerning the whole prediction range: the past and the future that we want to predict. Past actions are actually those already performed by the robot, future actions are those planned but not yet executed and therefore affected by uncertainty.
In their paper, [32] have shown that action-conditioned predictive models are superior, in the sense that they produce qualitatively better predictions, than non-action conditioned predictive models.
The reason why this approach improves performance is subtle, but logical and interesting. In nature, living beings developed the ability of sensory perception for the purpose of moving and acting in the world. Starting from this consideration, it is a question of using the awareness of **egomotion** (i.e. self motion) as a supervisory signal for feature learning. As an example to better understand, consider a linear movement of your arm on the surface of the table, starting from the right, you move your arm to the left. Now, close your eyes and try to predict where your arm will be at the end of the movement. Having full command of your muscles and control of your movement the task will be much easier, even without having a direct visual image of its position.

3.2.2. Methodology

The formulation of the tactile prediction problem will be described in this section, but first emphasis will be placed on the neuroscientific studies that have unveiled the role of forward models for humans and the subsequent attempt to implement them in robots.
Forward models for physiological motor control have been a major topic of neuroscience research over the years [35]. It has been proposed that the central nervous system internally mimics the behaviour of the muscular system in planning, control, and learning based on theoretical and computational investigations. Such an internal forward model of the motor system is a representation of the motor system that forecasts the following state using the motor command and current state of the motor system.
One popular idea is that the cerebellum generates an internal, predictive model of sensory states. This could be used to learn sensorimotor associations as a basis for feedforward

motor control, resolve sensory ambiguities, and predict sensory signals [6]. These internal representations could also be useful for feedforward movement control. A recent neuroimaging study suggests that the cerebellum can learn new forward models after training on an eye–hand tracking task. Miall et al.[34] studied long-term learning of different eye–hand coordination patterns during a tracking task. When the train was performed with the hand leading the eye, the activation was seen in hand-related cerebellar regions, and vice versa. The implication is that the cerebellum was predicting the state of one effector, the hand, to lead the other, the eyes. In a nutshell, the results of the studies show how the human brain uses forward models to predict sensory states to be integrated into motion control. This has found interesting applications in robotics, neural networks and adaptive control such as this work.

Tactile Prediction aims to estimate future tactile images based on a set of previous tactile images $\mathbf{x}_0, ..., \mathbf{x}_{c-1}$ obtained from physical interactions, where c is the length of the context window. Specifically, the objective is to sample from the conditional distribution $p(\mathbf{x}_{c:T}|\mathbf{x}_{0:c-1})$, where $\mathbf{x}_i$ denotes the ith tactile image in the sequence and T is the sum of the context window length and the prediction horizon length.
Stated the importance of the action conditioning approach, since the robot's actions alter the environment during physical interaction, we incorporate action conditioning to predict tactile sensation more accurately. The action-conditioned tactile prediction problem is formulated as predicting the future tactile images $\mathbf{x}_{c:T}$ given a sequence of previous robot actions $\mathbf{a}_{0:c-1}$, previous tactile images $\mathbf{x}_{0:c-1}$, and a sequence of future/planned robot actions/trajectory $\mathbf{a}_{c:T}$. Here, a robot action, $\mathbf{a} \in \mathbb{R}^6$, refers to the end-effector task space position and orientation (Euler angles) with respect to the robot base, while a tactile image is represented by $\mathbf{x} \in \mathbb{R}^{64\times64\times3}$, which captures the surface deformation caused by the applied force.
The conditional distribution will be:

$$p(\mathbf{x}_{c:T}|\mathbf{x}_{0:c-1}, \mathbf{a}_{0:T}) \tag{3.10}$$

Factorising this we can define the model as $\Pi_{t=c}^{T} p_\theta(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{a}_{0:t})$. Learning now involves training the parameters of the factors θ.

3.2.3. Tactile Forward Model (TFM)

Here, the new developed predictive model based on tactile data retrieved from the image-based tactile sensor is presented. By virtue of which, henceforth in this discussion, we will refer to it as **Tactile Forward Model** (**TFM**).

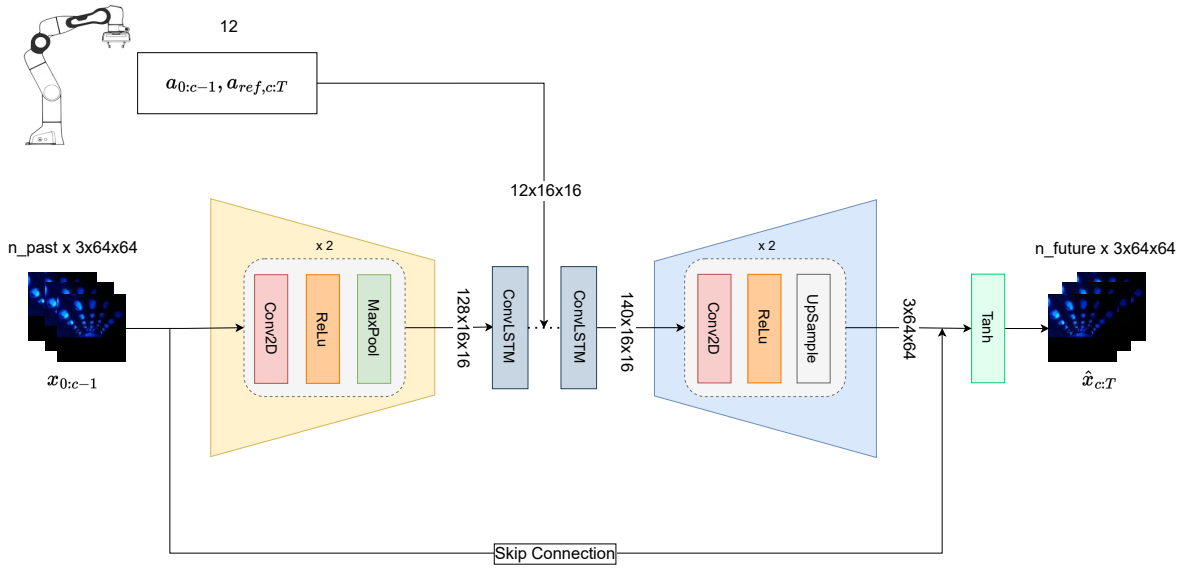


Figure 3.14: The architecture of the **Tactile Forward Model** (**TFM**), a novel architecture for tactile video predictive models: as input, a predefined number of context frames is passed and, as output, a predefined number of future frames is obtained. Between the two ConvLSTM we pass the concatenation of actions and current state of the robot to add information to the network and obtaining better prediction results.

As input we have a predefined number of past frames $\mathbf{x}_{0:c-1}$, information that the model, through the tactile sensor, has already experienced and actions related to the entire horizon: $\mathbf{a}_{0:c-1}$ already performed actions; $\mathbf{a}_{ref,c:T}$ actions planned but not executed yet. Tactile inputs goes into an encoding phase (orange box), with the aim of reducing the size of the image, to simplify and speed up features learning for the ConvLSTMs chain afterwards, and exploiting the number of channels to extract much more features and learning all the details in the scene. We perform two iterations in this encoding phase, each one is characterised by a Conv2D layer (a convolutional layer) followed by the Relu activation function and 2D maxpooling operation. After the encoding phase our original input of dimensions $3 \times 64 \times 64$ will be $128 \times 16 \times 16$.

These latent space features with downsampled width and height and a larger number of channels are fed to the ConvLSTM chain. These layers process the spatiotemporal dependencies among the latent features and, as we have seen before, the model learn

these time dependencies frame by frame. Here, robot action sequences are concatenated with latent tactile features and the model becomes an **action-conditioned predictive model**. These actions, in order to be concatenated, are appropriately converted to same dimensions of latent tactile features.

After this point, we need to upscale the features to reach the tactile image size and reconstruct the images. To this aim, we will now enter a decoding phase (blue box) consisting of two iterations through a sequence of Conv2D layer, a Relu activation function and a 2D UpSampling layer that brings the image back to the input width and height. To apply the pixel motion changes to the input, we use the skip connection for the input tactile image, this connection relate our predicted image to the input one so that it is not an image generated from nowhere but the next in the sequence from the one passed in as input. Finally, we apply a last *tanh* activation function and as output we have the next tactile images in the sequence.

We perform a total of T iterations through the model. The first $c-1$ iterations serve for the model to learn the context. From the iteration c, the model will start to produce the predictions over the horizon we're interested in. At successive instants of time, each time we resort to the model, we pass $c-1$ frames into the past to obtain c to T future frames (Fig.3.15).
The model was entirely made using PyTorch libraries and Python programming language, the definition and initialisation of its layers are available in B.

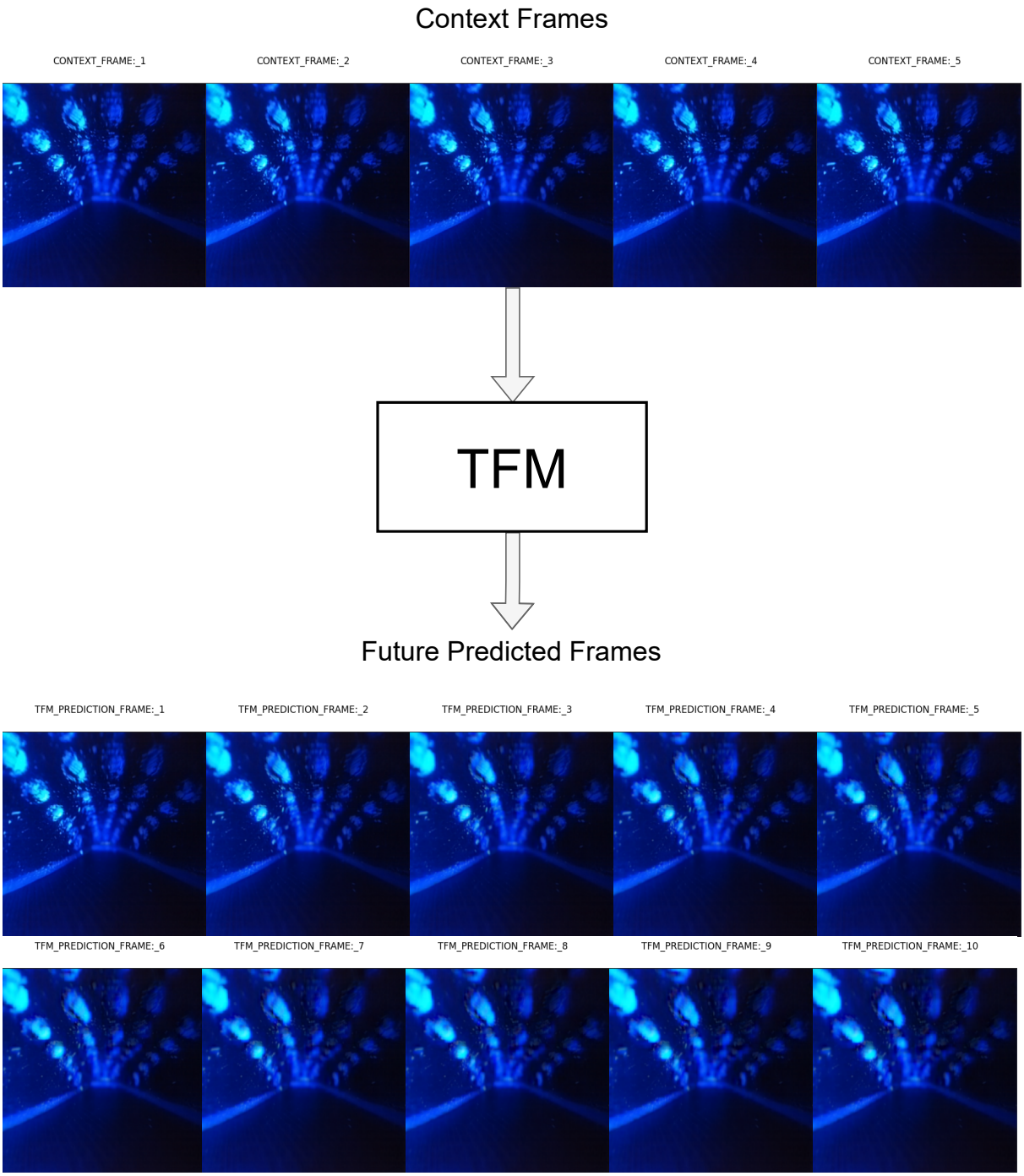


Figure 3.15: Example of predictions using our custom-made image-based tactile sensors. In this example we use five frames for context and we predict the subsequent ten frames. In this particular case, a deformation is occurring in the centre of the sensor, which is actually captured in the predictions (note how the white markers are larger than in the first frames). Towards the last moments of time, the predicted images are less sharp, as the model takes all possible outcomes into account and the result is an average of these outcomes, which in the long run can generate blurring. Other examples available in D.

3.2.4. Different architectures for TFM

Before arriving at this final version of the model, various architectures were tested, mainly differing in input size (Fig.3.19 and 3.20), number of iterations (Fig.3.17) in the encoder and position of the action concatenation (Fig.3.16).
The parameters considered for the evaluation were: the quality of the predictions, assessed through a *qualitative visual analysis* of the predictions and a *quantitative analysis* through common computer vision metrics; the *inference time*, as the model will have to run in real time in a real robotic system, the less computational time it takes to make the predictions the better. Each architecture was trained and tested five times to account for variability in the training phase; the results displayed are averages.

The **qualitative analysis** was carried out as a visual inspection of the generated predictions. In particular, attention was paid to correctness and quantity of the features replicated and on the deterioration of the quality of predictions as future instants of time pass. The qualitative analysis is only reported for cases with different input dimensions, as in the other cases the difference was very minimal.

The **quantitative analysis** concerns the evaluation of the following metrics: **Mean Absolute Error (MAE), Mean Square Error (MSE), Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM)** (The implementations of PSNR and SSIM is available in C).
These values were calculated on the results obtained in the offline test (see section 5.3) using unseen samples previously collected. First we compared the entire prediction horizon (ten frames) with the ground truth, then only the last prediction (in the tables it is indicated by the metric acronym accompanied by "_last").

The **inference time** was calculated with PyTorch *time* function, trivially as the time elapsed in a complete execution of the model's forward pass.

Starting by Figure 3.16, we tested three version of the model with input of size 128×128 and four iterations in the encoding block. In this analysis the difference lies in the **position of the action concatenation**, whether they will be processed by the ConvLSTM chain right from the start, in the middle of the chain or at the end. Of course processing it at the beginning or in the middle will affect the computational time, which was therefore the determining factor for the selection in this case. In the end we opted for the concatenation in the middle, as a trade-off between low computational time and slightly

better metric scores, thus better predictions then in the case of the concatenation at the end.

