Perceive, Represent, Generate: Translating Multimodal Information to Robotic Motion Trajectories

Fábio Guilherme Morgado Vital

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Francisco António Chaves Saraiva de Melo
Prof. José Alberto Rodrigues Pereira Sardinha

Examination Committee

Chairperson: Prof. Daniel Jorge Viegas Gonçalves
Supervisor: Prof. Francisco António Chaves Saraiva de Melo
Member of the Committee: Prof. Rodrigo Martins de Matos Ventura

November 2021

Acknowledgments

I want to thank my family, especially my parents and my sister for their friendship, encouragement, and caring over all these years. You were always supportive and the ones who laid the first stone in my academic journey.

I also want to thank my supervisors, Professor Francisco Melo and Professor Alberto Sardinha, for their support and guidance over this past year. In addition, I want to thank Eng. and Ph.D. student Miguel Vasco for helping me in all steps of this thesis.

To each and every one of you, thank you.

Abstract

In this thesis, we contribute a novel pipeline that maps perceptual information of different modalities (e.g., visual or sound), corresponding to a sequence of commands, to an adequate sequence of movements to be executed by a robot. Our Perceive-Represent-Generate (PRG) framework comprises three stages. In the first stage, we perceive and pre-process the given inputs, isolating individual commands. The second stage, a core element in our pipeline, uses a deep generative model that captures the joint distribution of the perceptual information and the robot movement. Such representation enables the robot to determine the adequate movement given the perceptual input from a command. Finally, the third stage combines the movement for the different individual commands into a dynamic movement primitive, which the robot must execute. We evaluate our pipeline in the context of robotic handwriting, where the robot receives as input a word or sentence (in printed form, handwritten form, or as a sound stream) and determines the complete movement required to write it. We evaluate each stage of our pipeline separately, and as a whole. We discuss the performance of different multimodal generative models within our PRG framework and show that our pipeline can generate coherent and readable handwritten words, regardless of the modalities provided to the model.

Keywords

Human-Robot Interaction; Robotic Handwriting; Dynamic Movement Primitives; Multimodal Learning.

Resumo

Nesta dissertação, nós contribuímos com um novo sistema que mapeia informação perceptual proveniente de diferentes modalidades (por exemplo, visual ou sonora), que corresponde a uma sequência de comandos, em uma sequência de movimentos para serem executados por um robô. O nosso sistema, chamado Observar-Representar-Gerar, inclui três fases de processamento. Na primeira fase, o sistema recebe e processa os dados recebidos, isolando-os em comandos individuais. A segunda fase, que é um elemento central no nosso sistema, usa um modelo generativo que captura a distribuição conjunta dos dados provenientes das várias modalidades e do movimento do robô. Esta representação permite que o robô determine o movimento adequado, dada a entrada de informação perceptual a partir de um comando individual. Finalmente, na terceira fase, os movimentos provenientes dos diferentes comandos individuais são combinados numa primitiva de movimento dinâmico, que o robô deve executar. O nosso sistema é avaliado no contexto da escrita manuscrita robótica, onde o robô recebe como entrada uma palavra ou frase (na forma impressa, escrita à mão através de um certo movimento ou como um fluxo de som) e determina o movimento completo necessário para escrevê-la. O nosso sistema é avaliado como um todo e cada etapa também é avaliada individualmente. Nós avaliamos também o desempenho de diferentes modelos generativos multimodais usados para testar o nosso sistema e ainda mostramos que nosso sistema consegue gerar palavras manuscritas coerentes e legíveis, independentemente das modalidades fornecidas ao modelo.

Palavras Chave

Interação Humano-Robô; Escrita Robótica; Primitivas de Movimento Dinâmico; Aprendizagem Multimodal.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

AE	Auto-Encoder
ALMA	Average Latent Multimodal Approximation
AVAE	Associative Variational Auto-Encoder
CVAE	Conditional Variational Auto-Encoder
DAE	Denoising Auto-Encoder
DMP	Dynamic Movement Primitive
DNN	Deep Neural Network
ELBO	Evidence Lower BOund
HRI	Human-Robot Interaction
iid	independent and identically distributed
JMVAE	Joint Multimodal Variational Auto-Encoder
KL	Kullback–Leibler
LWR	Locally Weighted Regression
MDN	Mixture Density Network
MUSE	**M**ultimodal **U**nsupervised **Se**nsing
MoE	Mixture of Experts
MMVAE	MoE-Multimodal Variational Auto-Encoder
MVAE	Multimodal Variational Auto-Encoder
OpenRAVE	Open Robotics Automation Virtual Environment
PoE	Product of Experts
PRG	Perceive-Represent-Generate
RBM	Restricted Boltzmann Machine
RNN	Recurrent Neural Network

ROS	Robot Operating System
VAE	Variational Auto-Encoder

1

Introduction

Contents

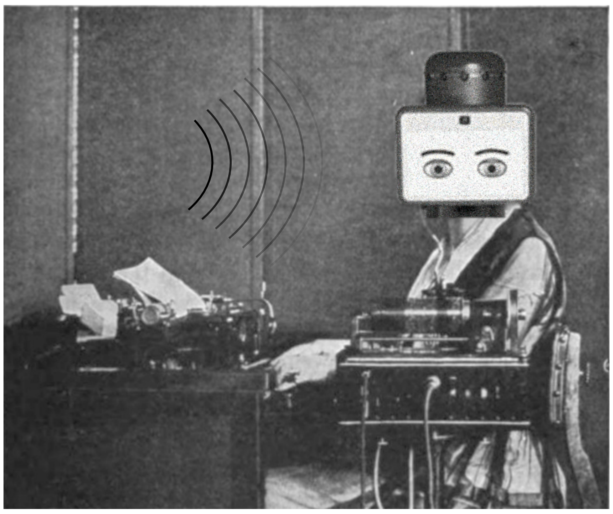


Figure 1.1: In this work, we study how to translate multimodal information into a sequence of movements executed by a robotic agent. In the illustration, a human can dictate a text or present it visually. The agent must process and decompose that information to derive the movement to type each text letter. Adapted from The Expert Typist by Charles Smith (1922).

Recent advancements in artificial perception [1–3] and actuation [4, 5] have fostered the widespread use of robotic platforms in a myriad of tasks, from autonomous driving [6, 7] to industrial manufacturing [8] as well as in medical [9–11] and education [12, 13] scenarios. Furthermore, humans expect to collaborate with robots in various tasks, taking advantage of capabilities that robots have surpassed humans, such as endurance, strength, repeatability, and operate in hazardous environments. Freeing humans from these tasks have the benefit of making them focus on tasks that fit their inherent skills and capabilities, such as supervising, cognition, adaptability, ambiguity, and flexibility. To have a helpful collaboration between humans and robots, we need to solve some challenges regarding safety, the quality of their interaction, and the mismatch between their perceptual, cognitive, and actuation capabilities [14, 15].

One crucial aspect for improving the efficiency in collaborative environments passes to broad the cognitive capacities of the robotic agent. Adding different types of sensors permits capturing different modalities for a given perceptive interaction. Combining the knowledge and awareness obtained from different modalities empowers the agent's deductive technique to make sense of the current situation and, as such, improves the capability to infer a decision or action. Giving such redundant and complementary inputs can also be favorable when only partial information, one or a few modalities, is available to the agents. By making the collaboration between humans and robots more flexible, the robotic agent will adapt to more changes in the environment and new complex tasks without modifying the robot's programming procedures.

In such collaborative scenarios, humans may employ different natural communication channels at once to provide instructions to the robotic platform, such as speech, gestures, or motion. In this work, we address the problem of how to translate multimodal commands provided by a human user through different communication channels to a sequence of movements executed by a robotic agent. In particular, we consider a scenario where the human user provides high-dimensional perceptual data (e.g., sound, images, motion trajectories) related to the agent's task, and the agent's role is to correlate the given observations, retrieving high-level aspects, to then decompose them into a sequence of individual actions to perform. Consider the example of a robotic dictaphone, as depicted in Figure 1.1: a human user dictates a text or word to a robot. The robot's objective is to decompose the words provided to perform the trajectory with each letter. Another implementation of this scenario occurs when the agent's objective is to handwrite the given text or word instead of typing it. In this new implementation, one noticeably and significant concern is how to correlate and extract relevant information from the input modalities (sound and image) that is effective and suitable to construct the motion to handwrite the text or word. Here, we have two concerns. The first one is how to correlate all modalities to extract a joint representation that captures high-level concepts. The complication is that such concepts can be highly nonlinear due to each modality's different data structure and dimension. The second concern arises when one of the modalities is missing. The modalities' information extraction process must be flexible enough to support missing data (modalities), which turns the agent adaptable and capable of acting even when only part of the information is available. In the last segment of this scenario, there is also the necessity to create handwritten motions to write each word that are smooth and as realistic as possible. Therefore, it is necessary to create character motions coherent with the given input modalities and adequate trajectories to connect consecutive letters of the same word to accomplish the mentioned outcome.

1.1 Research Question

Arising from the discussion introduced above, we formalize the following research question.

Research Question *How can a robotic agent receive multimodal commands and infer the corresponding sequence of movements from each, while being robust to missing input modalities?*

We extend the earlier description and define a multimodal command as independent chunks of information that the agent receives through different communication channels, one for each modality, employed by one or more sensors that will perceive this information. We further specify a movement as a physical motion to be executed by one or more actuators of the robotic agent to act on its environment. Additionally, we identify the type of uncertainty we mentioned. An agent acts under uncertainty, in our setting, if the received command contains only a fraction of all modalities.

Typically, approaches in this setting where the human user interacts with the robotic agent by giving a command, and the agent responds by acting take only one modality as input [16–20]. Such approaches are less flexible and less interactive since we have only one input modality to retrieve information from and map into commands, making them capable of only working on the exact specifications of their initial environment. Prominently, perceptual information changes or acting in more complex environments will potentially break these approaches. In this thesis, we lift the hardness of such configurations and give the agent the ability to retrieve information from different modalities and, more importantly, process and associate such information. The agent will have a complete description of the task, which will only benefit its reasoning to choose the sequence of movements to execute. Joining information from different modalities will also allow the agent to act on uncertainty, where, for example, only one input modality is present.

1.2 Proposed Solution

In this work, we contribute a novel three-stage framework Perceive-Represent-Generate (PRG), that maps multimodal commands provided from raw perceptual data given by a human user to a sequence of movements to be executed by the robot. Initially, the agent perceives the environment, collecting and processing the raw multimodal observations into a sequence of individual task components (e.g., letters in a word). Subsequently, the agent represents the individual task components, mapping them into multimodal representations to learn a high-level factorization, capturing high-level concepts that implicitly correlate all modalities. Crucially, as humans may not employ all communication channels to provide information to the agent, such multimodal representation must be robust to missing modality information. Finally, in the third stage, the agent generates and merges the motion information provided by the individual representations in order to execute the estimated action/motion.

We instantiate our PRG pipeline in the scenario of the Robotic Dictaphone, where the agent is provided with textual information (through a combination of sound, image, or motion trajectory observations) and generates a single motion trajectory related to the target word mimicking human handwriting. We chose the problem of handwriting to evaluate our pipeline since handwriting synthesis is still a complex open problem with practical applications such as text recognition systems, forgery detection, and improving CAPTCHA tests. Handwriting synthesis can also be used in education to help children learn how to write [21, 22].

To evaluate our pipeline, we conduct a quantitative evaluation of the representation stage separately, giving us insight into how well we encode the multimodal perceptions and thereby checking the degree of effectiveness that the agent can cross-generate missing modalities. We also evaluate the entire pipeline qualitatively, focusing on how well the agent can generate smooth handwritten words approximating it to the calligraphy of a human. The results show that our approach can robustly map multimodal commands to generate accurate handwritten word samples, regardless of the set of communication channels employed by the human to provide the commands.

1.3 Contributions

We now emphasize, in more detail, each contribution of this thesis succinctly presented in Section 1.2. The main contributions are the following.

1.3.1 Perceive-Represent-Generate Framework

We create a general-purpose framework, called Perceive-Represent-Generate (PRG), having as main objectives creating an agent that can receive and decode perceptual information from more than one input modality. Our framework enrolls in a one-way interaction process where the human user gives a command containing information from multiple modalities to a robotic agent that will process it and act upon it. Using multiple modalities aims to infer a more profound knowledge of the description of the problem and environment, having the overall benefit of decreasing possible errors and acting upon uncertainty, e.g., when the given command only contains information from a fraction of input modalities.

Our framework is composed of three main stages. In the first stage, the framework receives a multimodal command. This multimodal command is a composition of commands from different modalities. Each modality command is processed alone and decomposed into a sequence of sub-commands that are more processable and descriptive for the problem in question. The next step receives each sub-command sequence, one for each given input modality, and iteratively encodes the sub-command items into a sequence of multimodal latent representations of the same length. This latent representation sequence is then decoded into the target modality, in our case, motion. We use a deep generative model, a generic Multimodal Variational Auto-Encoder (MVAE) model, to perform the encoding and decoding processes. Such models allow high-dimensional data processing and learning an efficient high-level representation that contains and associates information from all input modalities. Finally, in the third stage, the resulting sequence of individual motion information is processed and merged into a Dynamic Movement Primitive (DMP) suitable for the robot's actuation.

1.3.2 Robotic Dictaphone Scenario

In finding a complex problem, we chose handwriting synthesis as our applicational problem. In this problem, we have the opportunity to learn and retrieve shared aspects of different concrete representations (sound modality when we hear a word, image modality when we read a word, or motion modality when we write a word) of a word (an abstract concept). We also give special attention to the motions to generate when handwriting a word since it is necessary to have a precise and smooth movement to handwrite eligible words similar to human samples.

Given such reasons, this problem is helpful to test the efficiency and scalability of our framework. Consequently, we instantiate our framework, PRG, in a novel Robotic Dictaphone scenario where textual information from multiple modalities, like speech, a visual representation, or even a motion trajectory of the word, is converted to robotic motion trajectory, mimicking human handwriting. As introduced in the beginning of Chapter 1, one of the challenges of this scenario is to correlate and extract valuable information from the given input modalities (image or sound) to infer the handwritten motion to write the underlying word. Our framework solves this problem inherently due to its design. The generative model, an MVAE used in the second and third stages, learns a latent representation that jointly embeds all modalities retrieving relevant information to regenerate any modality even if it was initially missing. Additionally, we test different architectures (implementations) for the generative model used, which allows us to vary the number of modalities and evaluate their performance, focusing on scalability and cross-modality inference. Our primary implementation takes into account all three modalities and a generative model that considers all modalities equally.

We also introduce a new algorithm in the last stage of our pipeline, PRG, to connect the resulting letter motions of the underlying word to handwrite. Using a basic approach produces only a straight line when connecting two consecutive letter trajectories. That is what a basic DMP would produce. The proposed algorithm overcomes several drawbacks of the basic solution since it considers several aspects, such as distance and angle of the connection trajectory and the following letter trajectory, and the order of the point to connect. The new algorithm creates a smoother connecting trajectory generating a more natural motion to the overall word trajectory. We evaluate different configurations of this algorithm. We also evaluate the complete pipeline as a whole, focusing on how well the pipeline can produce handwritten words that approximate a human's calligraphy. The results obtained show promising results where our pipeline can receive multimodal commands containing textual information of a word and generate the motion to handwrite the underlying word accurately. Finally, we employ and validate our work in a virtual simulation where a robotic agent operates using our framework.

Despite focusing on one scenario, our framework is generic and works on other scenarios. One example is a grasping scenario where a robotic arm grasps a specific object. A possible combination of modalities is the movement, RGB image of the robotic arm and object to grasp, depth image from the perspective of the robotic arm, and we could also have a 360-degree image of the object to grasp. It could be interesting to see how accurately the framework can generate the complete grasping motion (get closer to the object and then grasp it) given the 360-degree image with the RGB or depth image. Another interesting scenario is, for example, having a small robot to navigate indoors. The four modalities would be trajectory, lidar data, surrounding image, and label of the room of each point of the multi-step trajectory. Using our pipeline, we could, for example, infer the position in the map given the 360º image and its label or provide only the label to infer a possible 2D position that belongs to that label (room). This scenario would require developing specific perceptual maps, but the core of the contribution would be the same: using a multimodal generative model to encode and generate modality-specific data.

Lastly, we would also like to mention that our framework is reversible. Even though we are focusing on only one direction for the generation of modalities, from speech or an image to a handwritten movement, in the primary implementation of the Robotic Dictaphone scenario, we could easily, for example, change the pipeline to perform the generation of the speech from a given motion of the word. Since the used generative model considers all modalities equally, it is possible to generate any combination of modalities giving, as input, any other. In this case, we only need to change the third stage to generate the sound modality to give it to the agent's speaker.