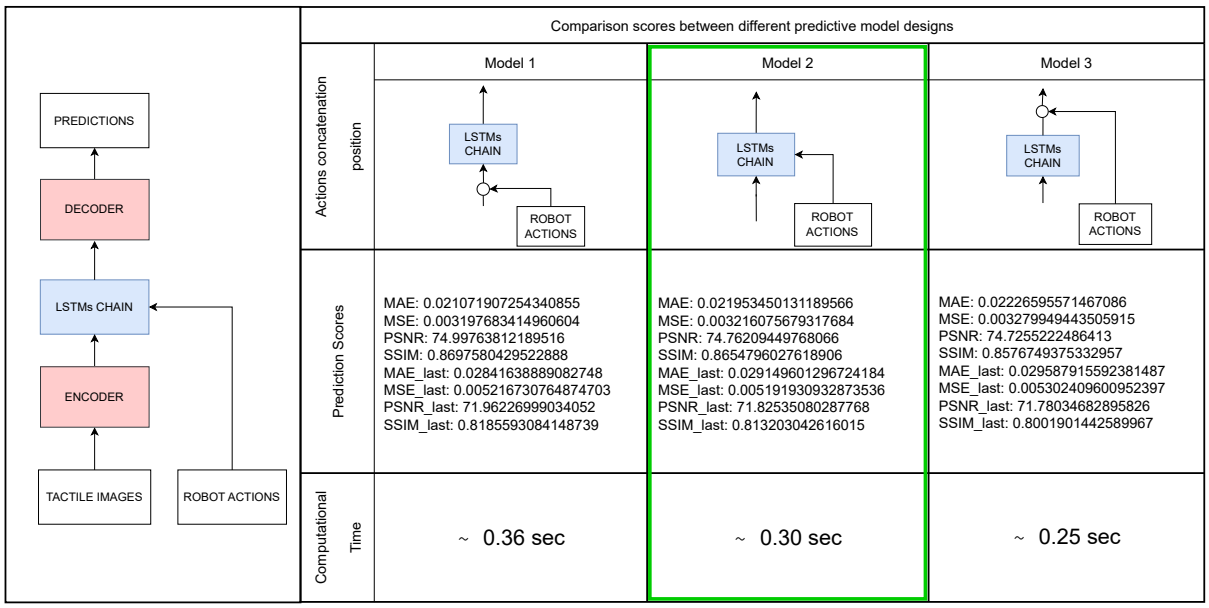


Figure 3.16: Comparison between three models with the same architecture of TFM and that take as input images of size $128 \times 128 \times 3$ but they differ in the position of the actions concatenation. Based on the results we decided to concatenate in the middle of the ConvLSTM chain.

Regarding the number of **encoding iterations** (Scores: Fig.3.17, Whisker plot: Fig.3.18), as one might expect, it is still the inference time that undergoes the most changes, small with only few iterations, it grows as the number increases. we chose two as number of iterations also because they have much better scores than the other cases. This can be explained by the fact that beyond a certain number of iterations, the image is over-compressed and the subsequent reconstruction will be affected by more noise. Two iterations, in this case, is the best choice.

We noticed the biggest changes when varying the **size of the input** to be passed to the model (Fig.3.19 and Fig.3.20). In the first case (256×256) the scores and predictions are better, but the inference time is larger and thus, not ideal for an online application. The middle case is the 128×128, still good predictions and score and quite a large reduction in inference time. The 64×64 test showed that, even though the resolution is lowered and the predictions are not as sharp as in the previous cases, they still manage to capture what is happening in the scene (the movements on the sensor membrane) well enough, plus the inference time drops below one second, which is definitely ideal for real-time applications. By virtue of this, we chose the latter resolution.

Comparison scores between different predictive model designs that differ in the number of encodings				
Iterations	Model 1	Model 2	Model 3	Model 4
	4	3	2	1
Prediction Scores	MAE: 0.02091714965543993 MSE: 0.0025796857573922075 PSNR: 75.50605848561163 SSIM: 0.8716592704472335 MAE_last: 0.02676634023816365 MSE_last: 0.004136785916710758 PSNR_last: 73.03608172872792 SSIM_last: 0.822836692566457	MAE: 0.019130350161424798 MSE: 0.0022761204220486156 PSNR: 76.0465956148894 SSIM: 0.8851694723834163 MAE_last: 0.024793667050645403 MSE_last: 0.003700666704706078 PSNR_last: 73.44357349561608 SSIM_last: 0.8357155180495718	MAE: 0.018234707981995914 MSE: 0.0020915372866590547 PSNR: 76.25638704714568 SSIM: 0.8893455098504606 MAE_last: 0.022907022892942896 MSE_last: 0.003261321288174140 PSNR_last: 73.96992442918861 SSIM_last: 0.8453119798846866	MAE: 0.01845732130839125 MSE: 0.002257993366979265 PSNR: 76.07562471472698 SSIM: 0.8874508090641188 MAE_last: 0.023304281280497493 MSE_last: 0.003538556126169051 PSNR_last: 73.7755821062171 SSIM_last: 0.8412945218708204
Time	~ 0.32 sec	~ 0.28 sec	~ 0.26 sec	~ 0.25 sec

Figure 3.17: Comparison between three models with the same architecture of TFM and that take as input images of size $128 \times 128 \times 3$ but they differ on the number of iterations through the encoding block, thus the amount of features extracted and size reduction. Based on these results we select 2 as number of iterations.

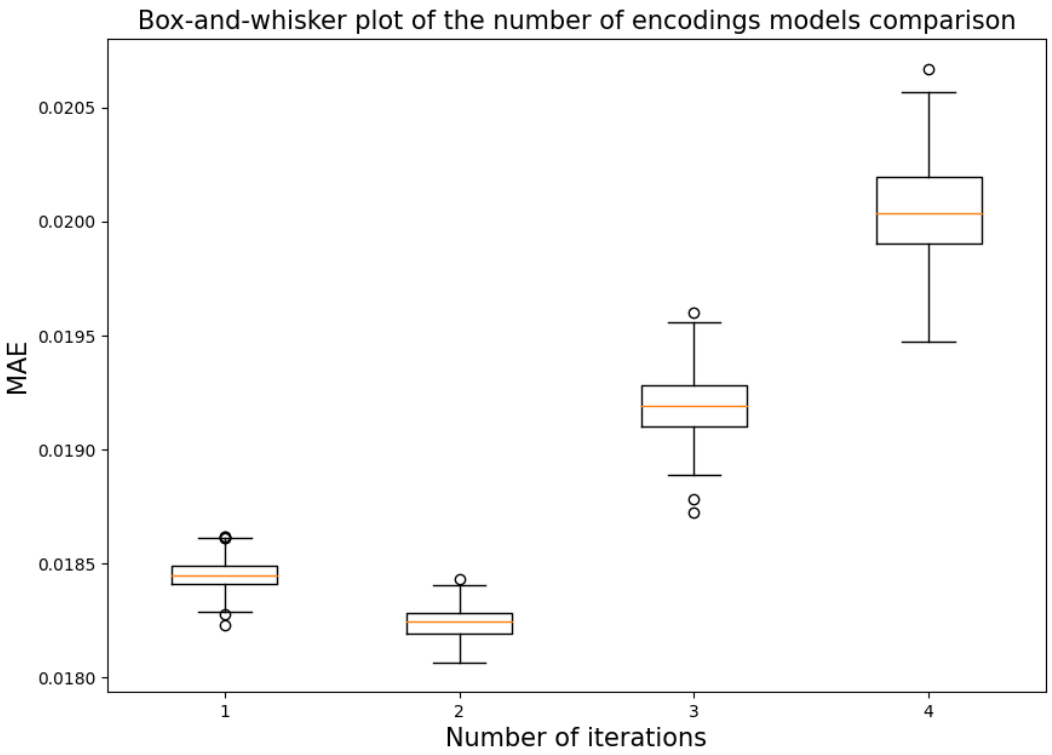


Figure 3.18: Box-and-whisker plot that reinforces the decision to choose for two iterations.

Quantitative analysis - input size comparison			
	Dimensios of the input tactile data	Prediction Scores	Computational Time
Model 1	256 x 256 x 3	MAE: 0.0179259759961101 MSE: 0.0023581537211809873 PSNR: 76.63848926710045 SSIM: 0.9272361307040505 MAE_last: 0.02521198701477893 MSE_last: 0.004039617259758929 PSNR_last: 73.10584026834239 SSIM_last: 0.8823526391516561	~ 1.400 sec
Model 2	128 x 128 x 3	MAE: 0.021953450131189566 MSE: 0.003216075679317684 PSNR: 74.76209449768066 SSIM: 0.8654796027618906 MAE_last: 0.029149601296724184 MSE_last: 0.005191930932873536 PSNR_last: 71.82535080287768 SSIM_last: 0.813203042616015	~ 0.300 sec
Model 3	64 x 64 x 3	MAE: 0.02382128357725299 MSE: 0.003286455541535321 PSNR: 73.9133417710014 SSIM: 0.8878807341275008 MAE_last: 0.03127342448367373 MSE_last: 0.005093646906417511 PSNR_last: 71.41106796264648 SSIM_last: 0.8326989768639855	~ 0.092 sec

Figure 3.19: Quantitative analysis of models that differ in the dimension of the input. Giving priority to the inference time, we chose $64 \times 64 \times 3$ as input dimension.

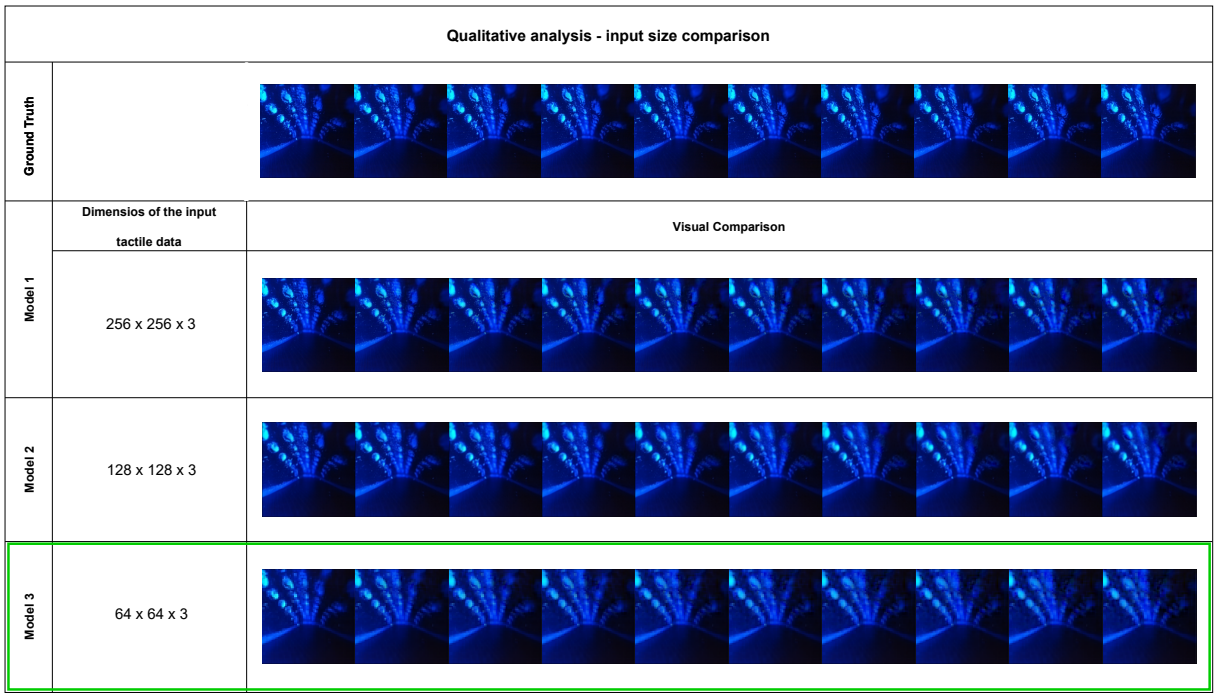


Figure 3.20: Comparison between three models with the same architecture of TFM and they differ on the size of the inputs evaluated thorugh a qualitative analysis.

4 | Deep Functional Predictive Control (d-FPC)

4.1. Control Strategy

In this chapter **Deep Functional Predictive Control (d-FPC)** is presented, as shown in [37]. d-FPC is the predictive controller that integrates the TFM previously explained (4.2).
It has already been mentioned that studies in the field of sensorimotor learning [35, 64] have shown that, for living beings, the central nervous system uses forward models that anticipate sensory perceptions within motion control, these model-based strategies are used in the control of multi-joint movements.
The aim of the controller is to stabilise the stem pushed in a fixed position throughout the entire pushing action, avoiding loose of contact between the stem and the robot finger or slips with risk of damaging the strawberries.
The strategy involves controlling the stem position at each time step providing wrist rotation around yaw axis as control action. Based on the tactile information retrieved from our new developed image-based tactile sensor, the Tactile Finger (5.1), we make predictions, as explained in section 3.2, to estimate at each future time step the position of the stem on the sensor membrane and we aim to minimize the difference with respect the centreline of the sensor.
The overall architecture is depicted in Fig.4.1. The control scheme takes as inputs from the robot tactile images and robot actions in the past from 0 to $c-1$, where $t = c-1$ is the current time instant, and provide as output the control action A_t corresponding to the angular velocity around the yaw axis of the robot gripper.

Functional Predictive Control was developed to outperform PID control, as it is based on predictions, it is able to embed systematic constraint handling and can deal with more challenging dynamics,, while being of similar cost and implementation complexity. Moreover, it is designed to be computationally efficient and relatively easy to implement, unlike Model Predictive Control (MPC).

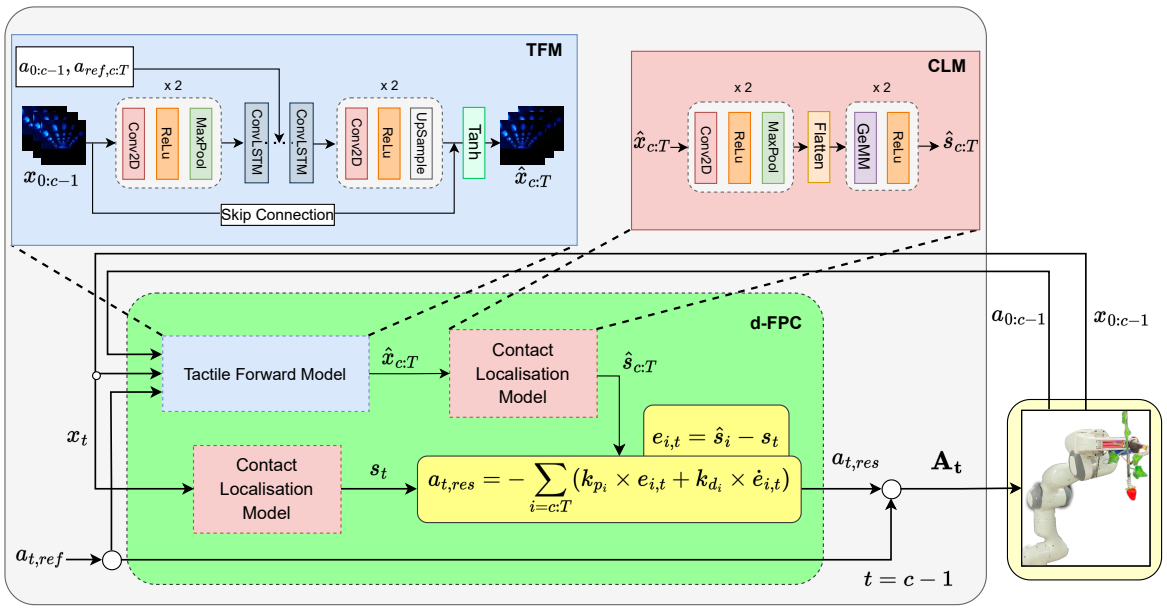


Figure 4.1: The block diagram of the proposed data-driven functional predictive control for pushing strawberries. The model consists of (i) tactile forward model (**TFM**), (ii) contact localisation model (**CLM**), and (iii) the functional predictive controller (**d-FPC**) that generates future actions resulting in the minimum stem displacement on the tactile finger.

Our novel deep functional predictive control pipeline consists of three key modules: a deep action-conditioned Tactile Forward Model (**TFM**), a deep Contact Localisation Model (**CLM**), and an online deep Functional Predictive Control (**d-FPC**) to generate control actions. In the next sections we will highlight the roles played by each modules.

4.2. Tactile Forward Model (TFM)

The idea and operating principle behind the Tactile Forward Model have already been extensively illustrated in Chapter 3. Therefore, here we will merely summarise its function within the control scheme.

With reference to the scheme in Fig.4.2, showing an enlargement of the model architecture highlighting input and output, our TFM takes as input from 0 to $c-1$ previous tactile readings of the Tactile Finger, from 0 to $c-1$ previous actions performed by the robot plus future actions, planned and not performed yet, from c to T and outputs predictions from c to T.

The model is pre-trained offline on a data set previously collected (see Chapter 5) containing samples of the collaborative robot Franka Emika Panda pushing a cluster of plastic

strawberries with the tactile sensor mounted on the robot finger. To reduce the inference time and achieving better results in the online testing, the model was optimised with the open-source toolkit OpenVINO, thanks to which the inference time was brought below 0.09 seconds.

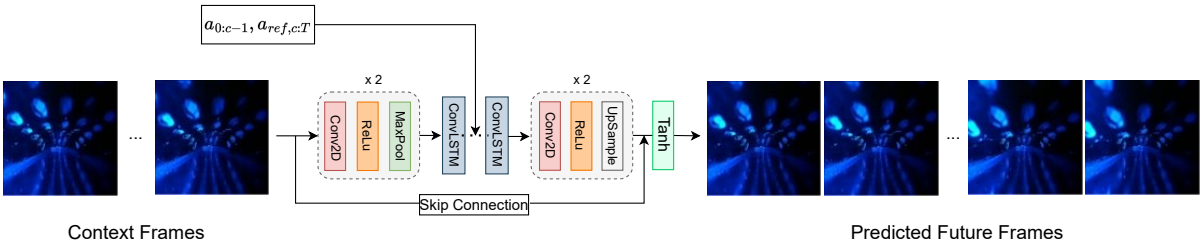


Figure 4.2: Focus on the TFM, the first module of our d-FPC. From 0 to $c-1$ previous tactile readings and from 0 to $c-1$ previous actions performed and c to T future planned actions, the model outputs future predictions from c to T.

4.3. Contact Localisation Model (CLM)

The motions of the marker array printed on the sensor are indicative of the magnitude and location of the applied force. For the current problem setting, we are more interested in force localisation for doing stem contact state control. The aim of the Contact Localisation Model (CLM) is to estimate the stem location from the motions of the white markers detected in the predictions provided by TFM.

Therefore, to find the mapping from raw tactile images to contact location in 1-dimensional space, we use this particular Convolutional Neural Network with the architecture shown in Fig.4.3. CLM consists of two convolutional and three dense layers. The output of CLM is the distance of the contact force from the sensor camera lens along the sensor conic axis, as shown in Fig.4.4.

The data set for training CLM consists of applying forces to the fixed sensor by a rod (mimicking strawberry stem) attached to the robot end-effector (EE) with a 5mm distance step. At each step, the robot applies force on the membrane toward the sensor base by a 1mm penetration step. Overall, 150 stem pushing samples in 10 locations are collected to train CLM.

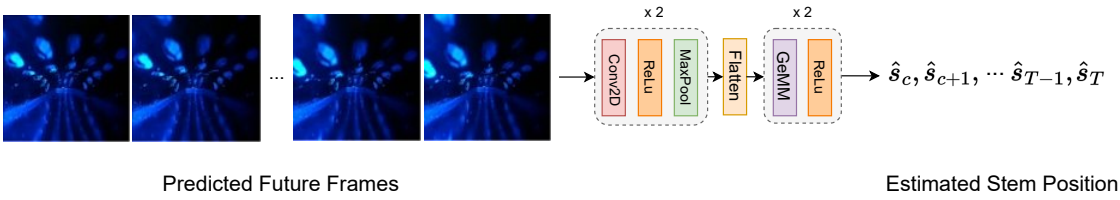


Figure 4.3: Insight of the Contact Localisation Model (CLM). For each predicted tactile image coming from the TFM, it outputs an estimate of the position of the stem on the sensor membrane. This is possible by detecting the displacement of the white markers in the images.

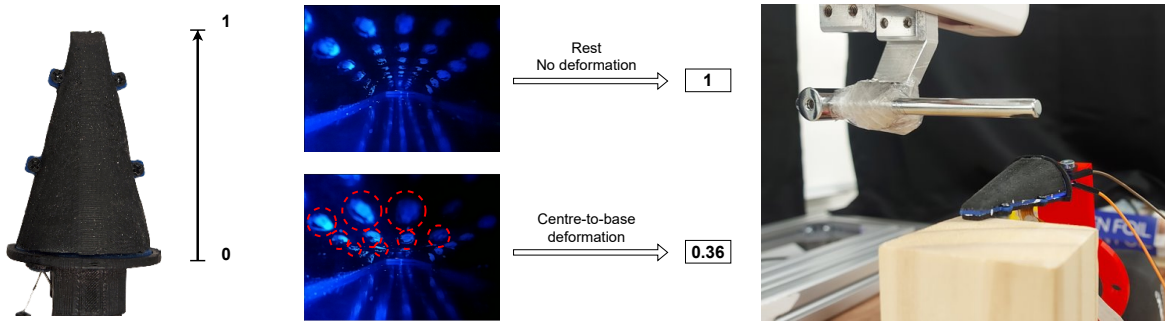


Figure 4.4: CLM operates with one dimensional localisation coordinate normalised from 0 to 1. When pressure is applied on the sensor membrane, based on the displacements and visibility of the markers is possible to estimate a position whitin this interval of values. On the Right the calibration procedure.

4.4. Deep Functional Predictive Control (d-FPC)

Functional Predictive Controllers (FPC) were introduced as a bridge between PIDs (of which they are the main alternative) and MPCs [19]. The advantage of FPC over PID control is the embedded ability to control dead time processes and to constrain both the manipulated and the controlled signal. In addition, all the parameters have physical meaning which helps in formulating the control strategy in practice. On the contrary, for big plants or systems to be controlled with many variables at play, MPCs are generally resorted to, which use a complex numerical optimization of the actual and future manipulated variable sequence by taking into account different constraints. This complexity can be difficult to manage and undoubtedly redundant or unnecessary for simple systems with only one input and one output (SISO) or generally few variables and parameters.
Before formulating the control action for our specific case, it is good to introduce the basic

formulation of the FPC as it originated, for simplicity's sake for a SISO system with a stepped reference.
The principle of FPC is that the controlled variable y accomplishes the reference trajectory at the target points using few changes in the manipulate variable u.
The desired change in the controlled variable y during the prediction horizon $t : T$ (t is the actual time) is calculated from the desired change of the reference trajectory and the predicted change of a model output y_m. The change in the reference trajectory and the predicted change in the model output during the prediction horizon allow for a calculation of the manipulate variable (Fig.4.5).

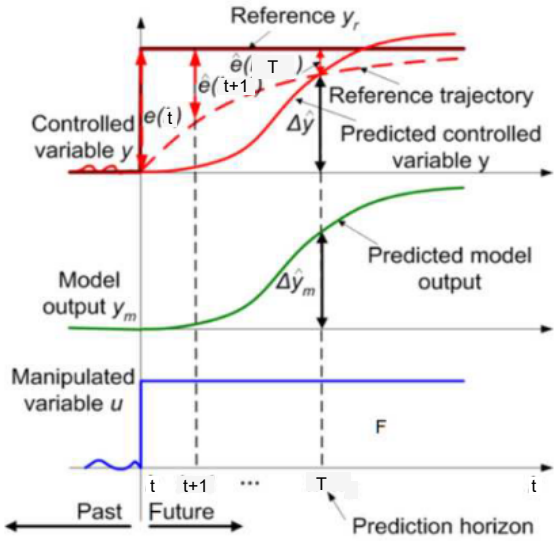


Figure 4.5: FPC principle in a SISO case and stepwise response.

In our case, the variable to be controlled is the location of the stem to be pushed respectively to the normalised (1-dimensional) sensor membrane. We want it to remain as close as possible to the position it was at the previous time instant, so, by minimising the deviation error, we ideally want it to remain at the first contact position it was at the start of the pushing action. The result will be a corrective action to the reference trajectory given to perform the push.

We denote the predicted stem location (from CLM) on the sensor at time t by s_t. The goal of our d-FPC, *deep* because the future estimated locations of the stem are retrieved from predictions made by a Deep Learning model, is to control the stem displacement on the tactile finger. Hence, this allows the robot to keep the contact fixed with the strawberry stem during pushing actions and avoid the contact location approaching the tip or the base of the sensor. These are sensor surface boundary zones and approaching them increases the probability of losing contact with the stem compromising the task. We

use the stem-finger contact point at time t as the reference for our d-FPC controller. We define an error signal as the distance of the contact point from the reference point:

$$e_{i,t} = \hat{s}_i - s_t,\ i = c, ..., T \tag{4.1}$$

where $\hat{s}_i$ is the predicted stem location for a sequence of planned robot movements. We formulate our d-FPC over the error signal as follows:

$$a_{t,res} = -\sum_{i=c:T} (k_{p_i} \times e_{i,t} + k_{d_i} \times \dot{e}_{i,t}) \tag{4.2}$$

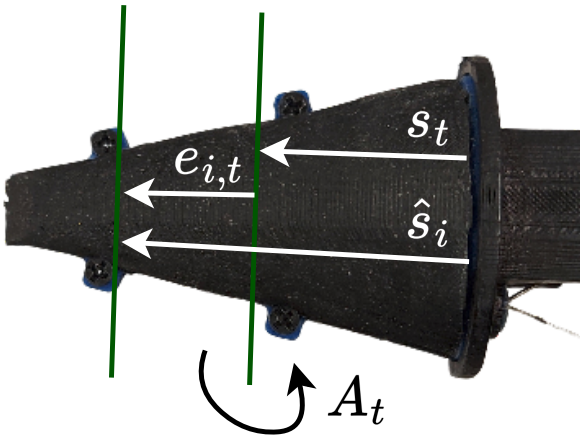


Figure 4.6: Variables involved in the d-FPC control action. s_t is the stem location at time t, $\hat{s}_i$ is the predicted stem location for a sequence of planned movements, $e_{i,t}$ is the difference of the two and, lastly, A_t is the control action generated: the angular velocity associated to a rotation around the gripper's yaw axis.

where $a_{t,res}$ is the residual action value to be added to the reference trajectory $a_{t,ref}$ to generate the control action A_t. A_t is a rotational velocity around the contact line axis. Fig.4.1 (green box) shows the schematic of the d-FPC. The generated control output is a rotational velocity proportional to the distance of the stem from the reference line. The derivative term avoids overshooting and having large instant rotations.

4.5. Benchmark PD control tactile servoing system

To compare the performances of the proposed d-FPC, a PD control-based tactile servoing system will be used with the simplified control scheme depicted in Fig.4.7. This benchmark control system does not rely on the predictions produced by TFM, it simply detects the current position of the stem and if it differs from the reference one, it generates a corrective action to counteract the displacement of the stem on the membrane. For this reason it is a *reactive* control system, as it only takes action when an event has already occurred, as opposed to to the previous approach, which we can define as *proactive* as it acts in advance before the slip occurs.

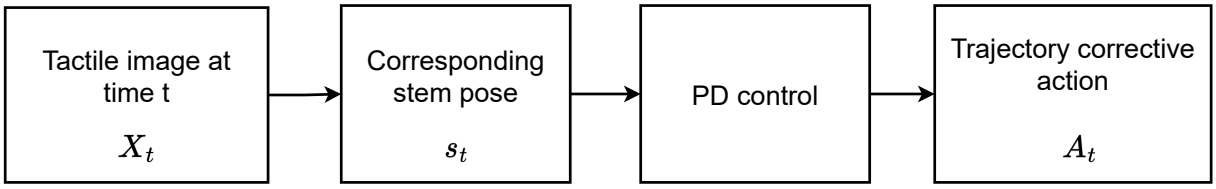


Figure 4.7: Schematic pipeline of the PD control tactile servoing system.

In Figure 4.8 a comparison between the baseline PD controller and d-FPC in the case of a linear push on a single stem. It can be seen that, after the contact with the stem at time $t = 0.85s$, d-FPC is more effective in keeping the stem in the reference position, in this case 0.5 corresponding to the centreline of the sensor. For further discussion of the results of the proposed control strategy and comparison with the basic benchmark controller, please refer to Chapter 6.

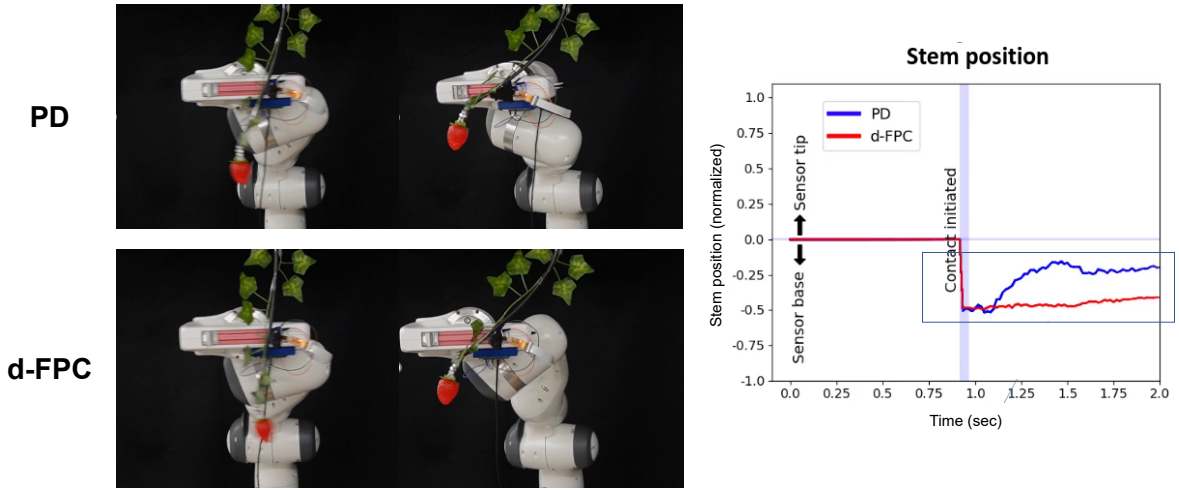


Figure 4.8: PD control vs d-FPC.

5 | Experimental Setup

5.1. The Tactile Finger

Various types of tactile sensors are discussed in the literature, including in [66]. For the cluster manipulation task, a customised camera-based tactile sensor for pushing strawberries similar to Tactip [62] was designed and developed. The idea was to make a sensor that could fit nimbly between the stems of the strawberries but thin and soft enough not to damage them. This sensor has a camera and a LED light looking at a deformable membrane with embedded white markers (Fig. 5.1). The applied pressure on the sensor yields a deformation that is captured by the camera.