1.4 Organization of the Document

The remainder of the document contains the following structure: Chapter 2 presents the necessary background knowledge to derive our solution. We present related work in Chapter 3. Next, in Chapter 4, we present our pipeline, PRG, in detail, as a general framework. Chapter 5 evaluates our pipeline in a specific scenario, Robotic Dictaphone, where textual information is converted to robotic motion trajectory, mimicking human handwriting. Finally, in Chapter 6, we conclude our work by presenting some concluding remarks.

2 Background

Contents

This section describes two topics we use in this research work, namely Variational Auto-Encoders (VAEs) and Dynamic Movement Primitives (DMPs).

2.1 Variational Auto-Encoders

In this sub-section, we first derive the VAE and then introduce recent works that extend the original framework to include more than one modality. We present some MVAE frameworks having different ways of joining and learning the latent representations of multiple modalities.

2.1.1 VAE

Given a dataset $\mathbf{X}$ containing N independent and identically distributed (iid) samples, where $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$, of some variable $\mathbf{x}$. We assume that the data is generated by a random process of the form $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$, where, first, the low-dimensional latent variable $\mathbf{z}$ is generated from a prior distribution (usually a Gaussian distribution), $\mathbf{z} \sim p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$.

The objective of the VAE model is to maximize the marginal distribution $p(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})\, d\mathbf{x}$, where $p_\theta(\mathbf{x}|\mathbf{z})$ is modeled using a deep neural network, called <u>decoder</u>, with parameters θ. Since this distribution is intractable, VAE is trained to maximize an approximate lower bound of it, called Evidence Lower BOund (ELBO), instead. Since we know that

$$\begin{aligned} D_{\mathsf{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{x}|\mathbf{z})) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z})}\right] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] + \log p(\mathbf{x}) \\ &= D_{\mathsf{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z})) - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] + \log p(\mathbf{x}) \end{aligned} \tag{2.1}$$

where D_{KL} is the Kullback–Leibler (KL) divergence between two distributions and $q_\phi(\mathbf{z}|\mathbf{x})$ is a distribution modeled with a deep neural network, with parameters ϕ, called <u>encoder</u>. The encoder tries to approximate the original and intractable posterior distribution $p(\mathbf{z}|\mathbf{x})$. Re-writing (2.1) as:

$$\log p(\mathbf{x}) = D_{\mathsf{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{x}|\mathbf{z})) + L_{\mathsf{ELBO}} \tag{2.2}$$

where L_{ELBO} is the ELBO, and it is defined as:

$$\log p(\mathbf{x}) \geq L_{\mathsf{ELBO}} = -D_{\mathsf{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \tag{2.3}$$

Instead of approximating $\log p(\mathbf{x})$, if we only approximate the ELBO operand, L_{ELBO}, in (2.2) we perform a maximization that is computationally tractable since we avoid computing the KL divergence between the approximate and exact posteriors, $D_{\mathsf{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{x}|\mathbf{z}))$. L_{ELBO} is a lower bound of $\log p(\mathbf{x})$ since the KL divergence, $D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{x}|\mathbf{z}))$, is always non-negative. For this reason, we maximize L_{ELBO}. We finally arrive at VAE's loss function that is the negative of L_{ELBO}.

$$\mathcal{L}_{\text{VAE}}(\mathbf{x}) = -L_{\text{ELBO}} = D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})) - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \tag{2.4}$$

As such, minimizing $\mathcal{L}_{\text{VAE}}$ is the same as maximizing L_{ELBO}.

From the VAE's loss function, $\mathcal{L}_{\text{VAE}}$, we sample $\mathbf{z}$ from $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$. Since sampling is a stochastic process it is impossible to back-propagate the gradient through z. For this reason, the reparametrization trick is introduced [23] in order to express $\mathbf{z}$ as a deterministic variable such that $\mathbf{z} = \mathcal{D}_\phi(\mathbf{x}, \boldsymbol{\epsilon})$, where $\mathcal{D}_\phi$ is a function parametrized by ϕ receiving as input $\mathbf{x}$ and an independent and random auxiliary variable $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$. For example, in the case of $\mathbf{z}$ being sampled by a Gaussian distribution, $\mathbf{z} \sim p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$, we can use instead $z = \mathcal{D}_\phi(\mathbf{x}, \boldsymbol{\epsilon}) = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$, where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\odot$ represents the element-wise product, resulting in a simplification of (2.4):

$$\mathcal{L}_{\text{VAE}}(\mathbf{x}) = \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log\left(\sigma_j^2\right) - \mu_j^2 - \sigma_j^2\right) + \frac{1}{L}\sum_{l=1}^{L}\log p_\theta\left(\mathbf{x}|\mathbf{z}^{(l)}\right), \tag{2.5}$$

where $\mathbf{z}^{(l)} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon^{(l)}$ and $\epsilon^{(l)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. See Figure 2.1 for an example.

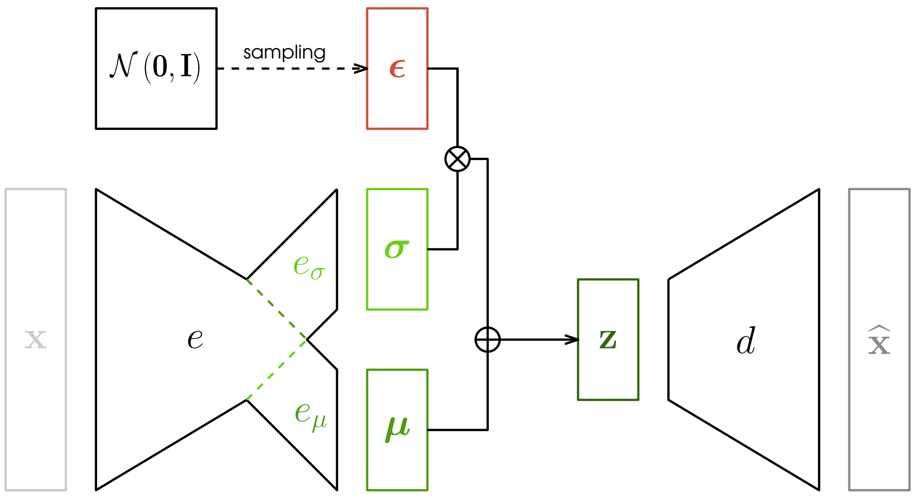


Figure 2.1: VAE architecture. We assume that $\mathbf{z}$ follows a Gaussian distribution, where after the reparametrization trick, we have $\mathbf{z} = e_\mu(e(\mathbf{x})) + e_\sigma(e(\mathbf{x})) \odot \boldsymbol{\epsilon}$. In the decoding phase, we re-generate the given input, where $\widehat{\mathbf{x}} = d(\mathbf{z})$. See Section 2.1.1 for more details.

2.1.2 CVAE

With the original VAE model, we do not have control over the data generation process. In particular, we cannot conditionally generate $\mathbf{x}$ given another variable y. The Conditional Variational Auto-Encoder

(CVAE) [24] permits this, taking advantage that, at inference time the generation process will only occur in one direction (from y to $\mathbf{x}$). The generation of $\mathbf{x}$ takes now the form $p_\theta(\mathbf{x}|\mathbf{z}, y)$, where $y \sim p(y) = \text{Cat}(y|\boldsymbol{\pi})$ and $\text{Cat}(y|\boldsymbol{\pi})$ is a categorical distribution. Also, the encoding's distribution considers y, with $q_\phi(\mathbf{z}|\mathbf{x}, y)$. Both encoding's and decoding's phases are additionally conditioned with y, see Figure 2.2. Therefore the CVAE's loss function takes the form:

$$\mathcal{L}_{\text{CVAE}}(\mathbf{x}, \mathbf{y}) = D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}) \parallel p(\mathbf{z})) - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z})] \tag{2.6}$$

CVAE has several drawbacks. Normally, the maximum number of conditional modalities that CVAE can receive is one. Moreover, CVAE only targets the cross-modal generation of data in one direction, limited to the generation of $\mathbf{x}$ given y.

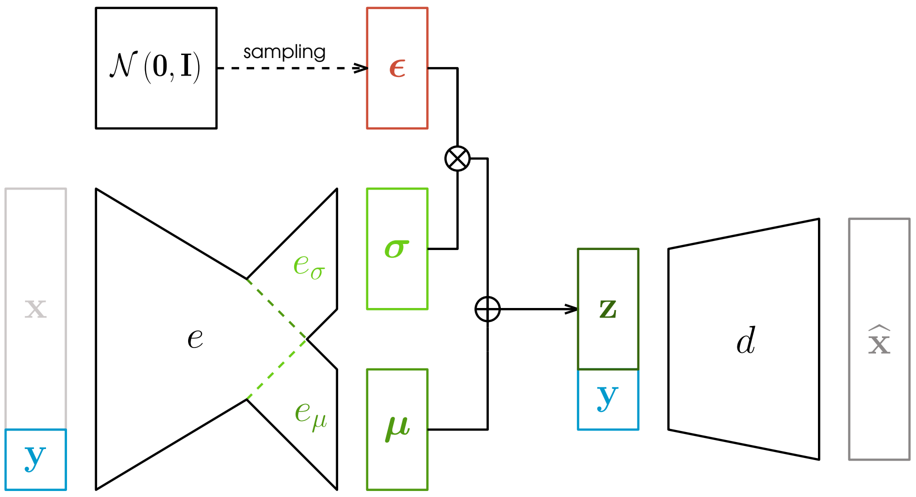


Figure 2.2: CVAE architecture. We assume that $\mathbf{z}$ follows a Gaussian distribution, where after the reparametrization trick, we have $\mathbf{z} = e_\mu(e(\mathbf{x}, \mathbf{y})) + e_\sigma(e(\mathbf{x}, \mathbf{y})) \odot \boldsymbol{\epsilon}$. In the decoding phase, we re-generate the given input, where $\widehat{\mathbf{x}} = d(\mathbf{z}, \mathbf{y})$. See Section 2.1.2 for more details.

2.1.3 AVAE

Another extension of the VAE enhances the original framework to scenarios having multiple modalities. Given two input modalities, the Associative Variational Auto-Encoder (AVAE) model aggregates two VAEs, one for each modality. With this configuration, it is possible to associate each modality's latent space to learn modality-specific information, and also, shared and high-level knowledge that enables the cross-generation of one modality given another one, see Figure 2.3 for a visual representation. Given $\mathbf{x}_1$ and $\mathbf{x}_2$, two input modalities, AVAE's loss function is the following:

$$\mathcal{L}_{\text{AVAE}}(\mathbf{x}_1, \mathbf{x}_2) = \mathcal{L}_{\text{VAE}}(\mathbf{x}_1) + \mathcal{L}_{\text{VAE}}(\mathbf{x}_2) + \alpha D^{\star}_{\text{KL}}(q_\phi(\mathbf{z}_{\mathbf{x}_1}|\mathbf{x}_1) \parallel q_\phi(\mathbf{z}_{\mathbf{x}_2}|\mathbf{x}_2)), \tag{2.7}$$

where $D^{\star}_{\text{KL}}(p \parallel q)$ is the symmetrical KL divergence between two distributions p and q, and α is the parameter that weights the similarity between the modality-specific latent spaces during training.

One massive downside of AVAE model is that the number of parameters explodes combinatorially as we linearly add more modalities. The reason is that we need a new encoder for each unique combination of modalities. Only with this structure AVAE model can learn a joint latent space capable of cross-generating data from any possible combination of modalities.

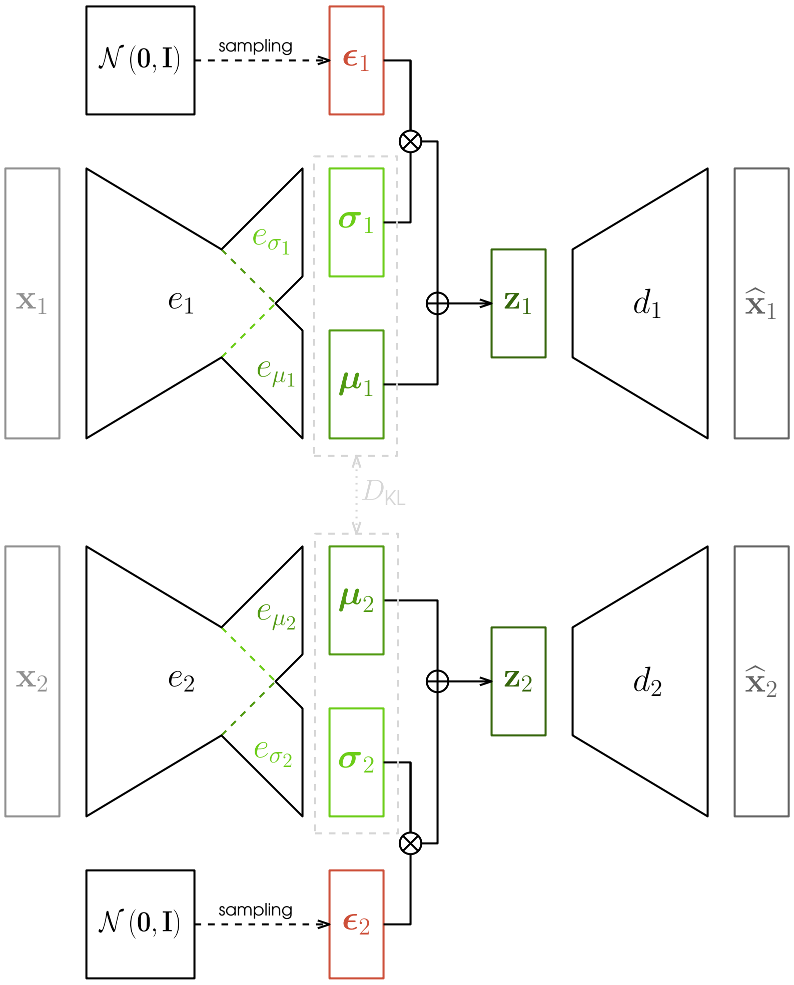


Figure 2.3: AVAE architecture for two input modalities, $\mathbf{x}_1$ and $\mathbf{x}_2$. There is a VAE model for each modality. This architecture approximates all VAEs using an additional divergence term. See Section 2.1.3 for more details.

2.1.4 MUSE

Multimodal **U**nsupervised **Se**nsing (MUSE) is a novel framework proposed to also manage multimodal information [25]. This model learns a multimodal representation, robust to missing modality information and scalable to more than two modalities due to the implementation of hierarchical representation levels. At a low level, the model encodes information specific to each modality, learning, this way, information that is intrinsic to each modality (modality-specific). MUSE merges the latent description of each modality-specific representation at a higher level, capturing shared cross-modality information. As a result, the high-level component encodes a robust representation that is coherent with all given modalities.

Assuming M modalities, $\mathbf{x}_{\text{multi}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$, each $\mathbf{x}_m$ is generated by the corresponding latent variable $\mathbf{z}_m$. MUSE's approach is composed of M VAEs for the low-level component, where the mth VAE generates the mth input modality, $\mathbf{x}_m$. Assuming that every modality is independent from each other, $p_\theta(\mathbf{x}_1, \ldots, \mathbf{x}_M) = \Pi_{m=1}^{M} p_\theta(\mathbf{x}_m)$, we implement the following loss regarding the low-level component:

$$\mathcal{L}_{\text{MUSE\_low}}(\mathbf{x}_1, \ldots, \mathbf{x}_M) = \sum_{m=1}^{M} \alpha_m D_{\text{KL}}(q_\phi(\mathbf{z}_m|\mathbf{x}_m) \parallel p(\mathbf{z}_m)) - \mathbb{E}_{q_\phi(\mathbf{z}_m|\mathbf{x}_m)}[\lambda_m \log p_\theta(\mathbf{x}_m|\mathbf{z}_m)] \tag{2.8}$$

where α_m regularizes the weight of each modality's regularization term and λ_m adjusts the reconstruction of each modality-specific observation, see Figure 2.4.

Regarding the high-level component (see Figure 2.5) the objective passes to learn a multimodal representation, $\mathbf{z}_\pi$. It is assumed that $\mathbf{z}_\pi$ generates modality specific data $\mathbf{c}_m$, $m \in \{1, \ldots, M\}$, where $\mathbf{c}_m$ is sampled using the encoder of the mth low-level VAE, $\mathbf{c}_m \sim q_\phi(\mathbf{z}_m|\mathbf{x}_m)$. This way, the following loss is employed to learn $\mathbf{z}_\pi$:

$$\begin{aligned}\mathcal{L}_{\text{MUSE\_high}}(\mathbf{x}_1, \ldots, \mathbf{x}_M) = \mathbb{E}_{q_\phi(\mathbf{c}_1|\mathbf{x}_1), \ldots, q_\phi(\mathbf{c}_M|\mathbf{x}_M)} \Bigg(& \beta D_{\text{KL}}(q_\phi(\mathbf{z}_\pi|\mathbf{c}_1, \ldots, \mathbf{c}_M) \parallel p(\mathbf{z}_\pi)) \\ & - \sum_{m=1}^{M} \mathbb{E}_{q_\phi(\mathbf{z}_\pi|\mathbf{c}_1, \ldots, \mathbf{c}_M)}[\gamma_m \log p_\theta(\mathbf{c}_m|\mathbf{z}_\pi)] \Bigg)\end{aligned} \tag{2.9}$$

where β adjusts the regularization of $\mathbf{z}_\pi$ and γ_m adjusts the reconstruction of the modality-specific variables.