The Tactile Finger has a half-conic geometry and a tapered tip (shown in Fig.5.1) designed to allow for easier penetration among stems and fruits, providing valuable tactile feedback. The deformable membrane of the sensor is 3D-printed and dot features are printed with a linear pattern on its conic inner surface. Changes in the marker pattern resulting from contact forces provide information about contact force value, geometry, and location, as shown in Fig. 5.2. The camera, which is located on the sensor base, and the LED, used for illuminating the markers, are powered by an onboard Raspberry Pi, and tactile images are transmitted at a frequency of 60 Hz. The sensor is mounted on a Franka Emika gripper, providing an effective and versatile tool for physical interaction in a range of applications.

5.2. Dataset Collection

We have collected the data from a series of strawberry-pushing tasks in 3-D. The pushing dataset includes data for single strawberry pushing and pushing a cluster of strawberries. To simulate the table-top strawberry growing scenario, we attached each plastic strawberry to a thin wire that makes a nonlinear elastic behaviour similar to those usually observed in tabletop-grown strawberries. To simulate realistic tactile feedback and movements similar to real ones during push actions , we added knots on the stalk of each strawberry (Fig.5.4) and injected silicone to increase their weight (each strawberry weighs c. 20 g to 30 g).

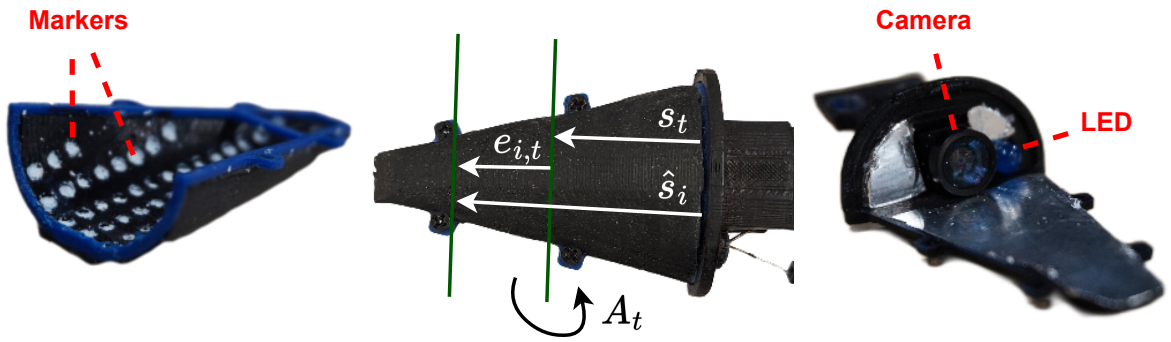


Figure 5.1: The tactile finger design features a deformable half-conic membrane with an integrated miniature camera and LED light. The initial contact line with the stem is considered as the reference line. We predict the location of the stem line $\hat{s}$ within the prediction window $c, ..., T-1$. The robot rotates around the stem contact line to counteract the predicted stem displacement with an action A_t.

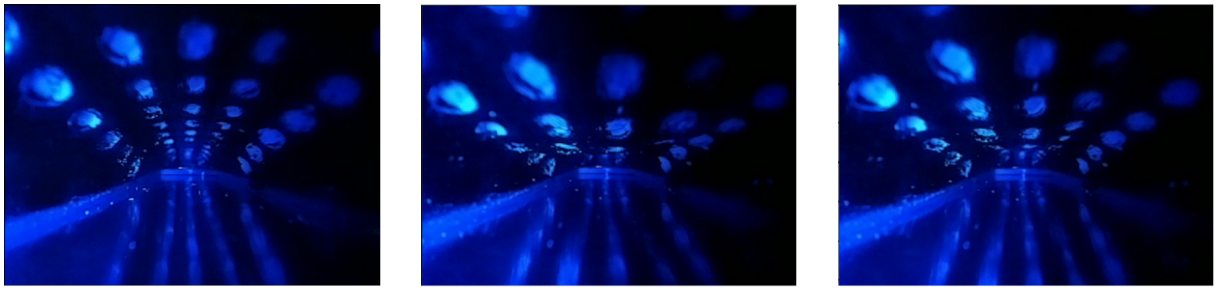


Figure 5.2: From left to right: the camera readings of the tactile finger at rest and when forces are applied to the membrane near the base and between the base and middle point, respectively.

We generate the pushing trajectories for the training data collection phase by two methods: first by Pilz industrial motion planner by specifying initial and target robot poses, and second by defining a minimum time reference trajectory using the robot's Cartesian velocity controller. We use the second method to be able to regenerate comparably similar trajectories in test time, as opposed to the first case where trajectories are generated by the motion planning library. Trajectories include linear and circular motion patterns to

perform the pushing tasks. Arc trajectories were used to collect more tactile-conditioned robot movements, where the finger followed the motion of the pushed stem/strawberry (Fig.5.5). These pushes started at a position p_0 and orientation q_0, followed an arc trajectory, and ended at a final position p_f with a value of z coordinate larger than initial position (conventions in Fig.5.3). The final orientation q_f is selected to maintain contact with the elements pushed. During linear pushes, on the other end, the difference between p_f and p_0 is just in y values, the axis along which we perform the pushes (Fig.5.4). The pushing actions were performed from right to left and vice versa, and they involved single or multiple stems, generating greater deformations on the membrane.

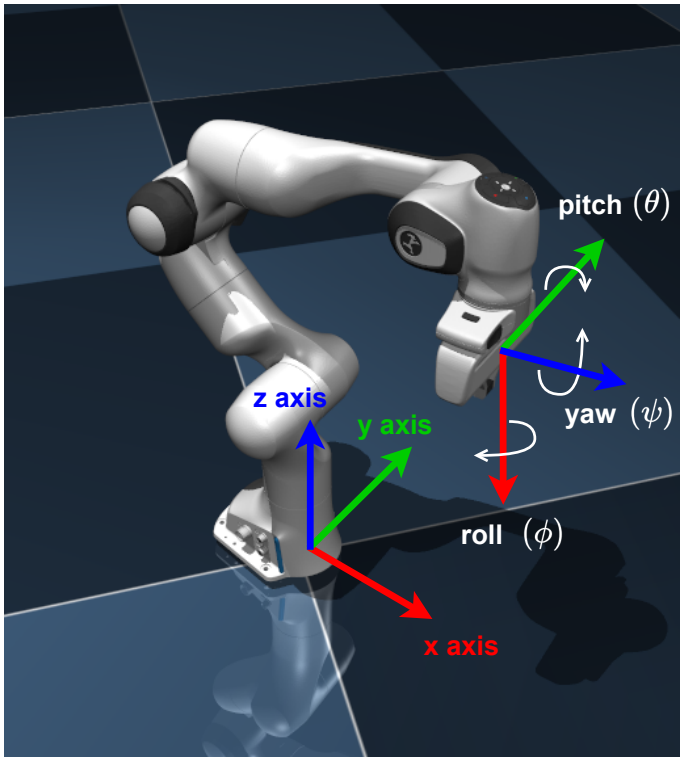


Figure 5.3: The Conventions adopted for the dataset collection highlighted through the base frame and the end-effector's frame.

We collected a total of 430 mixed linear/circular motion tasks containing (i) tactile images from the Finger at 60 Hz and (ii) robot state data sampled at 1000 Hz, representing the position and orientation of the end effector in the planned trajectory. These readings were synchronised using the ROS *ApproximateTime* policy and fed into the tactile forward model both in training and test times.

Considering the robot's motion, slip occurred mainly on the width and length of the finger but could also happen in other directions depending on the motion of the stems during the pushing actions. These cases are researched and the most significant to make the

TFM learn the complex dynamics of these interactions.

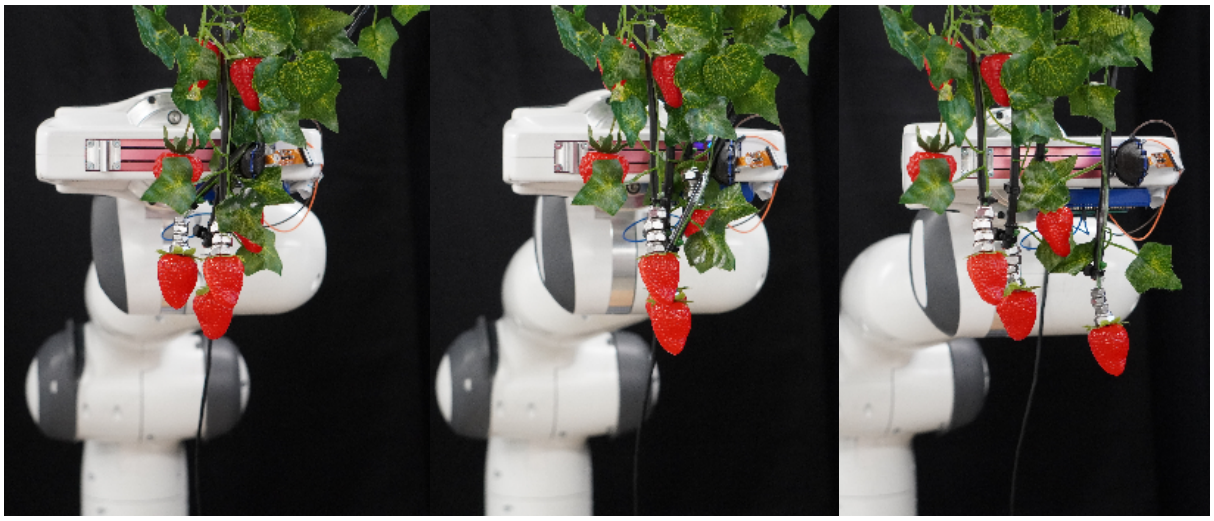


Figure 5.4: An example of linear push performed for data collection. The Franka Emika robotic arm is pushing a cluster of strawberries from right to left with the contact area corresponding to the sensor membrane.

Figure 5.5: An example of circular push performed for data collection. The Franka Emika robotic arm is pushing a cluster of strawberries from right to left with the contact area corresponding to the sensor membrane.

5.3. Model Training and Offline Testing

The collected data set containing samples of robot pushing strawberries, single and cluster, in 3D space was used to train the Tactile Forward Model.
Data were pre-processed before being loaded and passed to the model. Tactile images and actions were associated with the respective instants of time and were subsequently converted in *torch* tensors, since all the training and testing loops were implemented in PyTorch.
TFM was trained for five hundred epochs using Mean Absolute Error (MAE) as loss function (learning curve in Fig.5.6), *Adam* as optimizer and a learning rate equal to $1e-4$. Gradient clipping technique was adopted to avoid issues such as expldoding gradients and vanishing gradients.
To speed up the process, the training was carried out on a 48 GigaByte NVIDIA GPU.
Of the 430 samples of the dataset collected, 130 samples were used for the offline testing, these samples contain pushing actions and tactile readings "unseen" by the model, since they were not in the training part of the data, therefore they represented an additional challenge but a correct yardstick for the evaluation of the performance of the model. Of the 300 samples destined for training, the 10% (30 samples), was randomly selected for the model validation at the end of each training epoch.

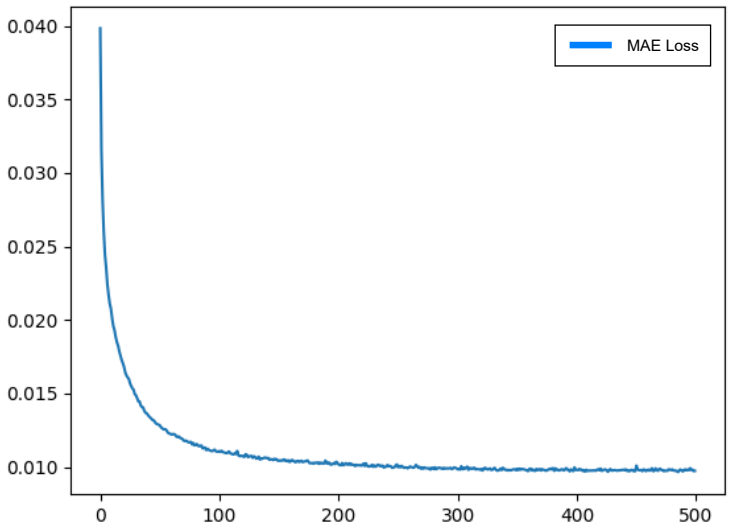


Figure 5.6: TFM training loss curve in five hundred epochs.

6 | Results and Discussion

We test the performance of our proposed control pipeline in real-time on pushing tasks of strawberry stems and compare the performance with a baseline controller and an open-loop system. The tactile sensor is mounted on Franka Emika robot connected to a PC with Intel® Core™ i7-8700K CPU @ 3.70GHz × 12 and 64GB RAM running Ubuntu 20.04 and ROS Noetic. Test manipulation tasks consist of performing pushing trajectories with linear and circular motion patterns using the robot's Cartesian velocity controller.

Performance metrics include: (I) Stem max displacement and (II) the number of stem slip instances on the sensor surface. If we denote stem location at time t by s_i where $i \in (0, 1, ..., T)$ for a pushing trial, metric (I) is defined as the absolute value of the difference of maximum and minimum stem location in a trial $|max(s_i) - min(s_i)| : i = 1, ..., T$. Metric (II) is defined as the number of time steps where the differential values $\dot{s}_i$ were larger than threshold γ. While metric (I) shows full stem displacement, metric (II) shows the stem's sudden large motion instances or slippage on the sensor surface. We also present the area under the curve of stem displacement and generated action. We repeat each test case 5 times and present the mean and standard deviation of the metric values. Overall we conducted 100 test-pushing trials.

To evaluate the effectiveness of d-FPC for pushing control, we compare the control performance with a PD control-based tactile servoing system as the baseline model. Both models' results are presented against the open-loop system with a pre-specified reference trajectory.

In this work, we utilise a minimum-time reference trajectory (such as bang-bang) for the open-loop system, although any desired reference trajectory can be used. To make valid comparisons among trials, we consider three initial contact zones for the stem including **Zone-1** where the contact point is between the middle and tip of the sensor, **Zone-2** has the contact point between the middle and base of the sensor, and **Zone-3** where the contact point is close to sensor centre line (Fig.6.1). Since the tactile sensor has varying deformation limits across its conic axis we compare the trials with corresponding initial contact zones together.

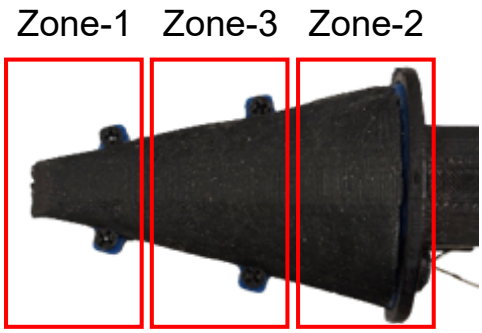


Figure 6.1: The three initial contact zones that we consider for our comparisons. **Zone-1**: contact between the middle and the tip of the sensor. **Zone-2**: contact between the middle and the base of the sensor. **Zone-3**: contact point close to the centre line.