MUSE can scale to a large number of variables by approximating the joint posterior distribution using a Product of Experts (PoE) approach:

$$p(\mathbf{z}_\pi|\mathbf{c}_1, \ldots, \mathbf{c}_M) = p(\mathbf{z}_\pi) \prod_{m=1}^{M} q(\mathbf{z}_\pi|\mathbf{c}_m) \tag{2.10}$$

Simply using this approach can, sometimes, occur that the solution becomes biased on overconfident experts, disregarding information from modalities having low dimensions. MUSE introduces a new training scheme called Average Latent Multimodal Approximation (ALMA) to solve this issue. This new scheme encodes the multimodal representation $\mathbf{z}_\pi$ and partial-view multimodal latent variables resulting from every other possible combination of modalities. In this setting, MUSE becomes robust to missing

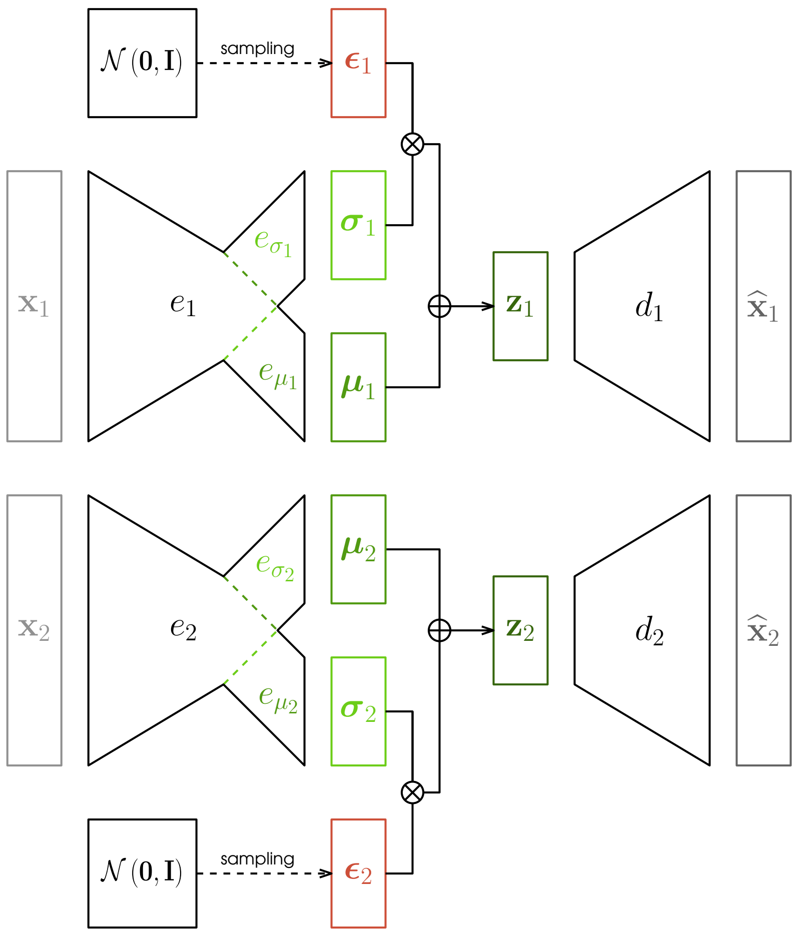


Figure 2.4: MUSE's low level component for two input modalities, $\mathbf{x}_1$ and $\mathbf{x}_2$. There is an independent VAE model for each modality. See Section 2.1.4 for more details.

modalities and agnostic to each modality's nature and dimensionality by forcing every new multimodal latent variable to be similar. The final loss function of MUSE considers low and high-level components and the presented training scheme:

$$\begin{aligned}\mathcal{L}_{\text{MUSE}}(\mathbf{x}_1, \ldots, \mathbf{x}_M) = {} & \mathcal{L}_{\text{MUSE\_low}}(\mathbf{x}_1, \ldots, \mathbf{x}_M) \\ & + \mathcal{L}_{\text{MUSE\_high}}(\mathbf{x}_1, \ldots, \mathbf{x}_M) \\ & + \frac{\delta}{|D|}\sum_{i=1}^{|D|} D^{*}_{\text{KL}}(q_\phi(\mathbf{z}_\pi|\mathbf{c}_1, \ldots, \mathbf{c}_M) \parallel q_\phi(\mathbf{z}_i|d_i))\end{aligned} \tag{2.11}$$

where D contains the partial-view multimodal observations and $d_i \in D$. Also, δ takes into account the weight to give to this $|D|$ partial multimodal terms.

For the Robotic Dictaphone scenario, we will use CVAE, AVAE, and MUSE models models in our framework, particularly in the Represent and Generate stages. We pretend to compare them and analyze which model better serves our pipeline regarding scalability and performance, primarily on cross-

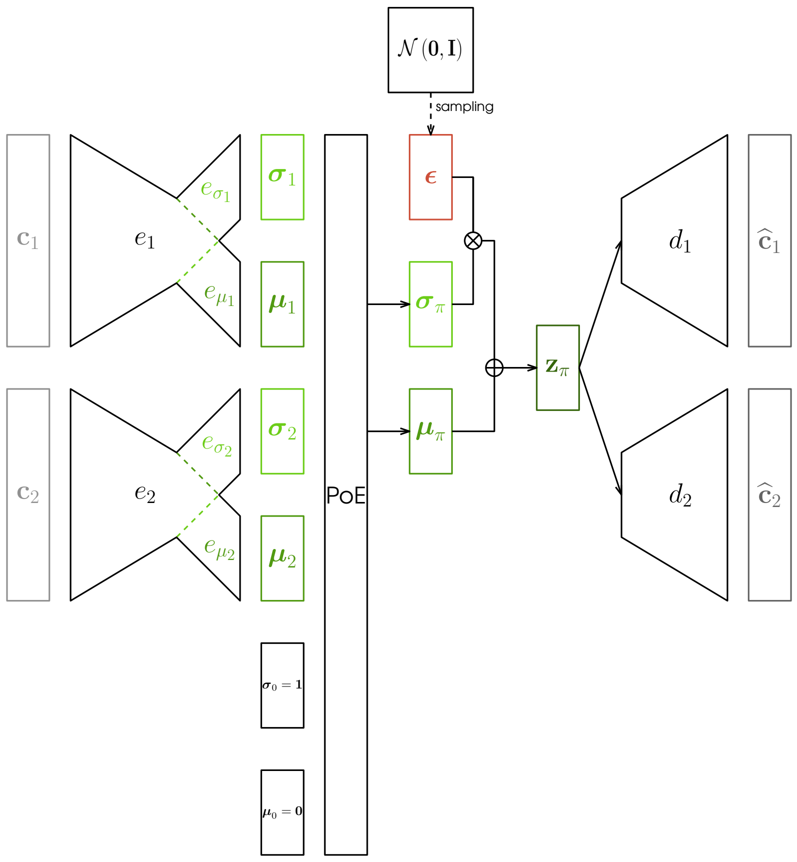


Figure 2.5: MUSE's high level component for two input modalities, $\mathbf{x}_1$ and $\mathbf{x}_2$. $\mathbf{c}_1$ and $\mathbf{c}_2$ are sampled from the encoding distribution of each input modality, $\mathbf{c}_1 \sim q_\phi(\mathbf{z}_1|\mathbf{x}_1)$ and $\mathbf{c}_2 \sim q_\phi(\mathbf{z}_2|\mathbf{x}_2)$. Each representation $\mathbf{c}_1$ and $\mathbf{c}_2$ are passed through new encoders. $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$ represent the ith variational parameters, and $\boldsymbol{\mu}_0$ and $\boldsymbol{\sigma}_0$ represent the prior parameters. All variational parameters are combined using a PoE approach that will return the variational parameters that with ϵ are used to sample $\mathbf{z}_\pi$. $\mathbf{z}_\pi$ is then used to regenerate $\widehat{\mathbf{c}}_1$ and $\widehat{\mathbf{c}}_2$. See Section 2.1.4 for more details.

modality inference. We will evaluate the models individually to understand the capacities of each one, independently of the task. We will also assess the performance of each generative model as part of our pipeline by examining the quality of the outputted handwritten words.

2.2 Dynamic Movement Primitives

DMP [26] is a method to learn and control trajectories. The original DMP formulation that we use was designed for discrete movements, having start and endpoints. Two systems define a DMP: the first, called transformation system, follows a second-order dynamical system

$$\tau \dot{z} = \alpha_z \left(\beta_z \left(g - y\right) - z\right) + f, \tag{2.12}$$

$$\tau \dot{y} = z, \tag{2.13}$$

where y and $\dot{y}$ are the current position and velocity of the trajectory, respectively. z and $\dot{z}$ represent the scaled velocity and acceleration, respectively. y, $\dot{y}$, z and $\dot{z}$ do not necessarily have a fixed dimension, but we define $y, \dot{y}, z, \dot{z} \in \mathbb{R}^2$ for this work. τ is a time constant, and α_z and β_z are positive constants. g is the desired endpoint (the goal) and the point attractor of the system.

The second system, called the canonical system, is a first-order linear dynamical system

$$\tau \dot{x} = -\alpha_x x, \tag{2.14}$$

where α_x is a constant. For a randomly chosen initial state, $x_0 = 1$ describes the start of time evolution. As x converges to 0, its stable fixed point, monotonically, the trajectory approaches its final state. Thus, x serves as a phase signal, re-parametrizing the time evolution $t \in [0, T]$ to $x \in (0, 1]$.

With both systems, we can write f, in (2.12), as:

$$f(x) = \frac{\sum_{i=1}^{N} \Psi_i(x) w_i}{\sum_{i=1}^{N} \Psi_i(x)} x \tag{2.15}$$

where each $\Psi_i(x)$ is a Gaussian basis function:

$$\Psi_i(x) = \exp\left(-\frac{1}{2\sigma_i^2}(x - c_i)^2\right), \tag{2.16}$$

with c_i and σ_i are constants corresponding to the center and width of the basis function Ψ_i, respectively. We define the centers, c_i, as [27]:

$$c_i = \exp\left(-\alpha_x i \frac{T}{N}\right),\ i \in \{0, \ldots, N-1\} \tag{2.17}$$

where N is the number of basis function to define. Through (2.17), the centers are equally spaced in the original time interval $[0, T]$. We formulate the widths, σ_i, of the basis functions as [27]:

$$\begin{aligned} \sigma_i &= \frac{c_{i+1} - c_i}{\sqrt{2\tilde{h}}},\ i \in \{0, \ldots, N-1\} \\ \sigma_N &= \sigma_{N-1} \end{aligned} \tag{2.18}$$

where $\tilde{h} > 0$ adjusts the overlapping range between basis functions (usually $\tilde{h} = 1$).

Such a definition of f starts to vanish when the generated trajectory approaches g, forcing the global equilibrium point (point attractor) to appear at $(z, y, x) = (0, g, 0)$. In addition, we assume that $g \neq y_0$, meaning the offset between the end and start point of the trajectory is never 0. To generate complex trajectories, we use learning algorithms, such as Locally Weighted Regression (LWR) [28], to regulate the parameters w_i of (2.15), forming discrete pattern generators as y tends to g. See Figure 2.6 for an example of fitting a DMP into a given trajectory.

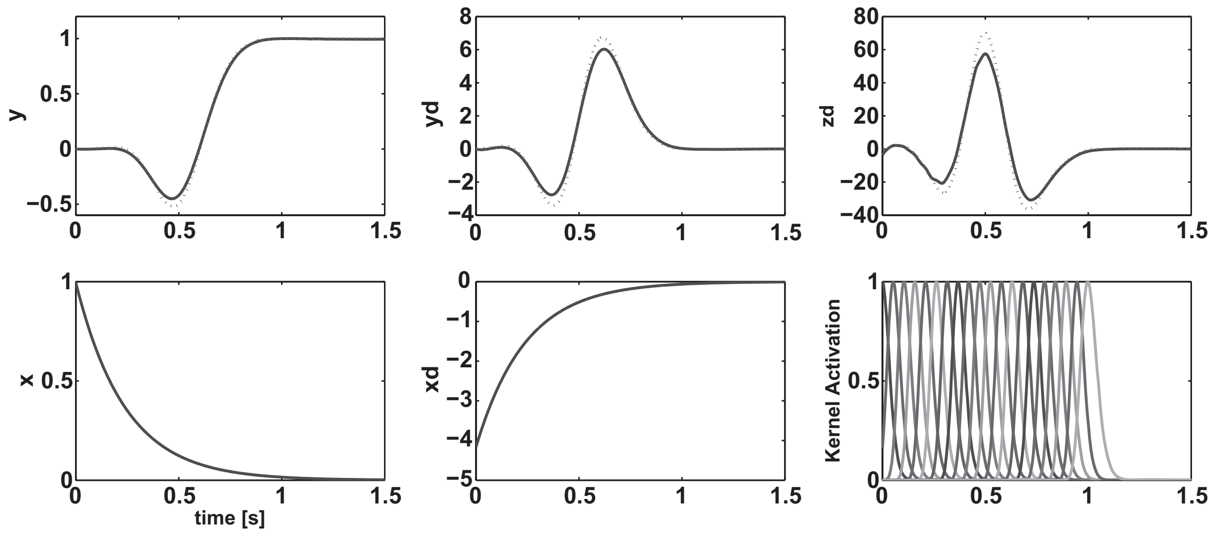


Figure 2.6: Example of a discrete dynamical system. The parameters w_i from (2.16) have been tuned to fit a trajectory that is a fifth-order polynomial. The top plots, from left to right, show the time evolution of the position, velocity and acceleration of the computed trajectory (dotted lines) and the real trajectory (solid lines). The bottom right plot shows the 20 exponential kernels as a function of time, where the equal spacing between them corresponds to an exponential spacing in x. Image taken from [29].

3 Related Work

Contents

This section presents relevant literature regarding robotic control, previous work in handwriting generation and two fundamental components of our approach: multimodal representation learning and DMPs.

3.1 Multimodal Representation Learning

Kingma *et al.* introduced a new generative model called the VAE [23]. VAE models have their wide usage in generative modeling of single-modality data in an unsupervised learning setting. First, an inference model is used to codify essential attributes of the observations into a latent space. It is then possible to re-generate the given observation from the latent space using a decoder.

Various works extend VAE's original work to consider the generative modeling of multimodal data. CVAE model [24] is one of them by conditioning the input and latent space with another independent conditional variable, allowing, at inference time, the cross-modal generation of the input variable from the added conditional variable.

Another work introduces the AVAE model [30]. This model uses an independent VAE for each input modality and considers the approximation of single-modality latent distribution, enforcing the similarity between them with a statistical loss term. With this model, we gain the possibility to perform cross-modality inference from any modality to any other one. The problem with this implementation is the explosion in the number of parameters as we increase the number of modalities since we need a new encoder for each unique combination of input modalities.

Another model, called Joint Multimodal Variational Auto-Encoder (JMVAE) [31], attempts to model the joint distribution of the input modalities explicitly. JMVAE uses a joint encoder that receives all modalities as input to encode a shared high-level representation. To handle missing modalities at test time, the model supplements the inference process with unimodal encoders. These unimodal encoders have a training process that focuses on matching the joint encoder using an additional divergence term. One drawback of this model is that we need an extra unimodal encoder to train for every new input modality used.

A more recent work introduced the MVAE model [32]. This model allows learning a multimodal representation with a higher number of modalities by merging modality-specific information. This model learns a joint distribution of all modalities using a PoE approach that receives the marginal posteriors of all modalities. With this approach, there is no need to have extra components or training steps to perform cross-modal inference at test time. Still, there is a possible problem with such implementation. When there are modalities with a vast difference in dimensionality, the model can struggle to perform cross-modal inference from a low dimensional input modality. In this case, the model suffers from overconfident experts discarding low-dimensional modalities.

Two recent studies try to overcome this problem of using the PoE method to learn a multimodal representation. The first one introduced the MoE-Multimodal Variational Auto-Encoder (MMVAE) model [33] and addressed the issue by using a Mixture of Experts (MoE) solution to merge information from multiple modalities instead. However, the training of this solution incurs a high computation cost, as it requires an importance weighted sampling training scheme.

The second approach, the MUSE model, was presented in the context of state representation for reinforcement learning agents in multimodal scenarios [25], allowing the possibility to learn multimodal representations scalable to a higher number of modalities and robust to missing modality information. The model achieves this by employing a PoE solution with a modified training scheme that considers all possible combinations between latent representations from each input modality as additional loss terms.