6.1. Without control: slip and failure cases

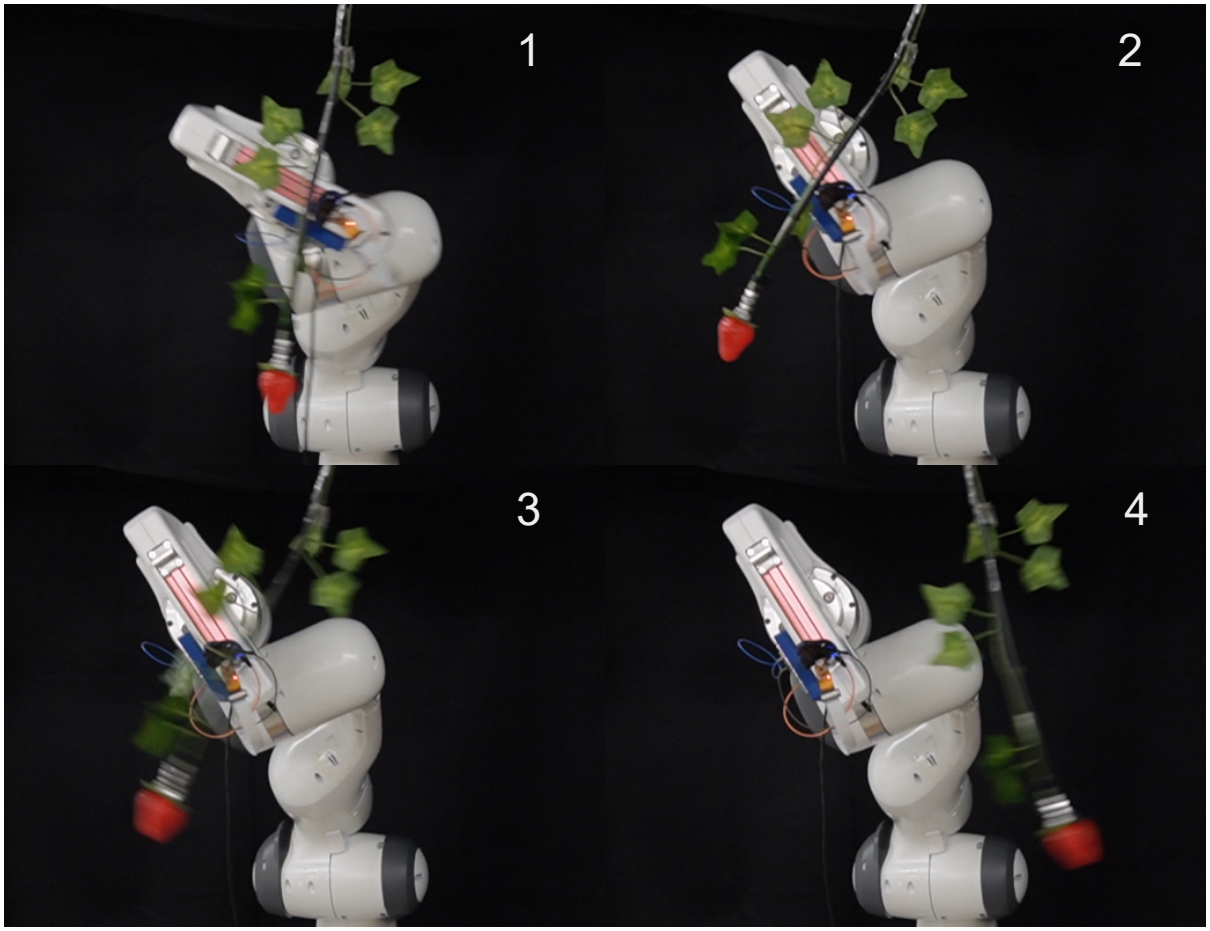


Figure 6.2: Failure example, during the push the stem begins to slip towards the tip of the sensor until it completely loses contact with it.

The objective of the control system is to avoid cases in which the stem slips on the sensor surface, which can lead to drastic cases of loss of contact such as the one depicted in figure (Fig.6.2). Without any wrist control, any possible response to stem slipping is completely absent; the manipulator simply executes the planned trajectory regardless of the effects resulting from the interaction with the element being pushed.

6.2. Single strawberry pushing

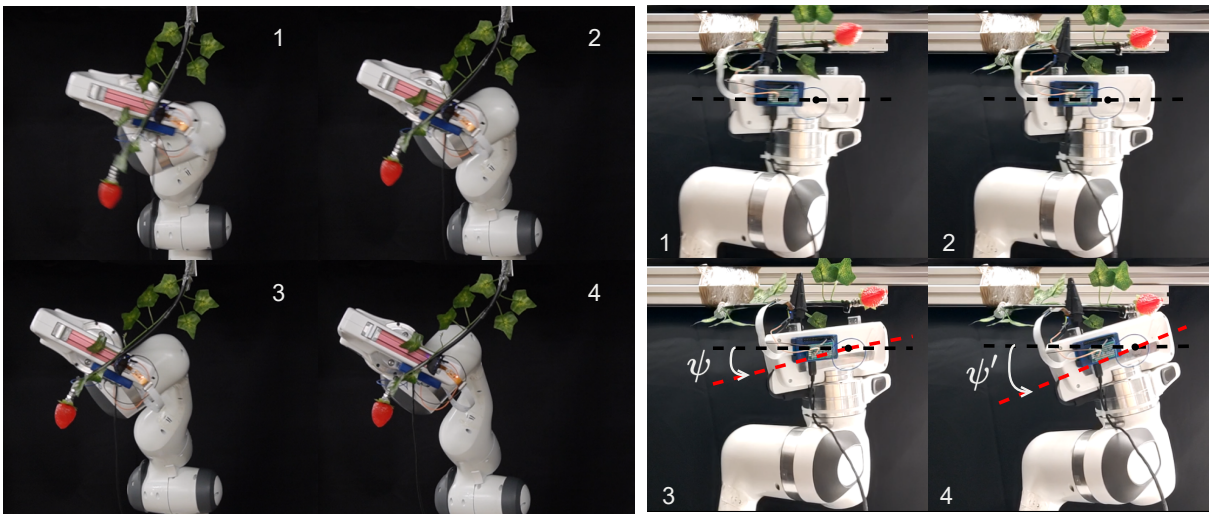


Figure 6.3: d-FPC in action in pushing a single stem following a circular trajectory. When the stem starts to slip from the base (Zone-2) of the sensor towards the tip, the controller takes action by inducing a rotation around the yaw axis to keep the stem in the same position.

To begin with, we conducted tests by pushing a single stem. The results will tell us whether the approach works in situations where other stems and natural elements do not disturb the action of moving a single stem and do not contribute to changing the information detected by the tactile sensor.
We conduct a comparison test with a one-degree-of-freedom (DOF) horizontal pushing along the Y-axis of the robot's base frame. Both PD and d-FPC controllers generate control actions for the robot hand's rotation around the contact line to prevent stem slip on the sensor surface. The results are presented in Table 6.1, where test cases are conducted separately for each initial contact zone. Both PD and d-FPC controllers decrease the stem's maximum displacement. We observe that d-FPC outperforms the PD controller for Zone-1 and Zone-3, but PD shows better performance for Zone-2 very close to the sensor base. This is because the sensor has its largest deformation limit in the Base zone, resulting in relatively large initial deformation after making contact, making it difficult

Table 6.1: Control performance for the PD and d-FPC in pushing a single strawberry along a linear trajectory.

Model	Contact zone	Stem max disp.	Stem slip instances	Disp. integral	Action integral	Comp. time (ms)
Open-loop	1	0.80 ± 0.2	31.23 ± 4.3	0.83 ± 0.1	-	-
	2	1.35 ± 0.2	50.19 ± 5.7	0.91 ± 0.1	-	-
	3	0.91 ± 0.1	39.83 ± 3.2	0.86 ± 0.2	-	-
PD	1	0.65 ± 0.1	27.2 ± 6.5	0.75 ± 0.1	2.93 ± 0.7	18.73 ± 2
	2	**0.36** ± 0.0	10.2 ± 2.4	0.48 ± 0.0	5.12 ± 3.8	20.30 ± 1
	3	0.63 ± 0.1	24.2 ± 1.6	0.47 ± 0.1	9.73 ± 5.4	19.73 ± 1
d-FPC	1	**0.20** ± 0.0	**5.0** ± 1.2	**0.12** ± 0.0	3.74 ± 0.8	60.49 ± 6
	2	0.43 ± 0.0	**7.2** ± 0.7	**0.18** ± 0.0	4.27 ± 1.2	55.02 ± 2
	3	**0.25** ± 0.1	**6.0** ± 0.6	**0.09** ± 0.0	4.57 ± 2.4	58.54 ± 3

Table 6.2: Comparison of the controllers in linear and circular pushing trajectories (* integral is the integral of the * magnitude.).

Model	Robot trajectory	Stem max disp.	Stem slip instances	Disp. integral	Action integral
Open-loop	Linear	1.21 ± 0.18	44.38 ± 10.3	0.88 ± 0.4	-
	Circular	1.35 ± 0.46	48.18 ± 5.2	1.02 ± 0.5	-
PD	Linear	0.58 ± 0.21	25.53 ± 4.2	0.63 ± 0.1	5.39 ± 6.2
	Circular	1.20 ± 0.01	17.6 ± 2.0	0.44 ± 0.0	9.89 ± 0.8
d-FPC	Linear	**0.29** ± 0.04	**8.11** ± 1.4	**0.13** ± 0.0	4.49 ± 2.5
	Circular	**0.54** ± 0.05	**5.0** ± 1.5	**0.22** ± 0.0	6.66 ± 0.8

for TFM to predict future stem states. The prediction of the error signal helps d-FPC to have more reaction time than PD.

We find that d-FPC is the most effective controller to reduce the number of stem slip instances, with the smallest area under the curve of displacement compared to the PD controller. We also present the computation time to show the relative computation complexity of each system. Since d-FPC has two stacked deep models, the computation time is larger than the PD controller.

To compare the performance of different controllers in a qualitative manner, we present the stem location obtained in two trials (shown in Fig. 6.4): Trial-1, where the stem-finger initial contact point is in Zone-1, is shown with solid lines, and Trial-2, with the contact point in Zone-2, is shown with a dashed line. Our results show that d-FPC outperforms PD controller and open loop in maintaining the stem contact, resulting in the smallest displacement of the stem. Furthermore, Fig. 6.4 shows the control actions generated by each controller. We observe that d-FPC generates actions of larger magnitude in Trial-1

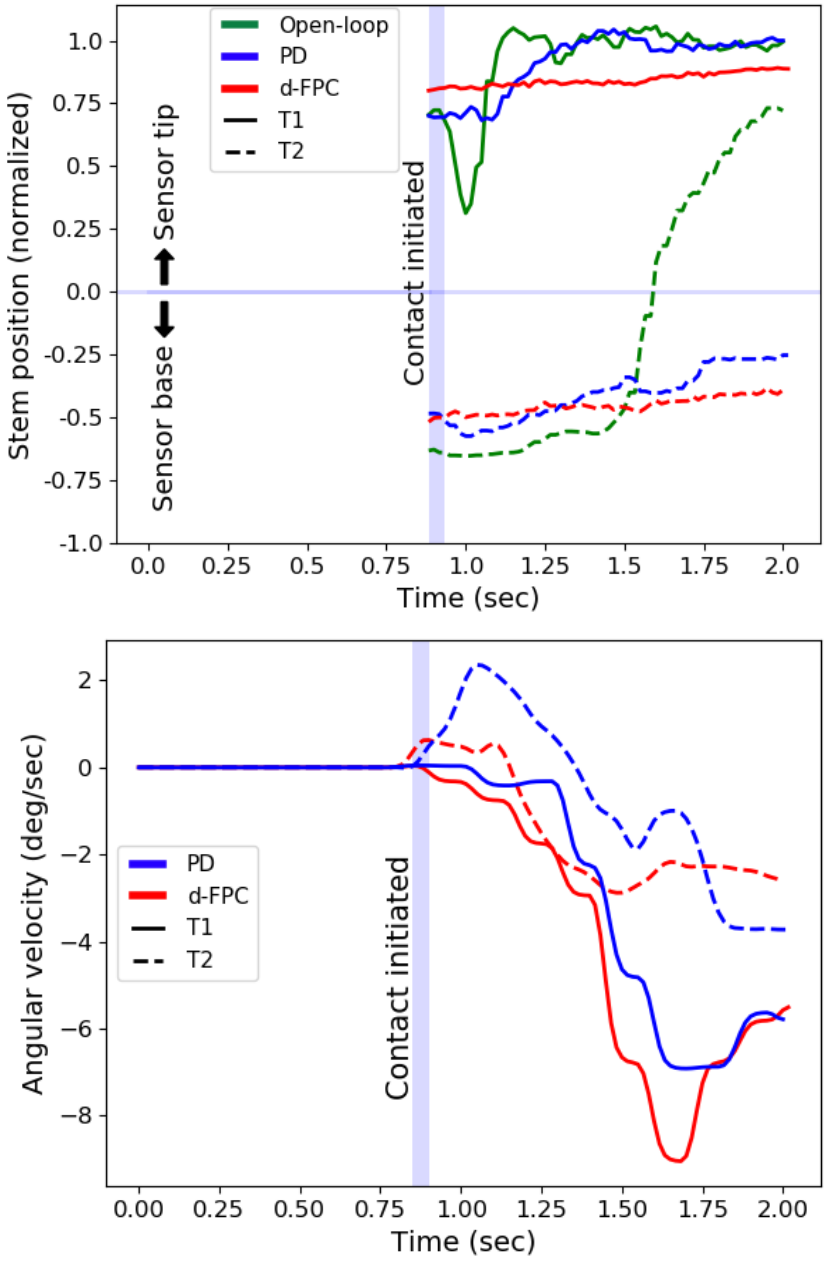


Figure 6.4: Comparison of control performance between d-FPC and open loop, as well as PD controller, in maintaining the location of the stem constant on the finger surface (Trial-1 (T1) solid and Trial-2 (T2) dashed lines) (a) At time 0.85 s, the stem makes contact with the tactile finger and the controllers activate. The graph shows that d-FPC can maintain the stem contact point on the tactile finger during the pushing action, while the open loop result for the trial where the contact is closer to the sensor base shows the stem moving out of the tactile finger surface. (b) The magnitude of the control input shows d-FPC provides larger wrist rotation to avoid stem contact displacement.

Table 6.3: Controller and open loop performances for Pushing a cluster of strawberries.

Model	Robot trajectory	Stem max disp.	Stem slip instances	Disp. integral
Open-loop	Linear	1.43 ± 0.30	49.33 ± 15.64	1.39 ± 0.33
	Circular	1.29 ± 0.67	47.98 ± 6.33	1.19 ± 0.23
PD	Linear	0.79 ± 0.21	29.4 ± 6.52	0.66 ± 0.21
	Circular	1.14 ± 0.23	20.5 ± 2.69	0.56 ± 0.84
d-FPC	Linear	**0.31** ± 0.08	**17.1** ± 2.39	0.25 ± 0.03
	Circular	**0.61** ± 0.11	**9.5** ± 4.58	0.27 ± 0.18

because the likelihood of losing the stem in Zone-1 (namely closer to the tip) is larger than in Zone-2. In Trial-2, the magnitude of d-FPC and PD controller actions is similar since the contact between the stem and sensor membrane is tighter due to a larger deformation of the sensor closer to the sensor base.

We test the performance of the systems in a three DOF task with a bang-bang reference for translation along Y, Z, and rotation W_x of Cartesian velocity space. This is a more challenging task because the robot wrist will rotate 45 degrees along the pushing trajectory which causes larger deformation of the stem and more slip instances (Fig.6.3). Based on Table 6.2 d-FPC is the most effective controller in decreasing the stem displacement and slip instances. PD has a smaller improvement in max displacement for the circular motion than the linear motion compared to the open-loop system. This indicates that not having enough reaction time in this task can lead to failure in achieving the control objective.

6.3. Cluster of strawberries pushing

We test the generalisation performance of the pushing controller when pushing a stem in a cluster of strawberries (shown in Fig.6.5). In this task additional to the target stem, other stems, leaves, or strawberries come into contact with the sensor which makes both tactile prediction and control more challenging. Table 6.3 shows the results for pushing a stem in a cluster. Although the control performance of PD and d-FPC degrades compared to pushing an isolated stem, both systems improve the performance metrics relative to the open-loop system.

Fig.6.6 shows cluster pushing results for sample trials of linear and circular pushing trajectories. For the linear push, PD has slight improvement compared to the open-loop system but d-FPC reduces stem displacement more effectively. For the circular push, while the open-loop system loses contact with the stem because of large stem slippage in the last part of the trial, both PD and d-FPC reduce the stem displacement to avoid large slips.

d-FPC keeps the displacement more bounded relative to the PD controller does.

Figure 6.5: d-FPC in action in a strawberries cluster configuration. The control action is calculated on the first stem with which the sensor makes contact.

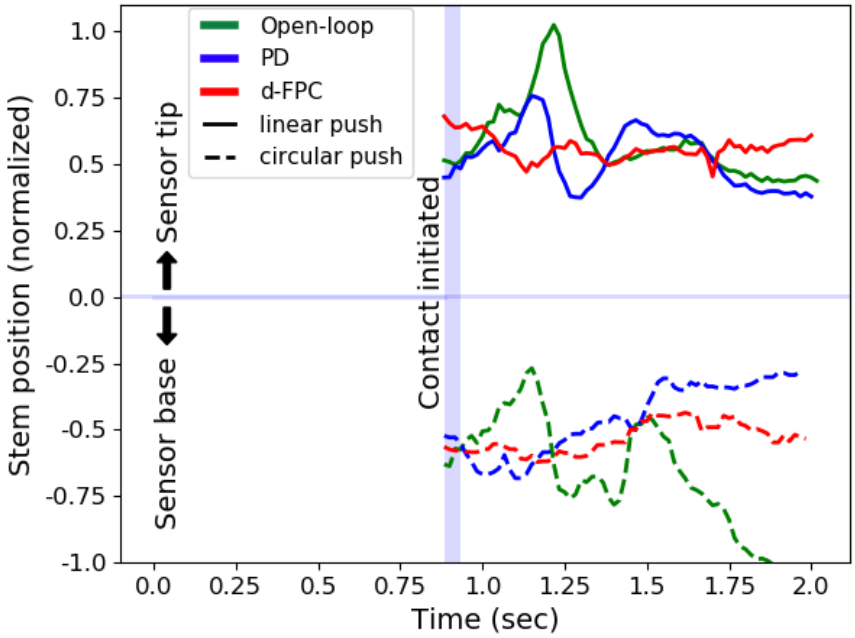


Figure 6.6: Strawberry cluster pushing results. We can clearly see how, even in this most complicated case where other stems interfere with the target one, d-FPC continues to have the best performance, with more precise control of stem position around the initial contact zone and less displacement and oscillation throughout the range in which the pushing action occurs.

6.4. Other Results

Some further results are shown here for singular stem (Fig.6.7) and cluster (Fig.6.8) pushes. In all cases, the action exerted by d-FPC is more effective than that of simple PD control, both in keeping the stem as close as possible to its initial position at the moment of contact until the end of the push and in keeping stem displacement bounded throughout the duration of the pushing action.

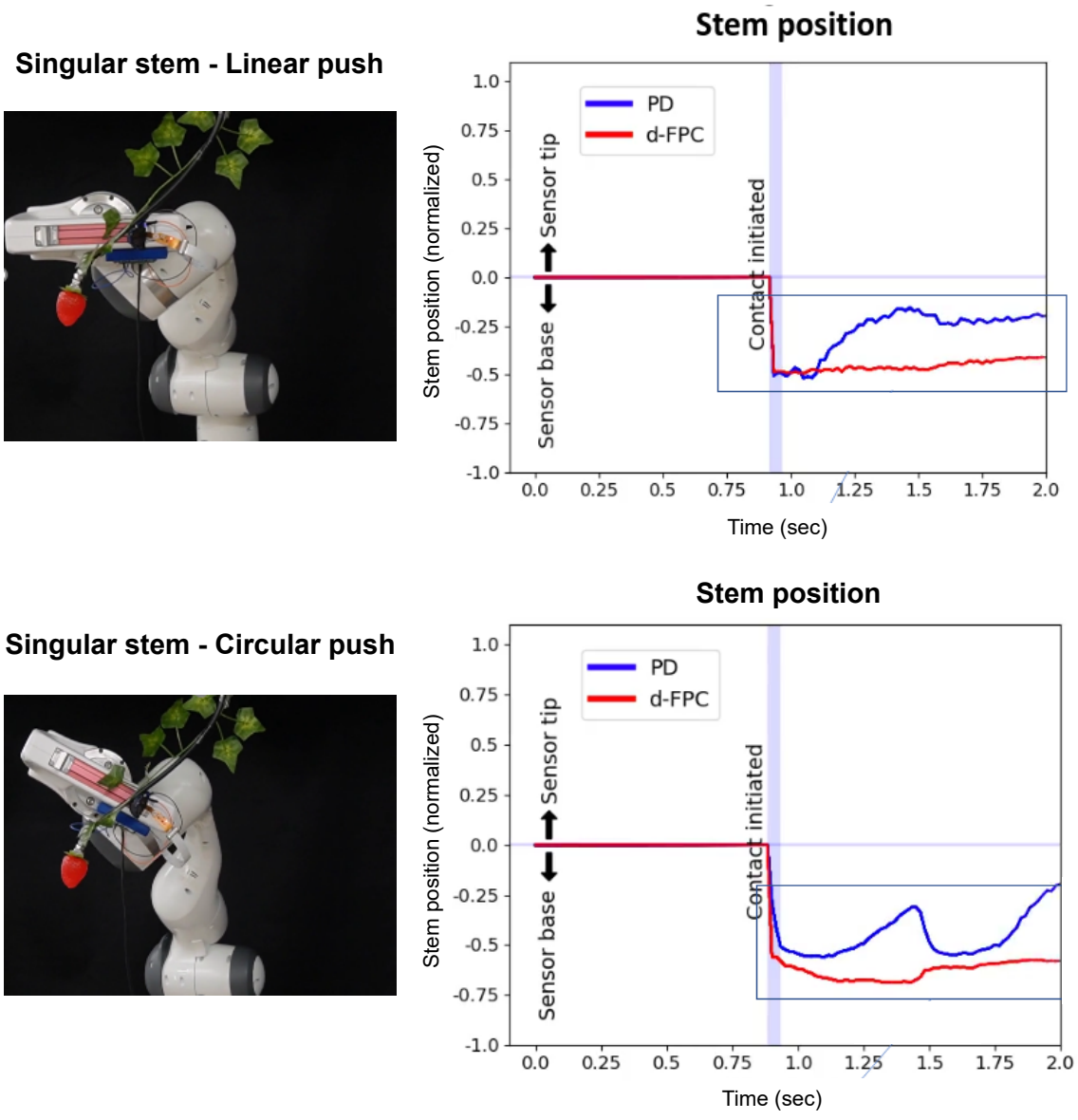


Figure 6.7: Additional singular stem pushing trials, linear and circualr push respectively. Thanks to d-FPC stem displacements remains bounded compared to simple PD control.

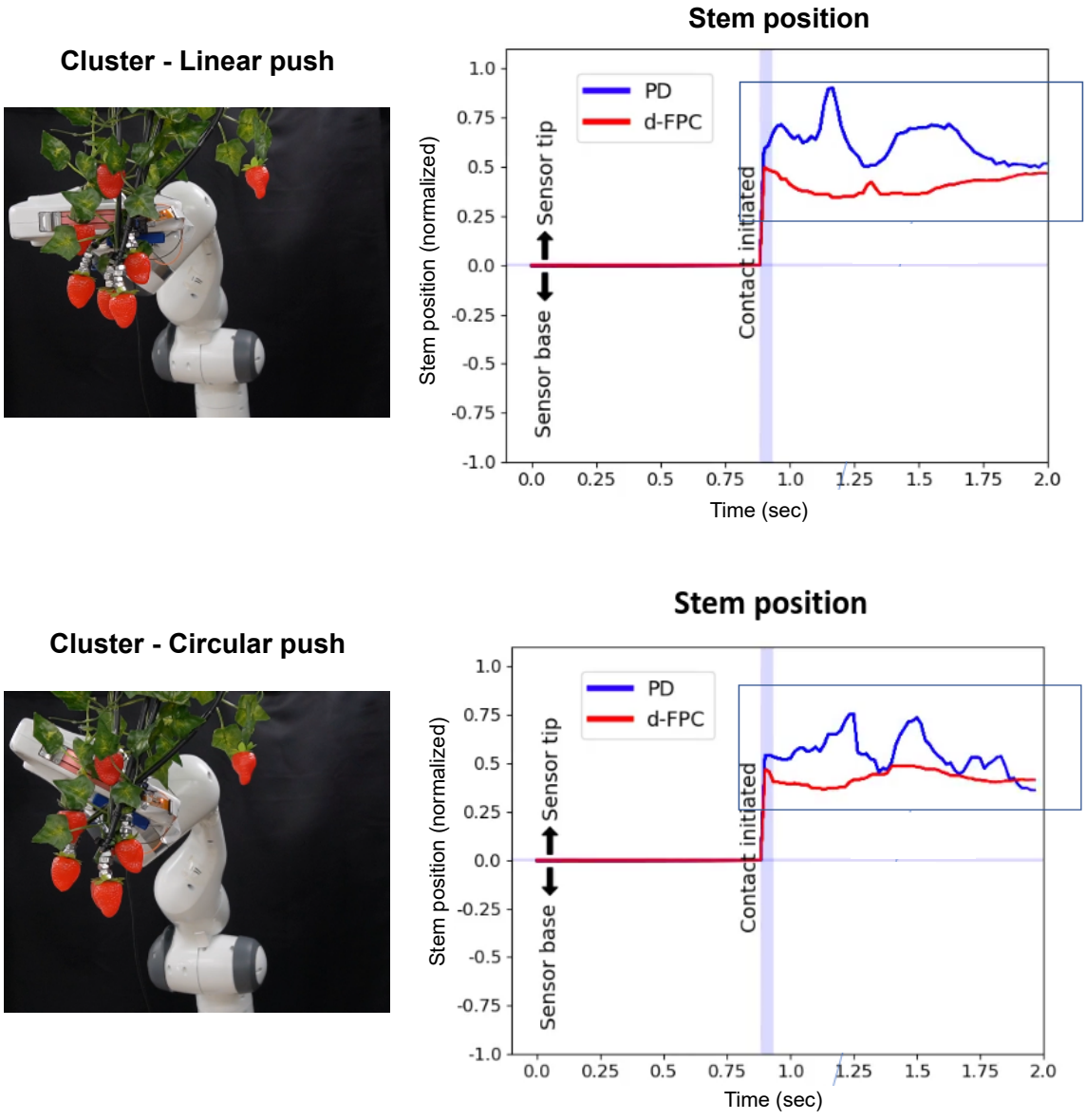


Figure 6.8: Additional cluster pushing trials, linear and circualr push respectively. Thanks to d-FPC the displacements of the stem pushed in the cluster remains bounded compared to simple PD control.

7 | Conclusions and future developments

The increase in demand for agricultural products due to the growth of the world's population combined with the continuous decrease in available human labour in the fields, have made it essential to develop plans to automate agricultural tasks such as fruit picking using robotic systems.
However, accomplishing this task has proven complicated [40], as it involves an unstructured environment (i) and the handling of soft products (ii) that require a radically different approach than the methods imparted to robots intended for industrial production lines.
The challenge of Physical Robot Interaction with natural elements to achieve a free reach-to-peak trajectory is important to address to minimise time and make harvesting more efficient, it is also stimulating because it allows the development and implementation of new control approaches from which the entire branch of robotic manipulation can benefit.

This thesis presented a novel deep Tactile Forward Model, an approach that could considerably change the way manipulation is understood so far.
As we know, when a robotic arm moves in an unstructured environment, contact with unprogrammed surrounding elements may be frequent and unavoidable, likewise in push or grasp-and-move operations, there are variables at play that can lead to unintended results, such as the loss of contact with the pushed object or the slipping of the grasped object.
Contact is unpredictable, but with our approach we can predict how the object will move from time to time on the robot's hand, based on past observations (learnt by the model during training), the recent past (input of our predictive model) and on the rich tactile-visual information provided by the sensor.
The Tactile Forward Model was integrated in a novel deep Functional Predictive Control (d-FPC) framework to control the contact location of a strawberry stem on our tactile finger. Our proposed control strategy leverages the Tactile Forward Model for generating action-conditioned tactile predictions and a convolutional neural network model (CLM)

converting the tactile images to contact location. We demonstrated the effectiveness of our approach through a series of experiments with a Franka Emika robot and a customised tactile finger, showing that our model can learn complex contact behaviours and generate actions to control the movements of flexible objects to keep them stable, e.g. pushing a cluster of strawberries.

Apart from our specific application, this method can be generalised to all those contexts where it is not possible to model interaction with the environment and where the fragility and delicacy of the objects we have to manipulate, like soft fruits, does not allow for abrupt, force-based approaches.

This work leads to a new frontier of manipulator controllers that, by making predictions about the movements of the objects they interact with, can act in advance in a more reasoned and intelligent way by leveraging trajectory adaptation methods in a more human-like behaviour.

Overall, this work highlights the potential of deep learning-based approaches in addressing the challenges of tactile sensing-based manipulation tasks and lays the foundation for future research in this field.

Bibliography

[1] University of lincoln. URL `https://www.lincoln.ac.uk/`.

[2] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *Proceedings of the IEEE international conference on computer vision*, pages 37–45, 2015.

[3] V. E. Arriola-Rios, P. Guler, F. Ficuciello, D. Kragic, B. Siciliano, and J. L. Wyatt. Modeling of deformable objects for robotic manipulation: A tutorial and review. *Frontiers in Robotics and AI*, Volume 7:page 82, 2020.

[4] C. W. Bac, E. J. Van Henten, J. Hemming, and Y. Edan. Harvesting robots for high-value crops: State-of-the-art review and challenges ahead. *Journal of Field Robotics*, Volume 31(Number 6):pages 888–911, 2014.

[5] R. Barth, J. Hemming, and E. J. van Henten. Design of an eye-in-hand sensing and servo control framework for harvesting robotics in dense vegetation. *Biosystems Engineering*, Volume 146:pages 71–84, 2016.

[6] A. J. Bastian. Learning to predict the future: the cerebellum adapts feedforward movement control. *Current opinion in neurobiology*, Volume 16(Number 6):pages 645–649, 2006.

[7] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[8] P. Boonvisut and M. C. Çavuşoğlu. Estimation of soft tissue mechanical parameters from robotic manipulation data. *IEEE/ASME Transactions on Mechatronics*, Volume 18(Number 5):pages 1602–1611, 2012.

[9] M. Boryga, A. Graboś, P. Kołodziej, K. Gołacki, and Z. Stropek. Trajectory planning with obstacles on the example of tomato harvest. *Agriculture and Agricultural Science Procedia*, Volume 7:pages 27–34, 2015.

[10] J. S. Bruner and C. C. Goodman. Value and need as organizing factors in perception. *The journal of abnormal and social psychology*, Volume 42(Number 1):page 33, 1947.