3.2 Dynamic Motion Control

There is an extended work in dynamic motion control concerned with generating precise trajectories and controlling their dynamic parameters. DMPs, Gaussian Mixture Models, and splines are examples of such methods. DMPs [26] encode a particular trajectory using a stable attractor system (damped spring model), making them robust to perturbations and guaranteeing convergence to a given target. Other essential characteristics make DMPs widely used in motion control systems. DMPs can react to external forces in the instance that it happens. There are also efficient algorithms that can learn and fit DMPs from the given data. Additionally, DMPs are adaptable enough to produce complex trajectories.

One crucial and desired feature to DMPs is joining individual and simpler DMPs to create a more complex movement. In this case, it is necessary to consider the conversion of each trajectory and how to handle the transition/joining from one trajectory to the next one. We call this problem DMP joining.

A past work [34] suggested a straightforward implementation where the next DMP starts before the current DMP finishes, and its velocity is below a predefined threshold. To prevent discontinuities, the initial state of the next DMP is the same as the state that the current DMP finishes.

Another work used an extension of the original DMP formulation to track a moving target at a specific velocity [35]. Since it is possible to know the exact time and velocity that the DMP passes by that target, if we create such a target point with information from the next DMP, it is possible to join both DMPs with a continuous velocity.

A different work [36] used a new DMP formulation where a piecewise-linear function replaced the goal function, and the canonical system is a sigmoidal decay function. The force-term was also updated to take into account the new canonical system. This work also uses a single set of overlapping kernels for the whole joint trajectory leading to smooth transitions between each trajectory.

3.3 Human-Robot Interaction

In Human-Robot Interaction (HRI) and robotic control, several studies focus on how to control an agent through commands and simplify that communication. For instance, turning the set of commands more flexible and abstract to interpret some level of uncertain information [20] turns instructions more high-level and, at the same time, gives the agent more freedom.

There are several examples of works, in different sectors, that use a set of instructions (commands) to control a robotic agent: in robotic navigation through the use of directional voice instructions [16] and another, with the same objective, but using electrocardiography signals as commands [17]; using voice commands to control a prosthetic robotic arm [18]; controlling industrial machines through command voices [19]. Another work in robotic navigation tries to include and process uncertain information in the voice commands [20], making it easier to learn and interact, which gives the user a better and natural experience. One drawback in these systems is that they tend to only focus on one perceptual modality, typically the sound.

Recently, an increasing amount of works have been focusing on including multimodal perceptual information in HRI problems. In these works, the challenge passes to use more than one modality to retrieve further information about the environment to have a better execution plan, minimize errors, and increase performance.

A recent work [37] presents a multimodal HRI system where the user can use natural speech to issue commands to apply a movement on a robotic agent. Initially, a speech model converts the emitted commands into text. A maximum entropy model further processes the given text to understand the intention of the user. Then the agent understands if the speech instruction is an independent command by itself and, if so, the command alone controls the robot actuation procedure. Another possible alternative is when the speech instruction contains information suggesting that the user intends to give more detailed information with his gestures and, in this case, a motion sensor captures the user's gesture. Subsequently, the captured motion is combined with the speech command to control the robotic agent. Our work differs from this one since we learn a joint representation of all input modalities. We can exchange information in every direction between the modalities. On the contrary, in this work, we have a principal modality (speech) and a second modality (motion) that can only complement the first one.

Another recent work [38] combines audio-visual inputs to derive commands to operate a robotic platform. This framework captures audio and video samples individually. There is also a set of commands that the robotic agent can recognize, where each one has a corresponding audio and gesture signature. After capturing and processing each modality sample individually, the model attributes a score to each command. The scores for the gesture and spoken commands are normalized and linearly combined to generate a single score. The command with the best score is the one that the agent will execute. In this work, there is an attempt to associate both modalities and, sort of, retrieve joint information from each. The drawback is that the modalities are first converted into a vector of scores and then linearly combined to retrieve a final estimation. Using a deep generative model, as in our framework, we further capture high non-linear relations that jointly embed and correlate both modalities.

A different work used three modalities, speech, hand motion, and body motion to determine the command, from a fixed set, to be executed by a robot [39]. This framework processes each input modality individually to retrieve features with low dimensionality but highly descriptive, using specific Deep Neural Network (DNN) approaches according to the modality's data type. These features are further concatenated and passed to a final DNN to retrieve the command's label to execute. The authors state that this framework obtains better results than the unimodal models for each modality. No results evaluate how the framework behaves when there are missing modalities and how it influences its accuracy. A related work captures information from speech and gesture using two individual DNN pipelines, one for each modality [40]. Both pipelines output the recognition confidence for the respective modality that will be further normalized and synthesized in a D-S evidence theory algorithm incorporating a rule-based intention voter scheme. This new approach considers both modalities' confidence to get higher precision and reliability, making this framework robust to missing and conflicting modalities. The output is a label that will be converted into a fixed motion for the agent to execute.

These two previous frameworks are suitable when there is a simple need for a many-to-one mapping. Accordingly, when given a set of multimodal inputs, there is only the need to compute a label that, in this case, will be translated into a hard-coded movement to be executed by the robotic agent. To mitigate this constraint and have a one-to-many mapping, we need a model equivalently capable of learning a joint representation of all modalities and, in addition, inferring back the new modality. To give just a few examples: when generating the motion of a letter, given its label, it will be more human-like and believable if the generated samples follow, to some extent, a random process; putting different types of items in a moving production line, the motion to place a specific item type, in the line, will not always be the same.

A previous study integrated sensory-motor time-series data in multimodal fused representations [41]. The proposed framework assumes that the motion modality, represented by the joint angles, is always presented. There could be other modalities complementing the motion modality, like the sound or image. For each optional modality, there is an Auto-Encoder (AE) network that compresses the corresponding modality. The main component is another AE that synchronizes the joint angles (motion) and the feature vectors of the optional modalities, having the capacity to reconstruct the input. It is possible to do cross-modal retrieval or temporal sequence prediction by creating a closed-loop in the main AE, where the network receives as new inputs the previous reconstructed outputs, after applying a specific shift of steps, for the retrieved, or all, modalities. However, such an approach is not fully resilient against uncertainty since it is always needed a batch of inputs containing all modalities. Also, these types of models cannot create novel instances, which could be helpful for the motion modality in certain types of problems.

Another work uses a multimodal architecture to perform cross-modal inference on visual and bathymetric data captured from an autonomous underwater vehicle [42]. The objective passes to correlate the information from the two sources to help the navigation, especially under uncertainty (missing modalities). First, this architecture learns a feature vector for each modality using a Denoising Auto-Encoder (DAE). Then a final layer learns a multimodal representation between all modalities using a mixture of Restricted Boltzmann Machines (RBMs) receiving, as input, the feature vectors previously learned. This multi-layer architecture can capture a joint distribution between all modalities and perform cross-modal inference (from one modality to the other). However, RBMs can be expensive, hard to train, and require a meticulous model design to maintain tractability.

3.4 Handwriting Generation

Most works on handwriting generation/synthesis only focus on writing individual letters. Such systems are far easier to implement than those that try to handwrite words or even sentences since there is no need to be concerned about connecting two consecutive letters or changing the trajectory motion of the following letter to be consistent with the connection between the two letters.

A recent work introduced a handwriting learning framework capable of making a physical robot learn how to write alphanumeric characters by imitating human examples [43]. The system used a modified version of the original DMP by changing the transformation system, constraining the force term to no longer be influenced by the goal position, making the framework work even when the goal position changes. The canonical system employs a linear decay system, and the force term uses truncated Gaussian kernels. The framework can also create new letters by changing the segmented learned strokes of the original sample. Still, this process is not fully automatic and learnable in an end-to-end manner.

Another work developed a framework to teach a robotic agent how to learn and synthesize handwriting motion from individual alphanumeric characters [44]. The framework uses an ensemble method and local cost function to learn local models that encode specific subsets of the data. Then, the handwriting motion of each sample is synthesized by sampling the kinematics feature space. This work also presents a user study proving that the synthesized samples from the framework are hard to distinguish from actual human handwriting samples.

Another recent work designed to handwrite individual letters and numbers used an MVAE model to learn how to associate the image and motion modality [30]. The model learns a joint representation between the image and motion of each letter, enabling the cross-generation of samples, e.g., infer the motion given an image of the character. The model could even generate the motion of a letter when given corrupted images, with a random quartile of the image covered.

The works presented so far were conceptualized to handle only handwriting generation of single characters, which is a far more straightforward task than generating the motion to write a word or a sentence. We now present works that also consider the handwriting synthesis of words/sentences. A former work, also introduced in Section 3.2, used a modified version of DMPs to convert and connect trajectories of individual letters [36]. The method uses a single set of overlapping kernels and a sigmoidal canonical system to address the problem of dynamically joining trajectories. However, the authors only tested the method with carefully chosen letter samples where each trajectory's initial and final portion have identical directions, which facilitates the joining motion.

A separate work performed handwriting synthesis using an online handwriting dataset containing thousands of English text sentences. The model is composed of an Recurrent Neural Network (RNN) component with an Mixture Density Network (MDN) component at the end. When given the model the online data points and high-level annotation sequence (character string), in the form of a soft window, the model can create novel handwriting motion given a new text sequence as input. The model also learned the various handwriting styles present in the dataset, resulting in realistic handwritten sequences. Because of the particular architecture of the network, it is complicated to extend this framework to handle multimodal information.

The PRG's illustration that we will present in handwriting, Chapter 5, differs from the works above since we, first, focus on handwriting words and not only individual letters. Second, we enable the use of different modalities to give as input to the system. This approach unlocks the interaction procedure via multiple communication channels and extends the system's practical use to more scenarios with different pre-requisites.

4 Methodology

Contents

In this work, we present a pipeline for HRI scenarios where the user controls the robotic agent by instructing commands, in our case, multimodal commands. The agent processes each command and infers a movement to execute that is adequate for the given command. One of our main focuses is creating a framework that makes the best use of multimodal information, trying to get an edge over frameworks that would do the same task but with only one modality. For this reason, we use an MVAE model to combine the received multimodal information to create a solid and robust environmental recognition system. This model can deduce a completer task description since it has more information from different modalities, which can also be helpful in the face of uncertainty.

To accomplish such advantages, we propose a novel three-stage pipeline titled Perceive-Represent-Generate (PRG) to address the problem above, as shown in Figure 4.1. In the first stage, Perceive, the agent collects multimodal information provided by the human user and processes the raw observation data transforming it into a sequence of sub-commands, one for each modality. In the second stage, Represent, the agent encodes the available information in a multimodal latent representation $\mathbf{z}_\pi$, using an MVAE. Finally, in the third stage Generate, the agent uses the same MVAE to decode the representation to generate the target modality, in our case motion, suitable for the robot's actuation.

The core of our framework is the MVAE model used in the second and third stages, as previously described. In a previous phase of the initialization of our pipeline, we train this generative model. The dataset used has samples with all modalities present, and they are highly similar to the sub-commands given to the model in each run of our pipeline. Therefore, when used in our pipeline, the model already learned how to correlate modalities and cross-generate them. Additionally, as briefly described above, we also process the perceived modalities before passing and retrieving them from the MVAE, which allows us to decouple what the agent perceives, with its sensors, and how it actuates, with its actuators, with what the MVAE model handles as input.

In the following sections, we discuss in detail each of these stages.

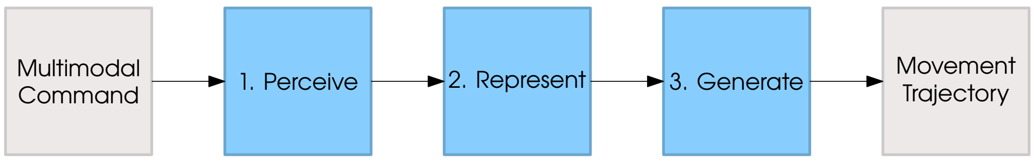


Figure 4.1: High level view of the proposed Perceive-Represent-Generate (PRG) pipeline for multimodal actuation. PRG is composed of three consecutive stages: Perceive, Represent, and finally Generate. The output of one stage is the input of the next one.

4.1 Perceive

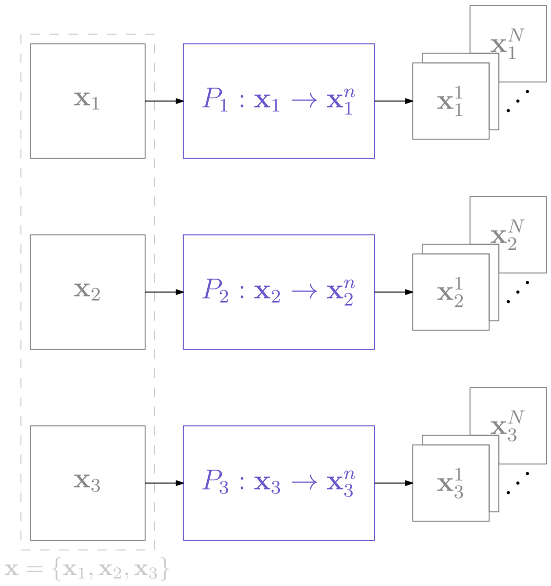
In the initial stage of our pipeline (Perceive), the agent collects the commands from the human user. In further detail, the agent's interaction cycle with the environment starts by collecting raw multimodal observations provided by the user. One or more agent sensors perceive these raw multimodal observations. The data that composes each modality's observation is independent of all others. Therefore, a multimodal command is simply an aggregation of detached observations, one for each perceived modality. Capturing individual information for each modality assists our overall purpose since: (i) We can perform a pre-processing stage before passing it to the generative process (Represent stage) to transform the incoming data into a format suitable for processing by the next stage. (ii) When one or more sensors fail, the impact caused by it is that the multimodal command will not contain the modality observations retrieved from those sensors. In such conditions, we say that the agent is operating under uncertainty where one or more modalities are missing.

Defining a mathematical notation, we assume that there are M one-way communications channels (modalities) $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$, where $\mathbf{x}_m \in \mathbb{R}^{d_m}$ and d_m is the dimension of $\mathbf{x}_m$. $\mathbf{x}_m$ is the information provided by the mth input communication channel. The user can give the agent commands directly related to its task through at least one communication channel, $\mathbf{x}_m$.

Next, we decompose the information received for each modality into a sequence of N sub-commands, $\mathbf{x}_m := \left(\mathbf{x}_m^1, \mathbf{x}_m^2, \ldots, \mathbf{x}_m^N\right)$, suitable to be given to the next phase. This decomposition occurs in this phase, where the framework receives the raw information of each modality, $\mathbf{x}_m$, and applies an already established transformation procedure that will decompose and output a sequence of individual and processed observations, x_m^n where $n \in \{1, 2, \ldots, N\}$, one resulting sequence for each modality. Explicitly, we define M perceptual maps $\mathcal{P} = \{P_1, P_2, \ldots, P_M\}$, one for each input modality, where each function maps the original observation into a sequence of individual and processed observations, $P_m : \mathbf{x}_m \mapsto \left(\mathbf{x}_m^1, \ldots, \mathbf{x}_m^N\right)$, see Figure 4.2 for a graphical representation.

Applying such perceptual maps gives several benefits to our framework. It makes our framework more generic and independent from the task to solve and facilitates the employment to more complex tasks or environments where there is, for example, a discrepancy between the granularity of the data received as input and what the generative model accepts. The perceptual maps help solve this issue by decoupling or grouping the perceived data from each modality before giving it to the second phase. Another potential case occurs when the collected unrefined data from one or more modalities contain excessive information, especially information that does not help the system to solve the underlying task in any way. As a result, it is possible to create perceptual maps that will handle the observations of each modality and retrieve and process only the relevant information suitable to solve the task, discarding, at the same time, excessive and inapplicable information. Such an approach also gives the possibility to reduce the dimensionality of the input modality data. Supplementarily, since the resulting sequence of sub-commands for each given modality contain items independent from each other, we can pass such elements iteratively and independently to the generative model in the next stage.

The proposed pipeline is agnostic to the nature and number of the perception maps $\mathcal{P}$ because they can be learned or heuristically defined. For instance, to process sound information, one can employ a generative model to encode a low-dimensional representation (in the next stage) or employ a pre-trained speech-to-text model, as a perceptual function, to retrieve semantic information and decompose the words into individual letters (such as wav2vec 2.0 [45]). Another illustration occurs when we do not want to process the given modality. In that case, one can define an identity map that returns the input embedded in a sequence with only that element, $P_m(\mathbf{x}_m) = (\mathbf{x}_m^1)$, where $\mathbf{x}_m = \mathbf{x}_m^1$.