[11] O. Calicioglu, A. Flammini, S. Bracco, L. Bellù, and R. Sims. The future challenges of food and agriculture: An integrated analysis of trends and solutions. *Sustainability*, Volume 11(Number 1):page 222, 2019.

[12] V. Cortés, C. Blanes, J. Blasco, C. Ortiz, N. Aleixos, M. Mellado, S. Cubero, and P. Talens. Integration of simultaneous tactile sensing and visible and near-infrared reflectance spectroscopy in a robot gripper for mango quality assessment. *Biosystems Engineering*, Volume 162:pages 112–123, 2017.

[13] A. De Preter, J. Anthonis, and J. De Baerdemaeker. Development of a robot for harvesting strawberries. *IFAC-PapersOnLine*, Volume 51(Number 17):pages 14–19, 2018.

[14] E. Denton and R. Fergus. Stochastic video generation with a learned prior. In *International conference on machine learning*, pages 1174–1183. PMLR, 2018.

[15] L. M. Dischinger, M. Cravetz, J. Dawes, C. Votzke, C. VanAtter, M. L. Johnston, C. M. Grimm, and J. R. Davidson. Towards intelligent fruit picking with in-hand sensing. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3285–3291. IEEE, 2021.

[16] A. Doumanoglou, A. Kargakos, T.-K. Kim, and S. Malassiotis. Autonomous active recognition and unfolding of clothes using random decision forests and probabilistic planning. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 987–993. IEEE, 2014.

[17] S. Fountas, N. Mylonas, I. Malounas, E. Rodias, C. Hellmann Santos, and E. Pekkeriet. Agricultural robotics for field operations. *Sensors*, Volume 20(Number 9):page 2672, 2020.

[18] Z. Gao, C. Tan, L. Wu, and S. Z. Li. Simvp: Simpler yet better video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3170–3180, 2022.

[19] R. Haber, J. A. Rossiter, and K. Zabet. An alternative for pid control: predictive functional control-a tutorial. In *2016 American Control Conference (ACC)*, pages 6935–6940. IEEE, 2016.

[20] S. M. Hannum. Potential impact of strawberries on human health: a review of the science. *Critical reviews in food science and nutrition*, Volume 44(Number 1):pages 1–17, 2004.

[21] R. Harrell, P. Adsit, T. Pool, and R. Hoffman. The florida robotic grove-lab. *Transactions of the ASAE*, Volume 33(Number 2):pages 391–0399, 1990.

[22] T. Hietaranta and M.-M. Linna. Penetrometric measurement of strawberry fruit firmness: device testing. *HortTechnology*, Volume 9(Number 1):pages 103–105, 1999.

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

[24] F. R. Hogan, J. Ballester, S. Dong, and A. Rodriguez. Tactile dexterity: Manipulation primitives with tactile feedback. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 8863–8869. IEEE, 2020.

[25] D.-A. Huang, V. Ramanathan, D. Mahajan, L. Torresani, M. Paluri, L. Fei-Fei, and J. C. Niebles. What makes a video a video: Analyzing temporal information in video understanding models and datasets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7366–7375, 2018.

[26] R. S. Johansson and J. R. Flanagan. Coding and use of tactile signals from the fingertips in object manipulation tasks. *Nature Reviews Neuroscience*, Volume 10 (Number 5):pages 345–359, 2009.

[27] G. Jukema and R. van der Meer. Arbeidskosten in de akkerbouw en glastuinbouw. *Agri-monitor*, 2009(feb):pages 11–12, 2009.

[28] C. Kampouris, I. Mariolis, G. Peleka, E. Skartados, A. Kargakos, D. Triantafyllou, and S. Malassiotis. Multi-sensorial and explorative recognition of garments and their material properties in unconstrained environment. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1656–1663. IEEE, 2016.

[29] H. Khamis, B. Xia, and S. J. Redmond. Real-time friction estimation for grip force control. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1608–1614. IEEE, 2021.

[30] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Volume 86(Number 11):pages 2278–2324, 1998.

[31] I. Lenz, R. A. Knepper, and A. Saxena. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*, volume 10. Rome, Italy, 2015.

[32] W. Mandil, K. Nazari, et al. Action conditioned tactile prediction: a case study on slip prediction. *arXiv preprint arXiv:2205.09430*, 2022.

[33] S. Mghames, M. Hanheide, and A. Ghalamzan. Interactive movement primitives: Planning to push occluding pieces for fruit picking. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2616–2623. IEEE, 2020.

[34] R. Miall and E. Jenkinson. Functional imaging of changes in cerebellar activity related to learning during a novel eye–hand tracking task. *Experimental Brain Research*, 166:170–183, 2005.

[35] R. C. Miall and D. M. Wolpert. Forward models for physiological motor control. *Neural networks*, Volume 9(Number 8):pages 1265–1279, 1996.

[36] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

[37] K. Nazari, G. Gandolfi, Z. Talebpour, V. Rajendran, P. Rocco, et al. Deep functional predictive control for strawberry cluster manipulation using tactile prediction. *arXiv preprint arXiv:2303.05393*, 2023.

[38] K. Nazari, W. Mandil, and A. M. G. Esfahani. Proactive slip control by learned slip model and trajectory adaptation. In *Conference on Robot Learning*, pages 751–761. PMLR, 2023.

[39] C. Olah. Understanding lstm networks, 2015. URL `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[40] L. F. Oliveira, A. P. Moreira, and M. F. Silva. Advances in agriculture robotics: A state-of-the-art review and challenges ahead. *Robotics*, Volume 10(Number 2):page 52, 2021.

[41] S. Oprea, P. Martinez-Gonzalez, A. Garcia-Garcia, J. A. Castro-Vargas, S. Orts-Escolano, J. Garcia-Rodriguez, and A. Argyros. A review on deep learning techniques for video prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 44(Number 6):pages 2806–2826, 2020.

[42] A. J. Oxenham et al. Helmholtz: From enlightenment to neuroscience. *The Journal of Clinical Investigation*, Volume 121(Number 6):pages 2064–2064, 2011.

[43] S. Parsa, B. Debnath, M. A. Khan, et al. Autonomous strawberry picking robotic system (robofruit). *arXiv preprint arXiv:2301.03947*, 2023.

[44] R. P. Rao and D. H. Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, Volume 2(Number 1):pages 79–87, 1999.

[45] C. W. Roland Johansson. Johansson testing. URL `https://www.youtube.com/watch?v=0LfJ3M3Kn80`.

[46] M. Roser. Employment in agriculture. *Our World in Data*, 2013. https://ourworldindata.org/employment-in-agriculture.

[47] J. A. Rossiter and M. Abdullah. A new paradigm for predictive functional control to enable more consistent tuning. In *2019 American Control Conference (ACC)*, pages 366–371. IEEE, 2019.

[48] S. Saha. A comprehensive guide to convolutional neural networks — the eli5 way, 2018. URL `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53`.

[49] L. Scimeca, P. Maiolino, D. Cardin-Catalan, A. P. del Pobil, A. Morales, and F. Iida. Non-destructive robotic assessment of mango ripeness via multi-point soft haptics. In *2019 international conference on robotics and automation (ICRA)*, pages 1821–1826. IEEE, 2019.

[50] D. SepúLveda, R. Fernández, E. Navas, M. Armada, and P. González-De-Santos. Robotic aubergine harvesting using dual-arm manipulation. *IEEE Access*, Volume 8:pages 121889–121904, 2020.

[51] C. Shi, Z. Zhang, W. Zhang, C. Zhang, and Q. Xu. Learning multiscale temporal–spatial–spectral features via a multipath convolutional lstm neural network for change detection with hyperspectral images. *IEEE Transactions on Geoscience and Remote Sensing*, Volume 60:pages 1–16, 2022.

[52] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, Volume 28, 2015.

[53] A. Silwal, J. R. Davidson, M. Karkee, C. Mo, Q. Zhang, and K. Lewis. Design, integration, and field evaluation of a robotic apple harvester. *Journal of Field Robotics*, Volume 34(Number 6):pages 1140–1159, 2017.

[54] R. Sparrow and M. Howard. Robots in agriculture: prospects, impacts, ethics, and policy. *precision agriculture*, Volume 22:pages 818–833, 2021.

[55] Statista. Global market volume of agricultural robots from 2020 to 2030, 2022. URL `https://www.statista.com/statistics/1290013/agricultural-robot-global-market-unit-volume/`.

[56] Statista. Share of economic sectors in the global gross domestic product (gdp) from 2011 to 2021, 2023. URL `https://www.statista.com/statistics/256563/share-of-economic-sectors-in-the-global-gross-domestic-product/`.

[57] A. Tafuro, B. Debnath, A. M. Zanchettin, and E. A. Ghalamzan. dpmp-deep probabilistic motion planning: A use case in strawberry picking robot. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8675–8681. IEEE, 2022.

[58] G. Tian, J. Zhou, and B. Gu. Slipping detection and control in gripping fruits and vegetables for agricultural robot. *International Journal of Agricultural and Biological Engineering*, Volume 11(Number 4):pages 45–51, 2018.

[59] S. Tian, F. Ebert, D. Jayaraman, M. Mudigonda, C. Finn, R. Calandra, and S. Levine. Manipulation by feel: Touch-based control with deep predictive models. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 818–824. IEEE, 2019.

[60] F. Veiga, H. Van Hoof, J. Peters, and T. Hermans. Stabilizing novel objects by learning to predict tactile slip. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5065–5072. IEEE, 2015.

[61] H. Wang, Q. Zhao, H. Li, and R. Zhao. Polynomial-based smooth trajectory planning for fruit-picking robot manipulator. *Information Processing in Agriculture*, Volume 9(Number 1):pages 112–122, 2022.

[62] B. Ward-Cherrier, N. Pestell, L. Cramphorn, B. Winstone, M. E. Giannaccini, J. Rossiter, and N. F. Lepora. The tactip family: Soft optical tactile sensors with 3d-printed biomimetic morphologies. *Soft robotics*, Volume 5(Number 2):pages 216–227, 2018.

[63] P. Watts, A. Lewis, and B. Nagpal. Economic considerations in industrial robotics. In *Proceedings of the Twenty-third International Machine Tool Design and Research Conference*, pages 527–532. Springer, 1983.

[64] D. M. Wolpert, J. Diedrichsen, and J. R. Flanagan. Principles of sensorimotor learning. *Nature reviews neuroscience*, Volume 12(Number 12):pages 739–751, 2011.

[65] Y. Xiong, Y. Ge, L. Grimstad, and P. J. From. An autonomous strawberry-harvesting

robot: Design, development, integration, and field evaluation. *Journal of Field Robotics*, Volume 37(Number 2):pages 202–224, 2020.

[66] A. Yamaguchi and C. G. Atkeson. Recent progress in tactile sensing and sensors for robotic manipulation: can we turn tactile sensing into vision? *Advanced Robotics*, Volume 33(Number 14):pages 661–673, 2019.

[67] S. Yamamoto, S. Hayashi, H. Yoshida, and K. Kobayashi. Development of a stationary robotic strawberry harvester with a picking mechanism that approaches the target fruit from below. *Japan Agricultural Research Quarterly: JARQ*, Volume 48 (Number 3):pages 261–269, 2014.

[68] H. Yin, A. Varava, and D. Kragic. Modeling, learning, perception, and control methods for deformable object manipulation. *Science Robotics*, Volume 6(Number 54):page eabd8803, 2021.

[69] E. Yoshida, K. Ayusawa, I. G. Ramirez-Alpizar, K. Harada, C. Duriez, and A. Kheddar. Simulation-based optimal motion planning for deformable object. In *2015 IEEE international workshop on advanced robotics and its social impacts (ARSO)*, pages 1–6. IEEE, 2015.

[70] W. Yu, A. Kapusta, J. Tan, C. C. Kemp, G. Turk, and C. K. Liu. Haptic simulation for robot-assisted dressing. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 6044–6051. IEEE, 2017.

[71] W. Yuan, S. Wang, S. Dong, and E. Adelson. Connecting look and feel: Associating the visual and tactile properties of physical materials. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5580–5588, 2017.

[72] Y. Zhang, W. Yuan, Z. Kan, and M. Y. Wang. Towards learning to detect and predict contact events on vision-based tactile sensors. In *Conference on Robot Learning*, pages 1395–1404. PMLR, 2020.

[73] H. Zhou, X. Wang, W. Au, H. Kang, and C. Chen. Intelligent robots for fruit harvesting: Recent developments and future challenges. *Precision Agriculture*, Volume 23(Number 5):pages 1856–1907, 2022.

[74] H. Zhou, H. Kang, X. Wang, W. Au, M. Y. Wang, and C. Chen. Branch interference sensing and handling by tactile enabled robotic apple harvesting. *Agronomy*, Volume 13(Number 2):page 503, 2023.

[75] J. Zhou, L. He, M. Karkee, and Q. Zhang. Analysis of shaking-induced cherry fruit motion and damage. *Biosystems Engineering*, Volume 144:pages 105–114, 2016.

[76] J. Zhu, B. Navarro, P. Fraisse, A. Crosnier, and A. Cherubini. Dual-arm robotic manipulation of flexible cables. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 479–484. IEEE, 2018.