$\mathbf{x}_1$
$P_1 : \mathbf{x}_1 \to \mathbf{x}_1^n$
$\mathbf{x}_1^1$
$\mathbf{x}_1^N$
$\mathbf{x}_2$
$P_2 : \mathbf{x}_2 \to \mathbf{x}_2^n$
$\mathbf{x}_2^1$
$\mathbf{x}_2^N$
$\mathbf{x}_3$
$P_3 : \mathbf{x}_3 \to \mathbf{x}_3^n$
$\mathbf{x}_3^1$
$\mathbf{x}_3^N$
$\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$

Figure 4.2: PRG's Perceive (1st) phase: PRG receives a multimodal command $\mathbf{x}$ composed of individual data of 3 modalities, $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$. Each modality is processed individually by its corresponding perceptual map P_i, where $i \in \{1, 2, 3\}$, and decomposed into a sequence of N individual and processed sub-commands, $\mathbf{x}_i^n$, where $n \in \{1, 2, \ldots, N\}$.

4.2 Represent

In the Represent stage, we receive the sub-commands from each modality and process them iteratively. In each iteration, we process the sub-commands, from every given modality, with the same superscript index. This process involves providing a generative model with the selected sub-commands. The generative model will associate and encode them into a latent space to refine and retrieve modality-specific and high-level information. Modality-specific information allows the regeneration of the corresponding modality from the latent space. High-level information is obtained by the model when it correlates two or more modalities and allows the cross-generation of those modalities. Thus, it is possible to obtain information for one or more specific modalities from other ones and still preserve the common features. With the described conception, it is possible to understand the task and environment better. Furthermore, such redundant and associative high-level information can be beneficial when only some of the modalities (corresponding sequences of sub-commands) are present, where it is still possible to infer the hidden modalities. Therefore, the framework is robust enough to continue operating even when several sensors fail by extreme usage, human error, or environmental conditions change.

Specifically, we intend to learn L representation maps $\mathcal{R} = \{R_1, R_2, \dots, R_L\}$, where each map $R_l : \mathrm{proj}_{e^l} \to \mathcal{Z}$, where $l \in \{1, 2, \dots, L\}$. At iteration n, each map takes as input a projection of the input space, $\mathrm{proj}_{e^l}(\mathbf{x}^n)$, where $\mathbf{x}^n = \{\mathbf{x}_1^n, \dots, \mathbf{x}_M^n\}$, $e^l \in \mathbb{N}^M \wedge (e_i^l = 0 \vee e_i^l = 1)$, $i \in \{1, 2, \dots, M\}$, and finally $\mathrm{proj}_{e^l}(\mathbf{x}^n) = \{\mathbf{x}_i | \mathbf{x}_i \in \mathbf{x}^n \wedge e_i^l = 1\}$. Then, the map converts the input into a multimodal latent value, $\mathbf{z}_\pi \in \mathcal{Z}$, where $\mathcal{Z}$ is the multimodal latent space. This input projection can consider only one or a combination of different modalities, which means that the number of representation maps is not necessarily equal to the number of input modalities (can be less or greater). In a specific run of our pipeline, we only chose one representation map to encode the given modalities, for that run, into the latent representation. Since all representation maps are attached to the multimodal latent space $\mathcal{Z}$, it will always be possible to build a latent value that can reconstruct every modality even if it was initially missing, no matter the representation map used. Figure 4.3 presents graphically this phase.

These mappings, $\mathcal{R}$, are learned by instantiating an unsupervised learning framework. In our case, we use and train an MVAE model. Typically, the encoders of the MVAE correspond to the representation maps $\mathcal{R}$ described above. We also define the decoders of the MVAE as generation maps $\mathcal{G} = \{G_1, G_2, \dots, G_L\}$, where each map, G_l, receives a latent space vector $\mathbf{z}_\pi$ and regenerates the modalities present in the projection given to the corresponding representation map R_l, $\mathrm{proj}_{e^l}(\mathbf{x}^n)$. Further, we assume that a previous procedure trains the MVAE model used in our framework with a dataset closely related to the sub-command inputs of all modalities. When we employ the MVAE model in our framework, we only utilize its inference method.

We want to emphasize further that our framework will typically work under uncertainty. Since the movement modality is the one we want to generate, the output of our pipeline, commonly this modality is missing and is not given as input, is instead the one we want to infer. As a concrete example, let us imagine our framework receives a subset of modalities as input. After passing it through the first phase, we chose a representation map that receives the most modalities given as input and none of the missing modalities, and, as explained above, we pass the sub-commands of every modality with the same index in the same iteration to the selected representation map. After this, we end up with, also, N latent space values, $\left(\mathbf{z}_\pi^1, \dots, \mathbf{z}_\pi^N\right)$, ready to pass it to the next stage.

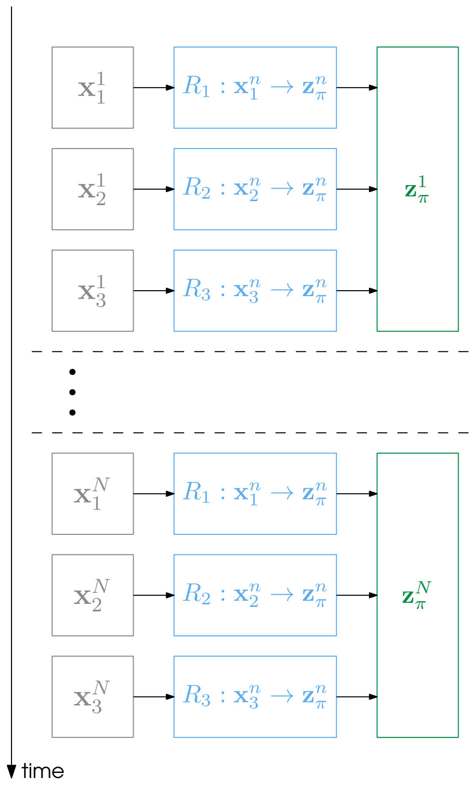


Figure 4.3: PRG's Represent (2nd) phase: The sequence of N sub-commands for each modality is received. The sub-commands are then passed iteratively to the generative model, particularly to its representations maps R, where, in the same iteration, we pass the sub-commands of all modalities with the same superscript index, $\{\mathbf{x}_1^n, \mathbf{x}_2^n, \mathbf{x}_3^n\}$. In this illustration, the generative model comprises three representation maps, where each map receives each input modality. For a selected representation map, the output is a multimodal latent value $\mathbf{z}_\pi$, which after all iterations, we have a sequence of N latent values, $\mathbf{z}_\pi^1, \ldots, \mathbf{z}_\pi^N$. Note that the number of representations maps does not have to be equal to the number of modalities. We could have a representation map that receives the first and second modality, $\{\mathbf{x}_1^n, \mathbf{x}_2^n\}$, at the same time. Furthermore, there might be missing modalities in a standard pass of our pipeline, and the pipeline only chooses one representation map to convert the input space into the multimodal latent space. Usually, the representation map that receives the most modalities from the ones given and none of the missing modalities is the one that applies the conversion.

4.3 Generate

In the last phase of our pipeline, Generate, we iteratively convert the sequence of latent spaces, given from the previous phase, $(\mathbf{z}_\pi^1, \ldots, \mathbf{z}_\pi^N)$ into a sequence of motions $(\mathbf{x}_t^1, \ldots, \mathbf{x}_t^N)$, where $\mathbf{x}_t^n \in \mathbf{x}^n$. As introduced in the previous section, Section 4.2, we use a generation map to perform this conversion. In this case, we use the generation map corresponding to the movement modality $G_t : \mathcal{Z} \to \mathbf{x}_t$, where $G_t \in \mathcal{G}$. We end up then with a sequence of N trajectories, $(\mathbf{x}_t^1, \ldots, \mathbf{x}_t^N)$. As explained in the previous section, this conversion is possible since the model can infer missing modalities from other ones (cross-generation). Then, it is possible to regenerate a modality not present in the input given to the pipeline.

Subsequently, we pass the sequence of trajectories through a final perceptual map, $P_t' : \mathbf{x}_t^n \mapsto \mathbf{x}_t^{n'}$, to transform them, if needed. We find it helpful to have a perceptual map, in this stage, to create complex trajectories suitable to solve the task/problem. It could happen that the trajectories returned from the movement's generation map, decoded by the generative model, do not consider what its contribution is to the final movement. We can imagine that the generative model outputs individual pieces of the final trajectory that must be further processed to construct the final and complete movement. Therefore, with this perceptual map, we can reconstruct the final movement from the sequence of trajectories. Practical transformations for this map are, for example, the resize and translation of trajectories. Finally, we concatenate all trajectories outputted by this final perceptual map and transform them into a single DMP. Having the final movement as a DMP provides several benefits, such as having a trajectory with a continuous velocity and acceleration. This DMP trajectory is the one that our pipeline will output, and it is ready to be executed by the robotic agent. See Figure 4.4 to have a general view of this phase. We also present the complete pipeline in Figure 4.5.

We would also like to emphasize that our pipeline is agnostic to the number of post-processing steps applied to the trajectory returned by our pipeline. We acknowledge that depending on the chosen robotic environment it could be necessary to transform the trajectory returned by our pipeline. For example, in some cases, it could be necessary to transform the trajectory into the joint space of that specific robot before execution.

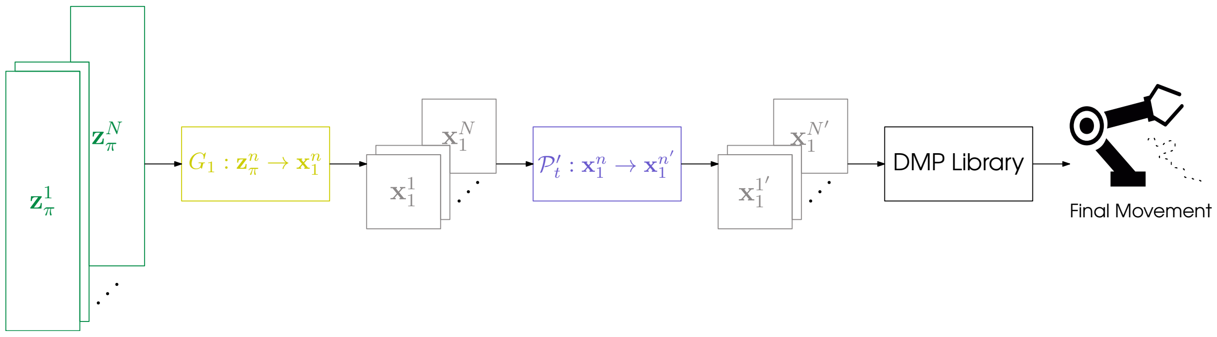


Figure 4.4: PRG's Generate (3rd) phase: The sequence of multimodal latent variables, $(\mathbf{z}_\pi^1, \ldots, \mathbf{z}_\pi^N)$, will be passed iteratively to the generation map of the movement modality, G_1. Here we are assuming that the movement modality is the first modality, $\mathbf{x}_1$. For each latent representation $\mathbf{z}_\pi^n$, the generation map regenerates the corresponding trajectory, $G_1(\mathbf{z}_\pi^n) = \mathbf{x}_1^n$. A final perceptual map, P_t', further processes each trajectory, $P_t'(\mathbf{x}_1^n) = \mathbf{x}_1^{n'}$. We then concatenate all processed trajectories, $\left(\mathbf{x}_1^{1'}, \ldots, \mathbf{x}_1^{N'}\right)$, and convert them into a DMP, ready to be executed by the robotic agent.

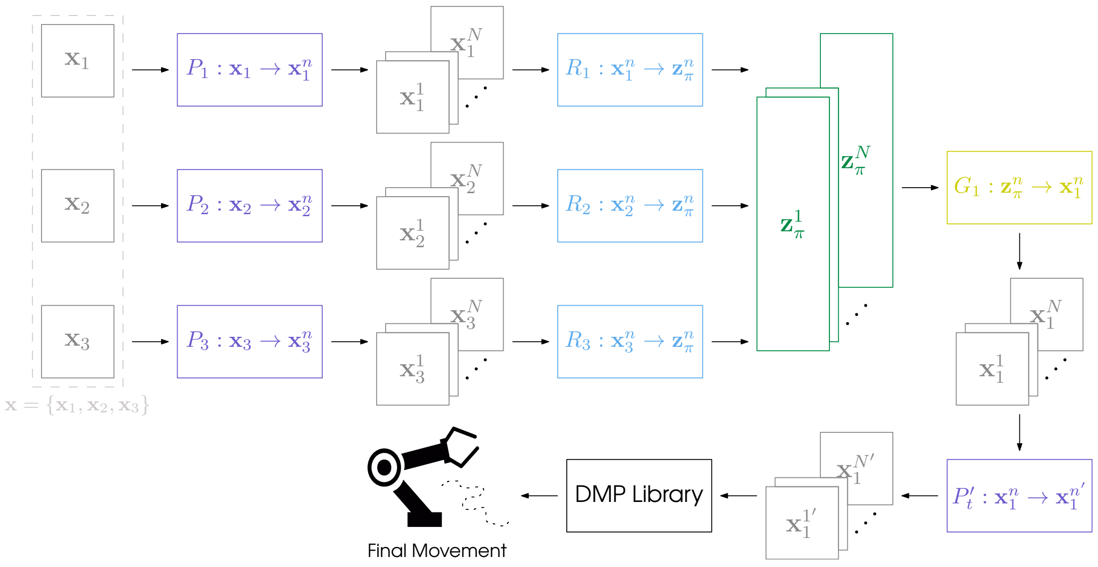

$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$
$P_1 : \mathbf{x}_1 \to \mathbf{x}_1^n$
$P_2 : \mathbf{x}_2 \to \mathbf{x}_2^n$
$P_3 : \mathbf{x}_3 \to \mathbf{x}_3^n$
$R_1 : \mathbf{x}_1^n \to \mathbf{z}_\pi^n$
$R_2 : \mathbf{x}_2^n \to \mathbf{z}_\pi^n$
$R_3 : \mathbf{x}_3^n \to \mathbf{z}_\pi^n$
$\mathbf{z}_\pi^1$
$\mathbf{z}_\pi^N$
$G_1 : \mathbf{z}_\pi^n \to \mathbf{x}_1^n$
$P_t' : \mathbf{x}_1^n \to \mathbf{x}_1^{n'}$
DMP Library
Final Movement

Figure 4.5: The proposed Perceive-Represent-Generate (PRG) pipeline for multimodal actuation. In the Perceive phase, the model collects commands from the human user through multimodal information $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$ (we assume the movement modality is $\mathbf{x}_1$) and performs an initial preprocessing to efficiently compress raw observation data and isolate individual commands $\mathbf{x}_m^n, n \in \{1, 2, \ldots, N\}$ and $m \in \{1, 2, 3\}$, using the perception maps P. In the Represent phase, we employ the representation maps R, provided by the multimodal generative model, to encode the individual commands $\mathbf{x}_m^n$ in a latent representation $\mathbf{z}_\pi^n$, suitable to be encoded from partial observations. In the Generate phase the model decodes each latent representation using the target generation map G_1, processes them using P_t', and merges the individual motion information into a DMP to generate target trajectory data $\mathbf{x}_1$.

5 Evaluation

Contents

In this section, we present an instantiation of our proposed framework, PRG, in the context of the Robotic Dictaphone scenario, where the goal of the robot is to generate handwritten word samples from information provided by the human user through speech or by giving a sequence of images. The ith image contains the ith letter of the word to write. We conduct a quantitative and qualitative evaluation of the generative capability of our model regarding its ability to map observations to a latent representation and show that our approach is robust to missing modality information. Finally, we present qualitative samples from handwritten words generated from different implementations of our pipeline, showcasing our pipeline's effectiveness as a whole.

5.1 Scenario

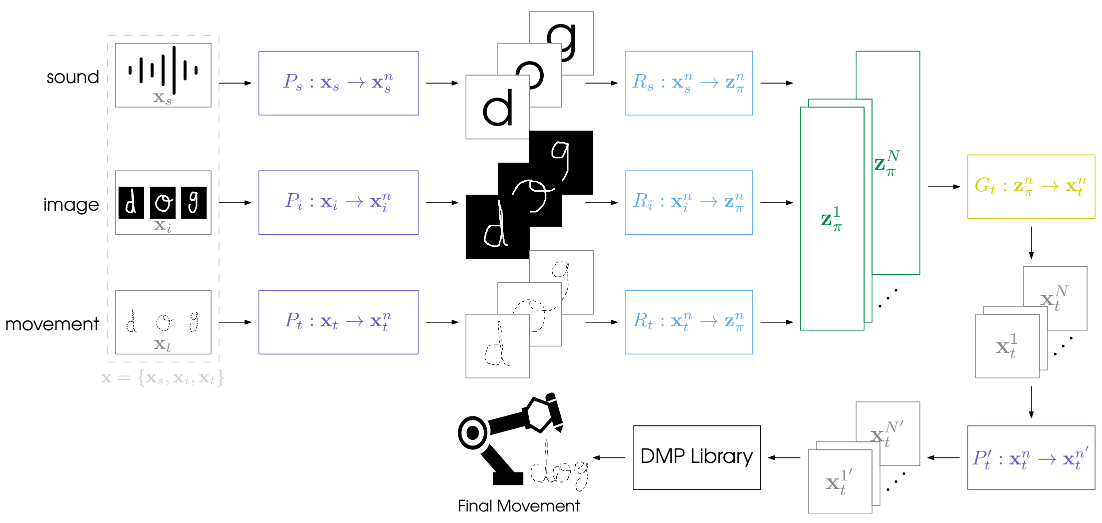
In this work, we consider the following scenario of a Robotic Dictaphone. A human provides a word to be transcribed by the robot through different communication channels $\mathcal{X} = \{\mathbf{x}_s, \mathbf{x}_i, \mathbf{x}_t\}$. Specifically, $\mathbf{x}_s$ is the sound produced to say the corresponding word. $\mathbf{x}_i$ is a sequence of images, where the ith image is a handwritten representation of the ith letter of the same word. $\mathbf{x}_t$ is a sequence of 2D motions, where the ith motion is a handwritten representation of the ith letter of the corresponding word.

The agent aims is to map such multimodal information in an internal representation, suitable for the downstream generation of coherent samples of the handwritten words. However, the human user may only employ a subset of such communication channels to provide the goal words despite being available. As such, the agent must learn to encode a representation that is robust to partial observations. This requirement is essential since the movement modality will be missing in a typical run of our pipeline since that is the modality we want to infer from the other modalities or a subset of them.

We implement 4 different instantiations of our pipeline for the case of the Robotic Dictaphone that differ on the number of communication channels used and the type of the MVAE model used in the Represent and Generate stages. We now describe in detail each of the stages.

5.1.1 Perceive

In this first stage, the raw observation data provided by the human user is processed and decomposed into sub-commands, in this case, word's letters. To do so, we define the perception maps specific to each communication channel $\mathcal{P} = \{P_s, P_i, P_t\}$. For the sound perception map P_s, we employ wav2vec 2.0, a self-supervised learning framework for speech recognition [45]. As such, we process the raw audio data into the label information (character) associated with each letter, allowing for more efficient downstream representation, capturing only the relevant information to solve our task and discarding the rest. For the imaging modality, we utilize a function map, P_i, that unpacks a given input sequence. $\mathbf{x}_i$ is a sequence of image tensors and, when given to P_i, we get $\mathbf{x}_i^n$, where $n \in \{1, 2, \ldots, N\}$, N is the number


sound
$\mathbf{x}_s$
image
$\mathbf{x}_i$
movement
$\mathbf{x}_t$
$\mathbf{x} = \{\mathbf{x}_s, \mathbf{x}_i, \mathbf{x}_t\}$
$P_s : \mathbf{x}_s \rightarrow \mathbf{x}_s^n$
$P_i : \mathbf{x}_i \rightarrow \mathbf{x}_i^n$
$P_t : \mathbf{x}_t \rightarrow \mathbf{x}_t^n$
$R_s : \mathbf{x}_s^n \rightarrow \mathbf{z}_\pi^n$
$R_i : \mathbf{x}_i^n \rightarrow \mathbf{z}_\pi^n$
$R_t : \mathbf{x}_t^n \rightarrow \mathbf{z}_\pi^n$
$\mathbf{z}_\pi^1$
$\mathbf{z}_\pi^N$
$G_t : \mathbf{z}_\pi^n \rightarrow \mathbf{x}_t^n$
$\mathbf{x}_t^1$
$\mathbf{x}_t^N$
$P_t' : \mathbf{x}_t^n \rightarrow \mathbf{x}_t^{n'}$
$\mathbf{x}_t^{1'}$
$\mathbf{x}_t^{N'}$
DMP Library
Final Movement
Figure 5.1: Implementation of PRG for the Robotic Dictaphone scenario. The pipeline receives three modalities, $\mathcal{X} = \{\mathbf{x}_s, \mathbf{x}_i, \mathbf{x}_t\}$: the sound produced to say a word, $\mathbf{x}_s$, the images of the handwritten letter that compose the same word, $\mathbf{x}_i$, and the trajectories of the same letters, $\mathbf{x}_t$. The perceptual maps for the image and trajectory modalities, P_i and P_t, will unpack and return the corresponding sequence of images and trajectories, respectively. For P_s, we employ wav2vec 2.0 model that will convert the sound wave into the characters that compose the underlying word. The given modalities for a specific run are given to one of the representation maps R to derive a multimodal latent value, $\mathbf{z}_\pi^n$. The generation map of the trajectory modality, G_t, converts the sequence of multimodal representations into the motions of each letter, $\mathbf{x}_t^n$. The regenerated letter trajectories are then passed to the final perceptual map, P_t', to be translated, resized, and to create intermediate (connection) trajectories connecting two consecutive letters. Finally, these trajectories, $\mathbf{x}_t^{n'}$, are concatenated and converted into a single DMP to be executed by the robotic agent. Note that the number of representation maps and what each receives as input are illustrative as it will depend on the MVAE model used.

of letters of the underlying word, and $\mathbf{x}_i^n$ is then the image tensor that corresponds to the nth letter of the underlying word. For P_t, we also unpack the sequence of letter trajectories and then normalize it, given a pre-computed mean and variance from the training dataset (used to train each multimodal VAE employed in the next stage).

5.1.2 Represent

In this stage, an MVAE model maps the observations corresponding to the sequence of sub-commands to the multimodal latent space $\mathbf{z}_\pi$. We employ and compare different MVAE models to learn such representation due to their low memory requirements and stable training. In particular, we implement four instantiations of our pipeline that differ in the representation model to use, each able to account for a different number (or combination) of modalities:

- $R_{\text{CVAE}}(\mathbf{x}_s, \mathbf{x}_t)$, PRG pipeline using CVAE model in the Represent and Generate stages. With CVAE we are able to generate trajectory information $\mathbf{x}_t$ conditioned on sound information $\mathbf{x}_s$. The model uses the loss function (2.6).

- $R_{\text{AVAE}}(\mathbf{x}_s, \mathbf{x}_t)$, this implementation of the PRG pipeline employs the AVAE model in the Represent and Generate stages. The AVAE model learns a joint representation of trajectory information $\mathbf{x}_t$ and sound information $\mathbf{x}_s$. The model uses (2.7) for training.

- $R_{\text{AVAE}}(\mathbf{x}_i, \mathbf{x}_t)$, PRG's Represent and Generate phases use the AVAE model. In this implementation, the AVAE model uses the trajectory, $\mathbf{x}_t$, and image, $\mathbf{x}_i$, information to learn a joint multimodal representation. The training of AVAE follows (2.7).

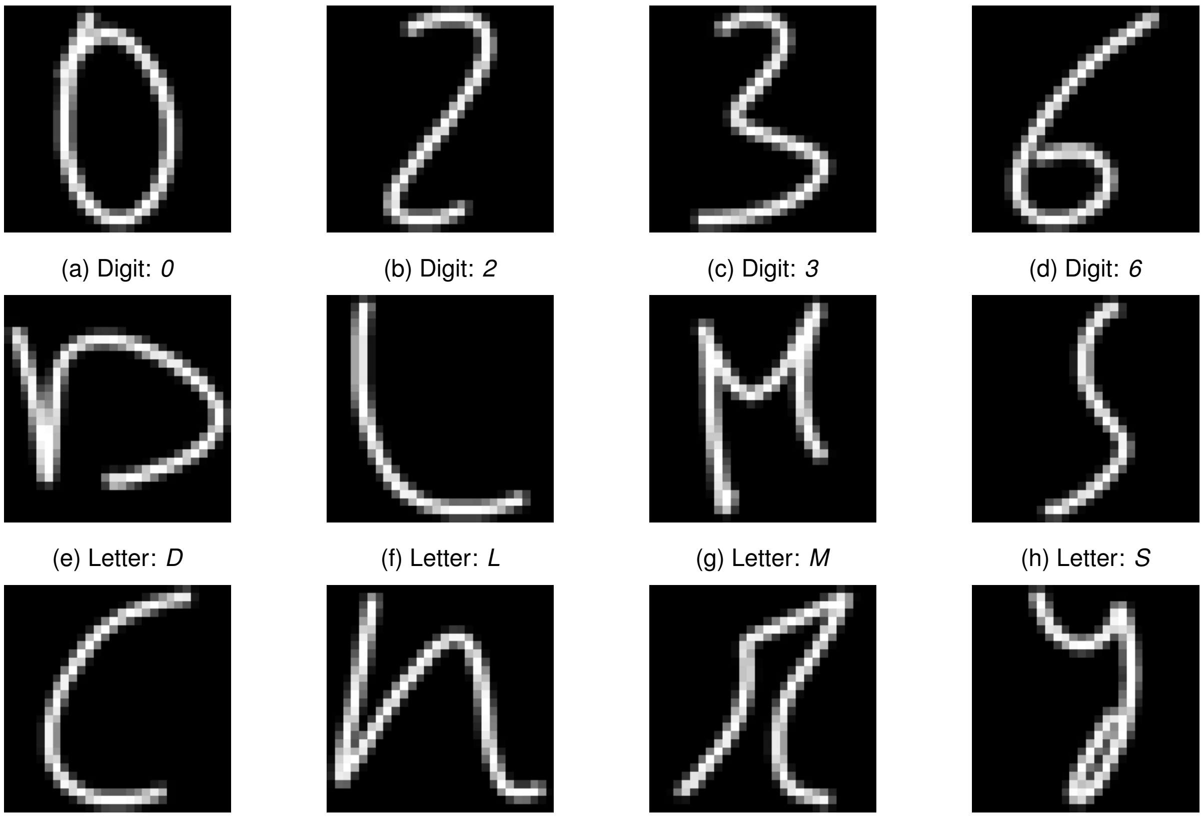
- $R_{\text{MUSE}}(\mathbf{x}_s, \mathbf{x}_t, \mathbf{x}_i)$, here we utilize the MUSE model in the Represent and Generate phases. For MUSE, we give it all modalities, $\mathbf{x}_t$, $\mathbf{x}_s$, and $\mathbf{x}_i$, to learn a joint representation. This model uses (2.11) for training.

We chose the CVAE model since it is one of the simplest models that allow multimodal inference. The model operates with only two input modalities. We expect to get high-performance levels since the model only learns one common latent space, given that the encoder's input concatenates the inference modality, $\mathbf{x}_t$, and the condition modality, $\mathbf{x}_s$. The same occurs before giving the latent values to the decoder. The downsides are that the model cannot extend to more than two modalities and cross-generation of modalities occurs only in one way, from $\mathbf{x}_s$ to $\mathbf{x}_t$.

Typically, CVAE's condition modality is a categorical variable that has a low dimension. To extend the cross-generation with high-dimensional modalities, we also implement the AVAE model. AVAE model has an individual VAE for each modality allowing for the modality itself to have any dimension. This model also enables performing cross-modality inference from any combination of modalities to any other combination. As explained in Section 2.1.3, this model still has some problems since extending it to more than two modalities implies a combinatorial explosion of parameters due to the necessity to have a different encoder for each distinct combination of modalities. We expect a lower performance for the same modalities as CVAE since each modality encodes an individual latent space. Subsequently, an additional KL divergence term approximates the latent spaces.

To overcome all the issues presented above, we use the MUSE model in our last implementation. In MUSE, the number of parameters scales linearly with the number of modalities since we only need 2 encoders and 2 decoders for each modality, 1 of each per level (low and high level), see Section 2.1.4. We give as input to this model all three modalities. Thus, we expect a slightly inferior performance to AVAE since we must consider and associate more information from different modalities. Another possible contribution to the decrease in performance is that we also have an extra level of indirection since the high-level associates the information from all modalities receiving their individual latent spaces as input (coming from the low-level latent spaces).

Before employing the generative models presented above in the PRG pipeline, we train them on the "UJI Pen Char 2" dataset [46]. For this dataset, we use only one-stroke samples for all letters and numbers. Using the chosen samples is enough to have a complete evaluation of the pipeline. The objective is to output one-stroke words so the robotic agent can perform the motion in one pass without lifting the pen. Moreover, this also facilitates the implementation of the generative models since we do not have to consider using RNNs because all samples will have the same dimensions. Also, this way, all input dimensions are relevant and can be treated similarly for all given samples. Plus, to address the small number of data points presented in the dataset, we learn a probabilistic model of each character and re-sample with perturbations constrained in a kinematics feature space, following the procedure described in [47]. This way, we generate more than $60\,000$ samples of 28×28 grayscale images and 200-dimensional representations of the associated trajectories, corresponding to 62 different classes (uppercase and lowercase English letters and all digits), which are also the three modalities used in PRG after the first stage. In addition, we standardize all trajectories. We present samples from the extended dataset in Figure 5.2.



(a) Digit: *0* (b) Digit: *2* (c) Digit: *3* (d) Digit: *6*

(e) Letter: *D* (f) Letter: *L* (g) Letter: *M* (h) Letter: *S*

(i) Letter: *c* (j) Letter: *n* (k) Letter: *r* (l) Letter: *y*

Figure 5.2: Image samples of handwritten digits and letters randomly retrieved from the extended “UJI Char Pen 2” dataset employed to train the MVAE models used in the Robotic Dictaphone scenario.

5.1.3 Generate

In this final stage, we use the movement's generation map to decode the collected latent representations into individual motion trajectories, one for each letter. We then apply the final perceptual map, $\mathcal{P}'_t$, to the sequence of motions which is composed of the following steps:

(1) We start by first defining a new size unit which we call a square. The side of this square has a predefined length l;

(2) We then define the size of each letter trajectory so that it has the expected proportion in the final word. Heuristically, we specify that every uppercase and lowercase letter has the size of 2 squares vertically aligned, except for the lowercase letters $\{\mathsf{a}, \mathsf{c}, \mathsf{e}, \mathsf{i}, \mathsf{m}, \mathsf{n}, \mathsf{o}, \mathsf{r}, \mathsf{s}, \mathsf{u}, \mathsf{v}, \mathsf{w}, \mathsf{x}, \mathsf{z}\}$ that only have a size of 1 square. Therefore, we re-scale every letter trajectory to have the determined size;

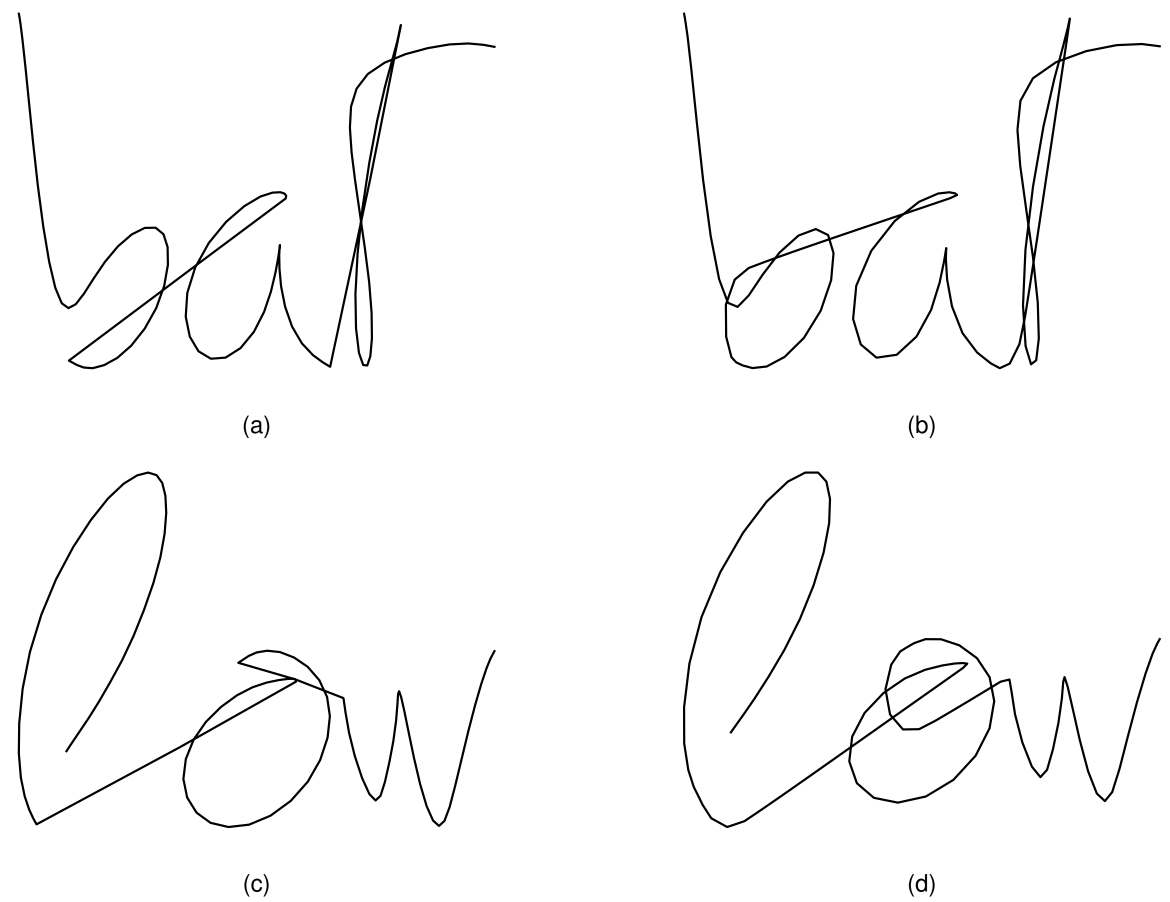
(3) We also translate each letter's trajectory vertically. Heuristically, every letter must start at the origin, $y = 0$, except for the $\{\mathsf{f}, \mathsf{g}, \mathsf{j}, \mathsf{p}, \mathsf{q}, \mathsf{y}\}$ letters which begin at $y = -l$, where l is the length of the square computed in (1);

(4) We define a fixed horizontal distance that will separate any two consecutive letter trajectories. This distance is a positive proper fraction of the length of our square's side, l;

(5) In the final processing step, we use a custom algorithm to exploit the space between each letter trajectory to derive a partial motion to connect the two letters. We chose this approach over using an algorithm that merges movements following a straight line. Our algorithm, instead, computes such intermediate trajectory iteratively, considering several aspects in each increment, resulting in a more natural and smooth handwriting motion, see Algorithm 5.1 for more details. We also present an illustration of how this algorithm works in Figure 5.4.