A | Appendix A

```
# PyTorch implemenation of a ConvLSTM Cell for Video Prediction
 class ConvLSTMCell(nn.Module):
    def __init__(self, input_dim, hidden_dim, kernel_size, bias, device):
        super(ConvLSTMCell, self).__init__()
        self.input_dim = input_dim
        self.device = device
        self.hidden_dim = hidden_dim
        self.kernel_size = kernel_size
        self.padding = kernel_size[0] // 2, kernel_size[1] // 2
        self.bias = bias
        self.conv = nn.Conv2d(in_channels=self.input_dim + self.hidden_dim,
            out_channels=4 * self.hidden_dim, kernel_size=self.kernel_size,
            padding=self.padding, bias=self.bias)

    def forward(self, input_tensor, cur_state):
        h_cur, c_cur = cur_state
        combined = torch.cat([input_tensor, h_cur], dim=1)
        combined_conv = self.conv(combined)
        cc_i, cc_f, cc_o, cc_g = combined_conv[:, :self.hidden_dim],
            combined_conv[:, self.hidden_dim:2*self.hidden_dim],\
            combined_conv[:, 2*self.hidden_dim:3*self.hidden_dim],
                combined_conv[:, 3*self.hidden_dim: 4*self.hidden_dim]
        i = torch.sigmoid(cc_i)
        f = torch.sigmoid(cc_f)
        o = torch.sigmoid(cc_o)
        g = torch.tanh(cc_g)
        c_next = f * c_cur + i * g
        h_next = o * torch.tanh(c_next)
        return h_next, c_next

    def init_hidden(self, batch_size, image_size):
```

```
        height, width = image_size
        return (torch.zeros(batch_size, self.hidden_dim, height, width,
            device=self.conv.weight.device),
                torch.zeros(batch_size, self.hidden_dim, height, width,
                    device=self.conv.weight.device))
```

B | Appendix B

```
# TFM Class Declaration
 class TFM(nn.Module):
    def __init__(self, features):
        super(TFM, self).__init__()
        self.features = features
        self.device = features["device"]
        self.context_frames = features["n_past"]
        self.n_future = features["n_future"]
        self.convlstm1 = ConvLSTMCell(input_dim=128, hidden_dim=128,
            kernel_size=(3, 3), bias=True, device=self.device)
        self.convlstm2 = ConvLSTMCell(input_dim=140, hidden_dim=140,
            kernel_size=(3, 3), bias=True, device=self.device)
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=64, kernel_size=5,
            stride=1, padding=2)
        self.conv12 = nn.Conv2d(in_channels=64, out_channels=128,
            kernel_size=5, stride=1, padding=2)
        self.conv2 = nn.Conv2d(in_channels=6, out_channels=3, kernel_size=5,
            stride=1, padding=2)
        self.upconv1 = nn.Conv2d(in_channels=140, out_channels=64,
            kernel_size=5, stride=1, padding=2)
        self.upconv2 = nn.Conv2d(in_channels=64, out_channels=3, kernel_size=5,
            stride=1, padding=2)
        self.maxpool1 = nn.MaxPool2d(2, stride=2)
        self.relu1 = nn.ReLU()
        self.upsample2 = nn.Upsample(scale_factor=2)
        self.tanh = nn.Tanh()
```

C | Appendix C

C.1. Peak Signal-to-Noise Ratio (PSNR)

Peak signal-to-noise ratio (PSNR) is a metric to measure the quality of a reconstructed predicted image with respect to the original one. It is defined as the ratio between the maximum possible power of a signal, the original image, and the power of corrupting noise that affects the predicted one.

```
# PyTorch implementations of Peak Signal-to-Noise Ratio (PSNR)
class PSNR:
    """Peak Signal to Noise Ratio
    img1 and img2 have range [0, 255]"""

    def __init__(self):
        self.name = "PSNR"

    @staticmethod
    def __call__(img1, img2):
        mse = torch.mean((img1 - img2) ** 2)
        return 20 * torch.log10(255.0 / torch.sqrt(mse))
```

C.2. Structural Similarity Index (SSIM)

Structural Similarity Index (SSIM) is a metric to measure the quality of a reconstructed predicted image with respect to the original one. SSIM measures the similarity between two images, a reference one (the original) and the one affected by noises (the prediction).

```
# PyTorch implementations of Structural Similarity Index (SSIM)
class SSIM:
    """Structure Similarity
    img1, img2: [0, 255]"""

    def __init__(self):
        self.name = "SSIM"

    @staticmethod
    def _ssim(img1, img2):
        C1 = (0.01 * 255) ** 2
        C2 = (0.03 * 255) ** 2

        img1 = img1.astype(np.float64)
        img2 = img2.astype(np.float64)
        kernel = cv2.getGaussianKernel(11, 1.5)
        window = np.outer(kernel, kernel.transpose())

        mu1 = cv2.filter2D(img1, -1, window)[5:-5, 5:-5] # valid
        mu2 = cv2.filter2D(img2, -1, window)[5:-5, 5:-5]
        mu1_sq = mu1 ** 2
        mu2_sq = mu2 ** 2
        mu1_mu2 = mu1 * mu2
        sigma1_sq = cv2.filter2D(img1 ** 2, -1, window)[5:-5, 5:-5] - mu1_sq
        sigma2_sq = cv2.filter2D(img2 ** 2, -1, window)[5:-5, 5:-5] - mu2_sq
        sigma12 = cv2.filter2D(img1 * img2, -1, window)[5:-5, 5:-5] - mu1_mu2

        ssim_map = ((2 * mu1_mu2 + C1) * (2 * sigma12 + C2)) / ((mu1_sq +
            mu2_sq + C1) * (sigma1_sq + sigma2_sq + C2))
        return ssim_map.mean()


    @staticmethod
    def __call__(img1, img2):
```

```
        if not img1.shape == img2.shape:
            raise ValueError("Input images must have the same dimensions.")
        if img1.ndim == 2: # Grey or Y-channel image
            return self._ssim(img1, img2)
        elif img1.ndim == 3:
            if img1.shape[2] == 3:
                ssims = []
                for i in range(3):
                    ssims.append(self._ssim(img1, img2))
                return np.array(ssims).mean()
            elif img1.shape[2] == 1:
                return self._ssim(np.squeeze(img1), np.squeeze(img2))
        else:
            raise ValueError("Wrong input image dimensions.")
```

Although a higher PSNR and SSIM generally indicate that the prediction is of higher quality, in some cases it may not. It heavily depends on the range of validity of these metrics. That is why scores are generally presented together with other metrics, e.g. MAE and MSE.

D | Appendix D

Two additional examples of tactile predictions obtained from the Tactile Forward Model. Five frames, providing the context, are given and, as results, ten future frames are obtained.

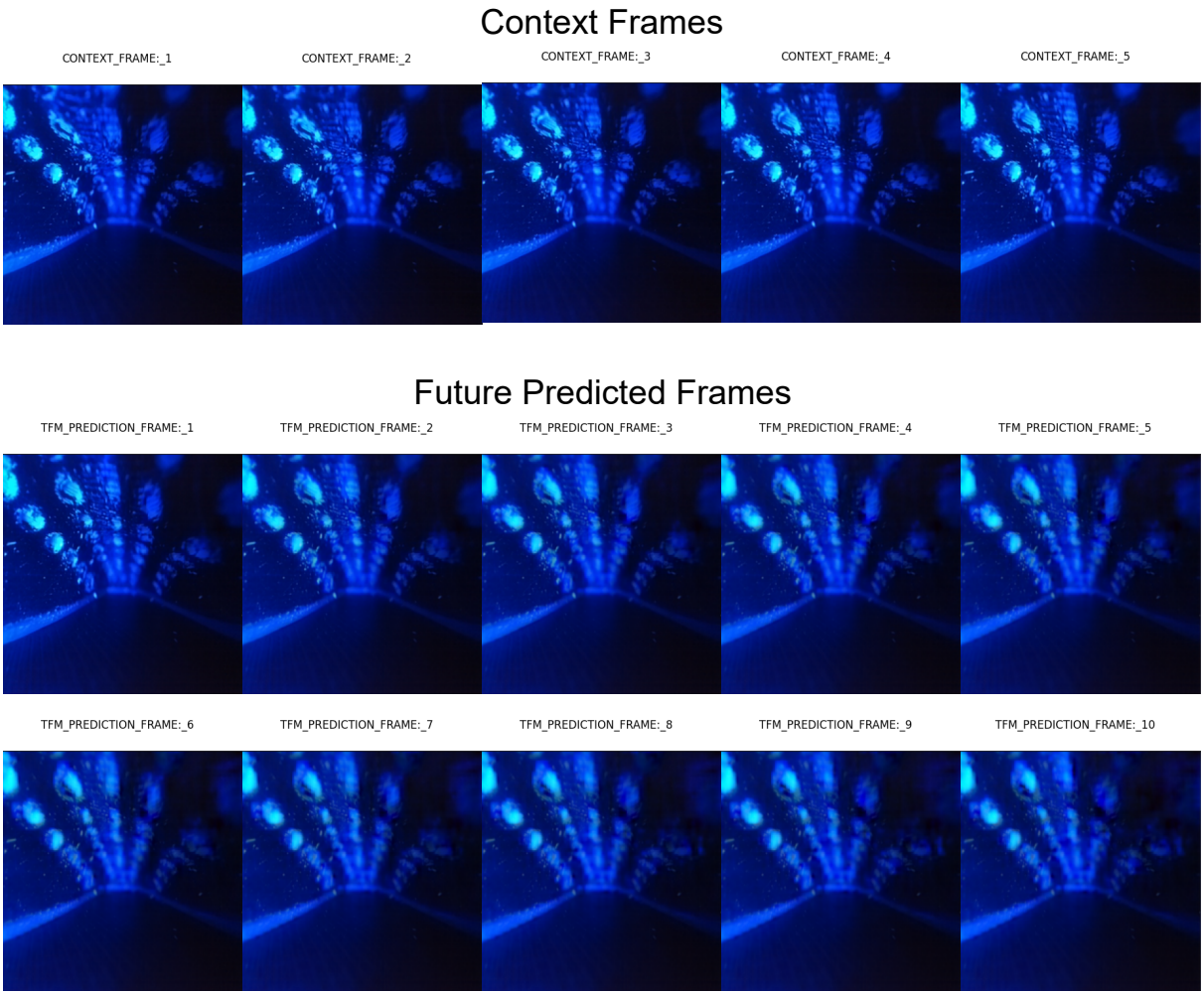


Figure D.1

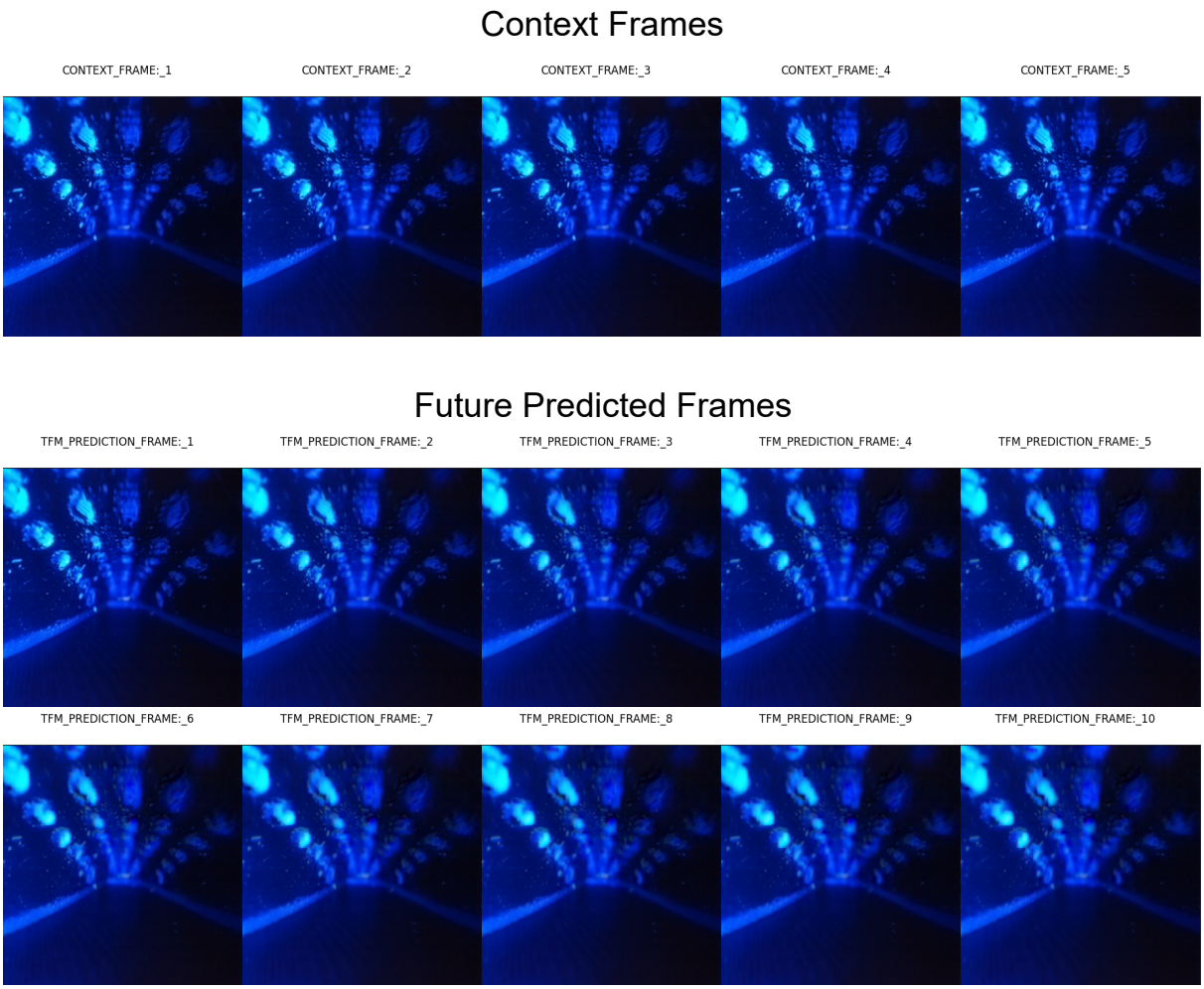

Context Frames
CONTEXT_FRAME:_1
CONTEXT_FRAME:_2
CONTEXT_FRAME:_3
CONTEXT_FRAME:_4
CONTEXT_FRAME:_5
Future Predicted Frames
TFM_PREDICTION_FRAME:_1
TFM_PREDICTION_FRAME:_2
TFM_PREDICTION_FRAME:_3
TFM_PREDICTION_FRAME:_4
TFM_PREDICTION_FRAME:_5
TFM_PREDICTION_FRAME:_6
TFM_PREDICTION_FRAME:_7
TFM_PREDICTION_FRAME:_8
TFM_PREDICTION_FRAME:_9
TFM_PREDICTION_FRAME:_10

Figure D.2

List of Figures

List of Tables

List of Symbols

Variable	Description	SI unit
$\boldsymbol{x_i}$	i-th RGB tactile image from tactile sensor	-
$\boldsymbol{\hat{x}_i}$	i-th predicted tactile image	-
$\boldsymbol{a_i}$	i-th planned robot trajectory (six values: X,Y,Z, ϕ, θ, ψ)	-
$\boldsymbol{s_t}$	stem location at time t	-
$\boldsymbol{\hat{s}_i}$	i-th estimated stem location based on the prediction $\hat{x}_i$	-
$\boldsymbol{e_{i,t}}$	difference between s_t and $\hat{s}_i$, error to minimise	-
$\boldsymbol{\psi}$	yaw angle, rotation around z axis of end-effector frame	rad
$\boldsymbol{\omega_z}$	angular velocity around z axis of end-effector	rad/s
$\boldsymbol{A_t}$	control action, output of d-FPC, in this case equivalent to ω_z	rad/s
$\boldsymbol{a_{t,ref}}$	reference trajectory at time t	-
$\boldsymbol{a_{t,res}}$	residual action value at time t to be added to $a_{t,ref}$	-
$\boldsymbol{k_{p_i}}$	proportional gain	-
$\boldsymbol{k_{d_i}}$	derivative gain	-
$\boldsymbol{c}$	context window	-
$\boldsymbol{T}$	prediction horizon	-
$\boldsymbol{p_0}$	starting end-effector position (X,Y,Z), before push	-
$\boldsymbol{p_f}$	final end-effector position (X,Y,Z), after push	-
$\boldsymbol{q_0}$	starting end-effector orientation (quaternion), before push	-
$\boldsymbol{q_f}$	final end-effector orientation (quaternion), after push	-
$\boldsymbol{\gamma}$	if $\dot{s}_i$ goes above this threshold, we have cases of sudden slippage	-

Acknowledgements

I would like to thank Prof. Amir Ghalamzan for the opportunity and trust given to me in the development of this work and for his supervision and valuable guidance.

My earnest gratitude goes to Prof. Paolo Rocco for assigning me this thesis and giving me the opportunity of this experience abroad and for his continuous availability and important advice.

I would also like to thank all the people who were part of this experience starting with the members of IntManLAB, the people from L-CAS and my fellow travellers. In particular, I would like to thank Willow, for his helpful teachings in the early stage of my work, and Kiyanoush, for his important support in the final testing phase.

I miei più sentiti ringraziamenti vanno alla mia famiglia, in particolare ai miei Nonni, a Francesca e a mio Papà, che mi hanno sempre sostenuto e supportato nel corso di questi anni mentre costruivo il mio futuro.
Il mio grazie più sincero va a mia Mamma, che ha sempre creduto in me anche quando nessun altro, me compreso, lo avrebbe fatto.