After all processing steps of the final perceptual map, the model has n letter trajectories and $n - 1$ connection trajectories. To finish the third stage and the pipeline, we concatenate all the letter and connection trajectories and convert them into a single DMP: the ith connection trajectory appears between the ith and $(i + 1)$th letter trajectories. In the end, we have one DMP, suitable for the robot's actuation to write the target word.

The steps of perceptual map $\mathcal{P}'_t$ presented above, Items (1) to (5), are strictly necessary to create a natural and human-like movement to handwrite a word that is easy to read. Since the generative models, in the previous stage, work on the character level (inputs and outputs modalities encode information of individual characters), the motions returned by the generation map do not consider their contributions to the final movement. With this in mind, we heuristically designed the first 4 steps, Items (1) to (4), of $\mathcal{P}'_t$ to scale and align every letter to write the underlying word. In Item (5), our algorithm smoothens the motion between letters, creating a more regular and eligible final movement. Especially without our algorithm, if we only concatenate the letter trajectories into a DMP or even if we use a DMP joint algorithm, the lines produced between letters become too straight, having abrupt changes in the velocity when transitioning between different trajectories. This simple approach turns the final movement unnatural and hard to read. Figure 5.3 presents two handwritten words, showing samples that include and not Algorithm 5.1 in $\mathcal{P}_t'$. We see that our algorithm smoothens the motion when transitioning between different trajectories making the final movement more natural, like a human sample, and easy to read.


(a)

(b)

(c)

(d)

Figure 5.3: Implementation of PRG for the Robotic Dictaphone scenario, (3rd) stage: word samples outputted giving the same letter trajectories to the final perceptual map $\mathcal{P}_t'$: for (a) and (b); and for (c) and (d). For the left-hand samples, (a) and (c), $\mathcal{P}_t'$ does not include Algorithm 5.1 (its last step, Item (5)). The other samples, (b) and (d), were generated with the original perceptual map, $\mathcal{P}_t'$, presented in Section 5.1.3.

Algorithm 5.1: Algorithm to create an intermediate (connection) trajectory between two consecutive trajectories.

```
Input: $\mathbf{x}_T^i$, $\mathbf{x}_T^{i+1}$ - Motion trajectories;
    $\delta$ - small trajectory increment;
    $\alpha_{\text{max}}$ - Maximum angular difference allowed between vectors;
    $\theta_o$ - Penalization cost over new connection points based on the index of the point in $\mathbf{x}_t^{i+1}$;
    $\theta_a$ - Penalization cost over the angle between the last connection trajectory vector and
    the new candidate vector to be included in the connection trajectory;
    $\theta_f$ - Penalization cost over the distance between new candidate point and the initial point
    of $\mathbf{x}_T^{i+1}$;
    $\theta_p$ - Penalization cost over the distance between new candidate point and the point in
    $\mathbf{x}_T^{i+1}$ to connect (target).
Result: Trajectory, $\mathbf{x}_I^i$, connecting $\mathbf{x}_T^i$ to $\mathbf{x}_T^{i+1}$
$\hat{x} \leftarrow \langle 1, 0 \rangle$;
$N_i \leftarrow length(\mathbf{x}_T^i)$;
$N_{i+1} \leftarrow length(\mathbf{x}_T^{i+1})$;
$\mathbf{x}_I \leftarrow []$;
$c \leftarrow \mathbf{x}_T^i[N-1] - \mathbf{x}_T^i[N-2]$;
$costs \leftarrow [0, \ldots, 0], costs \in \mathbb{R}^{N_{i+1}}$;
$angles \leftarrow [0, \ldots, 0], angles \in \mathbb{R}^{N_{i+1}}$;
$candidates \leftarrow [0, \ldots, 0], candidates \in \mathbb{R}^{N_{i+1}}$;
do
    for $j \leftarrow 0$ to $N_{i+1} - 1$ do
        $\alpha \leftarrow \min\left(\alpha_{\text{max}}, \angle(\mathbf{x}_T^{i+1}[j], \hat{x}) - \angle(c, \hat{x})\right)$;
        $angles[j] \leftarrow \alpha$;
        $candidates[j] \leftarrow c + \delta \cdot \langle \cos(\alpha), \sqrt{1 - cos^2(\alpha)} \rangle$;
    for $j \leftarrow 0$ to $N_{i+1} - 1$ do
        $costs[j] \leftarrow \theta_o \left(\frac{j}{N_{i+1}}\right) + \theta_a \left(\frac{angles[j] - \min(angles)}{\max(angles) - \min(angles)}\right)$;
        $costs[j] \leftarrow costs[j] + \theta_f \left(\frac{\|\mathbf{x}_T^{i+1}[0] - candidates[j]\| - \min_k(\|\mathbf{x}_T^{i+1}[0] - candidates[k]\|)}{\max_k(\|\mathbf{x}_T^{i+1}[0] - candidates[k]\|) - \min_k(\|\mathbf{x}_T^{i+1}[0] - candidates[k]\|)}\right)$;
        $costs[j] \leftarrow costs[j] + \theta_t \left(\frac{\|\mathbf{x}_T^{i+1}[j] - candidates[j]\| - \min_k(\|\mathbf{x}_T^{i+1}[k] - candidates[k]\|)}{\max_k(\|\mathbf{x}_T^{i+1}[k] - candidates[k]\|) - \min_k(\|\mathbf{x}_T^{i+1}[k] - candidates[k]\|)}\right)$;
    $j \leftarrow \arg\min(costs)$;
    $c \leftarrow candidates[j]$;
    $\mathbf{x}_I \leftarrow [\mathbf{x}_I, c]$;
while $\|\mathbf{x}_T^{i+1}[j] - c\| > \delta$;
```

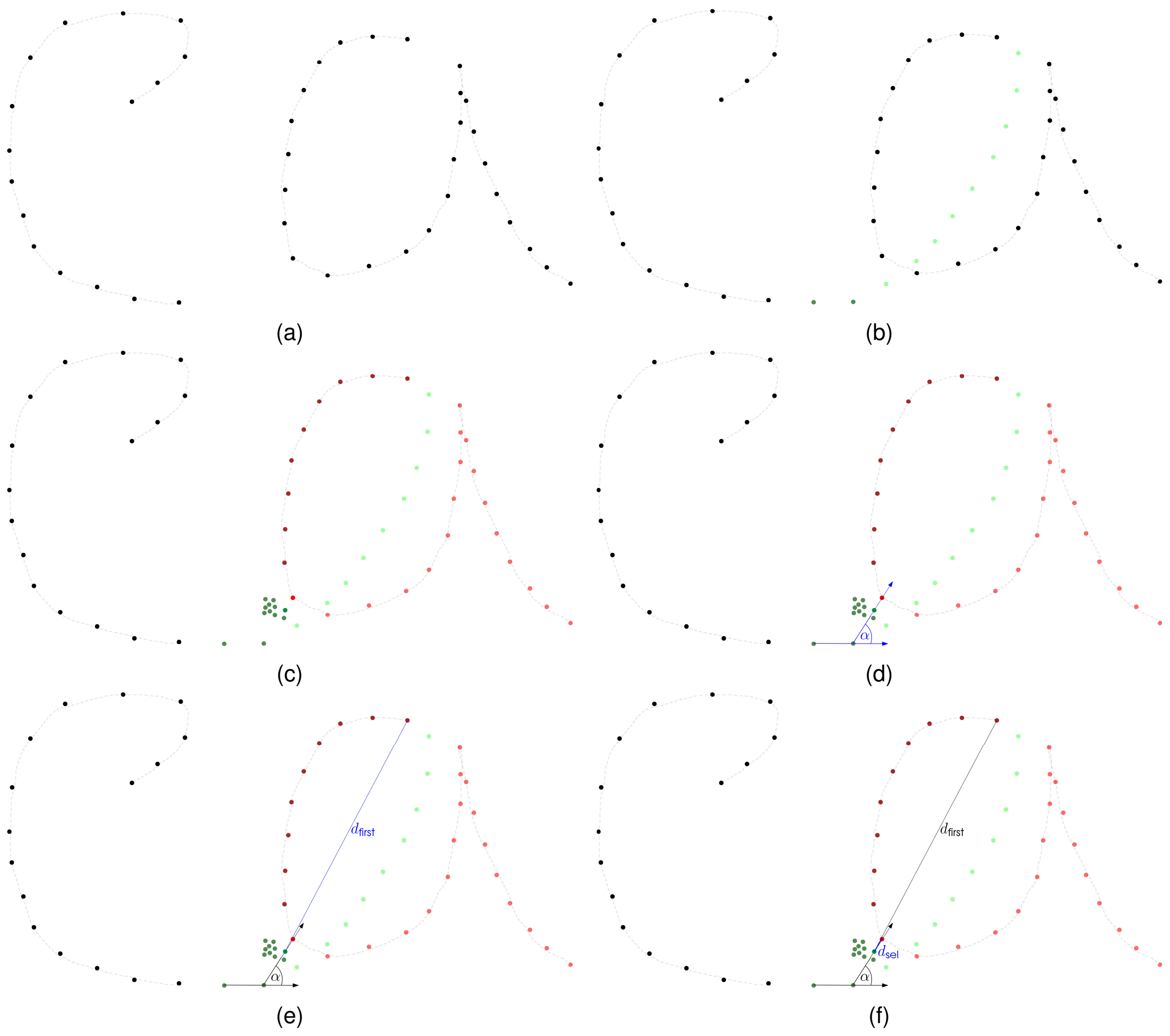


Figure 5.4: Illustration of Algorithm 5.1. (a): we present the original trajectories (black points) for two consecutive letters of a word, in this case, “c” and “a”; (b): Our Algorithm 5.1 will derive an intermediate trajectory to join both trajectories (dark and light green points). This algorithm is iterative, and at each iteration, we estimate a new point of the intermediate trajectory. Here, (b), we finish the second iteration since we have two computed points (dark green points) of the intermediate trajectory; (c): At each iteration, we produce a candidate point for each point of the consequent trajectory, in this case, “a”. We compute a cost for each candidate point and select the one with the lowest cost to be the final estimation point for this iteration. The cost for each candidate point considers the angle between the vector composed from the last two points of the intermediate trajectory and the vector that includes the current candidate point and the point of the subsequent trajectory, “a”, that it is trying to connect, as demonstrated in (d). This cost also has in consideration the distance between the current candidate point and the first point of the following trajectory, “a”, see (e). Similarly, the algorithm also ponders, in this cost, the distance between the current candidate point and the point of the subsequent trajectory that it is trying to join, (f). Finally, the index of the same point of the following trajectory, “a”, also weights the cost.

5.2 Results

In this section, we present the evaluation results of our framework, regarding the performance of the Represent stage (Section 5.1.2), Generate stage (Section 5.1.3), and handwritten samples generated by the complete pipeline (Section 5.2.3). We present quantitative and qualitative results to attest the following: (i) our pipeline allows the effective representation of multimodal information, being robust to missing information; (ii) our pipeline enables the generation of coherent and high-quality samples of handwritten words.

5.2.1 Represent Stage

We start by presenting the marginal and conditional log-likelihoods for all generative models employed. For AVAE and MUSE models, we compute the marginal log-likelihood with:

$$\log p\left(\mathbf{x}_j\right) \approx \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_j)}\left[\log \frac{p\left(\mathbf{x}_j|\mathbf{z}\right) p\left(\mathbf{z}\right)}{q\left(\mathbf{z}|\mathbf{x}_j\right)}\right] \tag{5.1}$$

using 1000 importance-weighted samples, and where $\mathbf{x}_j$ denotes one possible input modality. For the same models, we resort to the following equation to compute the conditional log-likelihood:

$$\begin{aligned}
\log p\left(\mathbf{x}_j|\mathbf{x}_k\right) &\approx \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_k)}\left[\log \frac{p\left(\mathbf{x}_j,\mathbf{z}|\mathbf{x}_k\right)}{q\left(\mathbf{z}|\mathbf{x}_k\right)}\right] \\
&= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_k)}\left[\log \frac{p\left(\mathbf{x}_j|\mathbf{z}\right) p\left(\mathbf{x}_k|\mathbf{z}\right) p\left(\mathbf{z}\right)}{q\left(\mathbf{z}|\mathbf{x}_k\right)}\right] - \log p\left(\mathbf{x}_k\right) \\
&= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_k)}\left[\log \frac{p\left(\mathbf{x}_j|\mathbf{z}\right) p\left(\mathbf{x}_k|\mathbf{z}\right) p\left(\mathbf{z}\right)}{q\left(\mathbf{z}|\mathbf{x}_k\right)}\right] - \log \frac{1}{N_c}\sum_{l=1}^{N_c} p_\theta\left(\mathbf{x}_k|\mathbf{z}^{(l)}\right)
\end{aligned} \tag{5.2}$$

where $\mathbf{x}_k$ is a different input modality from $\mathbf{x}_j$. We set N_c to 5000 importance-weighted samples and $\mathbf{z}^{(l)} \sim p\left(\mathbf{z}\right)$. For the CVAE model we can only compute the conditional log-likelihood using the following multiple lower bound:

$$\log p\left(\mathbf{x}_j|\mathbf{x}_k\right) \approx \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_j,\mathbf{x}_k)}\left[\log \frac{p\left(\mathbf{x}_j|\mathbf{z},\mathbf{x}_j\right)}{q\left(\mathbf{z}|\mathbf{x}_j,\mathbf{x}_k\right)}\right] \tag{5.3}$$

also using 1000 importance-weighted samples. The results are present in Table 5.1.

In addition, we present two complementary metrics to assess the cross-modality generation performance of the generative models: accuracy (we use a variation of [33]) and modality-distance. The former evaluates if the samples generated by cross-modality inference are semantically coherent with the available modality data provided to the model, e.g., a classifier should classify generated images from label "c" as image samples of that letter. The latter evaluates if the generated data by cross-modality inference are similar to the original samples in the dataset, e.g., generated images from label "c" should account for the different ways an image of that letter can be handwritten.

The evaluation procedure is as follows:

- **Accuracy**: First, for each modality, other than the label of each sample, we train a classifier on the training dataset. Next, we draw every possible combination of modalities, for each test sample, given the difference between all modalities and the target modality to cross-generate the target modality 100 times. Finally, we pass the generated modality samples to the respective classifier and compute the accuracy. When the target modality is $\mathbf{x}_s$, the label, we check if the generated label is the same as the actual label of the sample. The results are in Table 5.2;

- **Modality-Distance**: The first step of the procedure passes to create and train K AEs, where K is the number of the different labels in "Uji Pen Char 2" dataset. For each label k, $k \in \{1, 2, \ldots, K\}$, we train an AE giving only training samples having label k. Afterward, for each test sample, every possible combination of modalities is drawn from the difference between all modalities (that the generative model receives as input) and the target modality. Each combination of modalities is then passed to the MVAE model being tested to cross-generate the target modality, repeating 100 times. We then encode the true target modality of each sample (present in the original dataset) and the target modalities generated from cross-modal inference, resulting in a distribution of real dataset modalities, $\mathcal{N}\left(\mu_r^k, \Sigma_r^k\right)$, and generated modalities, $\mathcal{N}\left(\mu_g^k, \Sigma_g^k\right)$. Finally, we compute the modality distance from both distributions given by the Frechét distance, averaging over all labels:

$$D = \frac{1}{K} \sum_{k=1}^{K} \|\mu_r^k - \mu_g^k\| + \mathrm{Tr}\left(\Sigma_r^k + \Sigma_g^k - 2\sqrt{\Sigma_r^k \Sigma_g^k}\right) \tag{5.4}$$

  Table 5.3 presents the results of this evaluation procedure.

The results reveal an evident balance in performance between the intrinsic performance of the representation models and scalability to a higher number of modalities. Even though R_{CVAE} has a slightly worse conditional log-likelihood $\log p\left(\mathbf{x}_t|\mathbf{x}_s\right)$ than $R_{\text{AVAE}}\left(\mathbf{x}_s, \mathbf{x}_t\right)$, the accuracy and modality-distance results (Tables 5.2 and 5.3, respectively) are far better than both $R_{\text{AVAE}}\left(\mathbf{x}_s, \mathbf{x}_t\right)$ and R_{MUSE}. However, R_{CVAE} cannot learn a joint representation from multiple modalities nor scale to a higher number of modalities. The model can only perform a one-way cross-inference procedure from the conditional modality, in this case, $\mathbf{x}_s$ to the target modality $\mathbf{x}_t$. In contrast, both the R_{AVAE} models and R_{MUSE} model can learn a quality joint-modality latent representation. Table 5.1 shows that both R_{AVAE} models have better likelihood metrics than R_{MUSE}, except for $\log p\left(\mathbf{x}_s|\mathbf{x}_t\right)$, for $R_{\text{AVAE}}\left(\mathbf{x}_s, \mathbf{x}_t\right)$, and $\log p\left(\mathbf{x}_i|\mathbf{x}_t\right)$, for $R_{\text{AVAE}}\left(\mathbf{x}_i, \mathbf{x}_t\right)$. Still, the results for this metric are very similar. $R_{\text{AVAE}}\left(\mathbf{x}_s, \mathbf{x}_t\right)$ and $R_{\text{AVAE}}\left(\mathbf{x}_i, \mathbf{x}_t\right)$ outperform R_{MUSE} in all modality-distance evaluations and the accuracy metrics when giving only one modality as input, see Tables 5.2 and 5.3. The exception is when classifying $\mathbf{x}_t$ giving as input $\mathbf{x}_s$, for $R_{\text{AVAE}}\left(\mathbf{x}_s, \mathbf{x}_t\right)$ (which could be because we provide the same weight for the associative KL and individual VAEs, $\alpha = 1$ in

(2.7)). Yet, both R_{AVAE} models are unable to learn a multimodal latent space for more than two modalities without combinatorially exploding the number of parameters (see Section 2.1.3). On the other hand, R_{MUSE} can compositionally account for the information provided by multiple modalities to encode a more robust latent representation, as shown in Table 5.2 from the increased, or maintaining similar, accuracy performance when we provide more than one modality to the model.

Table 5.1: Log-likelihood metrics for the generative models R in the extended “UJI Char Pen 2” dataset.

(a) $R_{\text{CVAE}}(\mathbf{x}_s, \mathbf{x}_t)$

$\log p(\mathbf{x}_t \vert \mathbf{x}_s)$
-192.75

(b) $R_{\text{AVAE}}(\mathbf{x}_s, \mathbf{x}_t)$

$\log p(\mathbf{x}_t)$	$\log p(\mathbf{x}_s)$	$\log p(\mathbf{x}_t \vert \mathbf{x}_s)$	$\log p(\mathbf{x}_s \vert \mathbf{x}_t)$
-197.55	-4.17	-189.28	1.94

(c) $R_{\text{AVAE}}(\mathbf{x}_i, \mathbf{x}_t)$

$\log p(\mathbf{x}_t)$	$\log p(\mathbf{x}_i)$	$\log p(\mathbf{x}_t \vert \mathbf{x}_i)$	$\log p(\mathbf{x}_i \vert \mathbf{x}_t)$
-197.69	-743.57	-186.28	-730.44

(d) $R_{\text{MUSE}}(\mathbf{x}_s, \mathbf{x}_i, \mathbf{x}_t)$

$\log p(\mathbf{x}_t)$	$\log p(\mathbf{x}_s)$	$\log p(\mathbf{x}_i)$	$\log p(\mathbf{x}_t \vert \mathbf{x}_s)$	$\log p(\mathbf{x}_s \vert \mathbf{x}_t)$	$\log p(\mathbf{x}_t \vert \mathbf{x}_i)$	$\log p(\mathbf{x}_i \vert \mathbf{x}_t)$
-198.04	-4.53	-742.49	-198.10	1.96	-193.63	-735.02

Table 5.2: Accuracy (%) metrics for the generative models R in the extended “UJI Char Pen 2” dataset.

(a) $R_{\text{CVAE}}(\mathbf{x}_s, \mathbf{x}_t)$

Target	input
	$\mathbf{x}_s$
$\mathbf{x}_t$	81.52

(b) $R_{\text{AVAE}}(\mathbf{x}_s, \mathbf{x}_t)$

Target	input	
	$\mathbf{x}_t$	$\mathbf{x}_s$
$\mathbf{x}_t$	-	55.82
$\mathbf{x}_s$	62.96	-

(c) $R_{\text{AVAE}}(\mathbf{x}_i, \mathbf{x}_t)$

Target	input	
	$\mathbf{x}_t$	$\mathbf{x}_i$
$\mathbf{x}_t$	-	67.19
$\mathbf{x}_i$	64.03	-

(d) $R_{\text{MUSE}}(\mathbf{x}_s, \mathbf{x}_i, \mathbf{x}_t)$

Target	input					
	$\mathbf{x}_t$	$\mathbf{x}_s$	$\mathbf{x}_i$	$\mathbf{x}_t, \mathbf{x}_s$	$\mathbf{x}_t$, $\mathbf{x}_i$	$\mathbf{x}_s$, $\mathbf{x}_i$
$\mathbf{x}_t$	-	69.18	33.39	-	-	69.00
$\mathbf{x}_s$	52.09	-	34.94	60.81	-	-
$\mathbf{x}_i$	42.06	58.12	-	-	65.92	-

Table 5.3: Modality-distance metrics for the generative models R in the extended “UJI Char Pen 2” dataset.

(a) $R_{\text{CVAE}}(\mathbf{x}_s, \mathbf{x}_t)$

Target	input
	$\mathbf{x}_s$
$\mathbf{x}_t$	1.4395

(b) $R_{\text{AVAE}}(\mathbf{x}_s, \mathbf{x}_t)$

Target	input
	$\mathbf{x}_s$
$\mathbf{x}_t$	4.7258

(c) $R_{\text{AVAE}}(\mathbf{x}_i, \mathbf{x}_t)$

Target	input	
	$\mathbf{x}_t$	$\mathbf{x}_i$
$\mathbf{x}_t$	-	0.9678
$\mathbf{x}_i$	0.0059	-

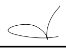
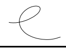
(d) $R_{\text{MUSE}}(\mathbf{x}_s, \mathbf{x}_i, \mathbf{x}_t)$

Target	input				
	$\mathbf{x}_t$	$\mathbf{x}_s$	$\mathbf{x}_i$	$\mathbf{x}_t, \mathbf{x}_s$	$\mathbf{x}_s, \mathbf{x}_i$
$\mathbf{x}_t$	-	4.6734	6.2185	-	2.8230
$\mathbf{x}_i$	0.0458	0.1475	-	0.0797	-

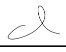
Furthermore, we also evaluate the potential of the models in encoding a representation robust to missing modality information by observing the trajectory samples generated from sound and image information. We show examples of such samples in Table 5.4. We see that the generated letter trajectories are quite different, depending on the representation model used. We can also see that the generated trajectories given the model $R_{\text{MUSE}}(\mathbf{x}_s, \mathbf{x}_i, \mathbf{x}_t)$ appear more wavy and unnatural that the other models, which could be related to the interpretation we proposed above.

Table 5.4: Implementation of PRG for the Robotic Dictaphone scenario, Represent (2nd) stage: generation of trajectory samples from all generative models R. For $R_{\text{CVAE}}(\mathbf{x}_s, \mathbf{x}_t)$ and $R_{\text{AVAE}}(\mathbf{x}_s, \mathbf{x}_t)$, the label information of a letter is given as input. For $R_{\text{AVAE}}(\mathbf{x}_i, \mathbf{x}_t)$ the image of a letter is given as input. For $R_{\text{MUSE}}(\mathbf{x}_s, \mathbf{x}_i, \mathbf{x}_t)$ we generate samples using the two options: giving the label information and, also, the image of a letter.

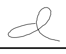
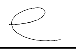
(a) $R_{\text{CVAE}}(\mathbf{x}_s, \mathbf{x}_t)$

Input	"P"	"d"	"e"
Trajectory			

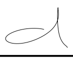
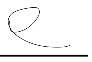
(b) $R_{\text{AVAE}}(\mathbf{x}_s, \mathbf{x}_t)$

Input	"P"	"d"	"e"
Trajectory			

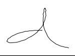
(c) $R_{\text{MUSE}}(\mathbf{x}_s, \mathbf{x}_i, \mathbf{x}_t)$

Input	"P"	"d"	"e"
Trajectory			

(d) $R_{\text{AVAE}}(\mathbf{x}_i, \mathbf{x}_t)$

Input			
Trajectory			

(e) $R_{\text{MUSE}}(\mathbf{x}_s, \mathbf{x}_i, \mathbf{x}_t)$

Input			
Trajectory			

5.2.2 Generate Stage

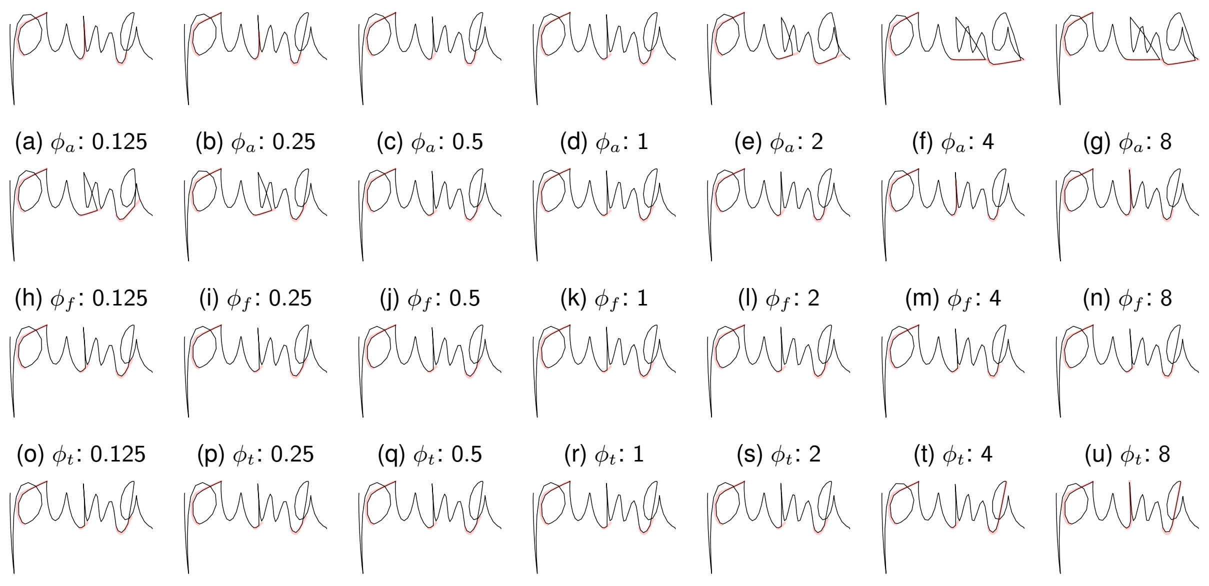
In this section we analyse Algorithm 5.1. More concretely, we study how the *penalization costs*, (ϕ_a, ϕ_f, ϕ_t, ϕ_o), individually interfere in the generation of the trajectories connecting consecutive letters. Since our goal is to write a word in one movement, we must consider where to finish each connection trajectory since its last point will always connect to the first point of the subsequent letter trajectory when converting it into a DMP. By such means, it is necessary to regulate all penalization costs to create smooth transitions between each letter and make it as natural as possible. Adjusting ϕ_a, we can see that as we increase its proportion concerning the other costs the connection trajectory tends to continue in a straight line connecting to the following letter. For ϕ_f, we see that as we decrease the influence of this cost compared to the others, the connection trajectory tends to ignore the initial point of the following letter trajectory and connect to a point in the middle of the trajectory instead. This conduct can lead to unnatural motions for specific letters, like "M" or "W". Throughout the range of values tested for ϕ_t, the changes are minimal. It seems that converting the final trajectory into a DMP overwrites the influence of this cost since the connection trajectory will always connect to the initial point of the subsequent trajectory. Finally, we can also see that as we increase the value of ϕ_o, the connection trajectories will tend to end where the subsequent trajectory starts. This way, higher values can turn the final trajectories more smooth since our object is to write words with only one motion. In our work, we attribute the following values for each penalization cost: $\phi_a = 1$, $\phi_f = 2$, $\phi_t = 1$ and $\phi_o = 8$.

5.2.3 PRG - Entire Pipeline

Finally, we qualitatively evaluate the efficacy of the entire full PRG framework in generating handwritten word samples $\mathbf{x}_t$ from image $\mathbf{x}_i$ or sound $\mathbf{x}_s$ information. We provide samples of the handwritten words generated by the different representation models in Table 5.5. Once again, our framework can generate high-quality and coherent handwritten word samples regardless of the representation model employed and the modality available to the framework. In particular, we observe that the R_{CVAE} and R_{AVAE} models allow for the generation of more high-quality word samples yet are unable to scale to more than two modalities. On the other hand, R_{MUSE} still allows for the generation of coherent word samples (despite a loss in quality in comparison with the other models), able to learn a multimodal representation robust to missing modality information, scalable to settings with a larger number of modalities.

5.2.4 Baxter Simulation

Lastly, we show the reliability of our framework in a simulated environment. We perform an authentic interaction where the actuation portion of the robotic agent receives the output of our framework and originates and executes a physical movement.



(a) ϕ_a: 0.125 (b) ϕ_a: 0.25 (c) ϕ_a: 0.5 (d) ϕ_a: 1 (e) ϕ_a: 2 (f) ϕ_a: 4 (g) ϕ_a: 8

(h) ϕ_f: 0.125 (i) ϕ_f: 0.25 (j) ϕ_f: 0.5 (k) ϕ_f: 1 (l) ϕ_f: 2 (m) ϕ_f: 4 (n) ϕ_f: 8

(o) ϕ_t: 0.125 (p) ϕ_t: 0.25 (q) ϕ_t: 0.5 (r) ϕ_t: 1 (s) ϕ_t: 2 (t) ϕ_t: 4 (u) ϕ_t: 8

(v) ϕ_o: 0.125 (w) ϕ_o: 0.25 (x) ϕ_o: 0.5 (y) ϕ_o: 1 (z) ϕ_o: 2 (aa) ϕ_o: 4 (ab) ϕ_o: 8

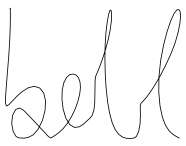
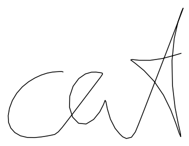
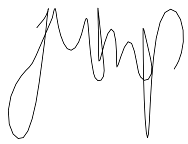
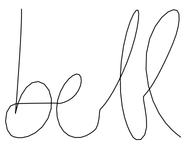
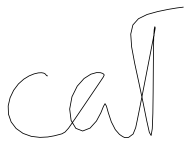
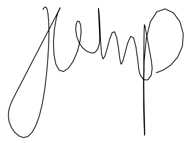
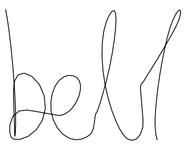
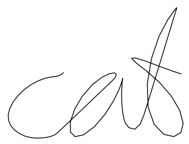
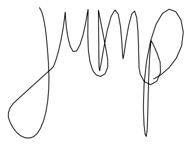
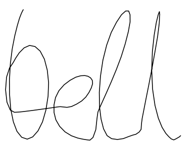
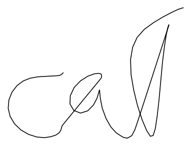
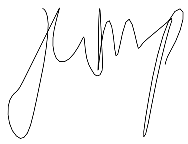
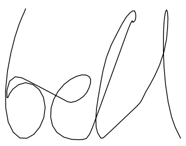
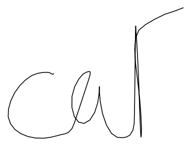
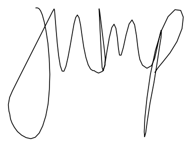
Figure 5.5: Implementation of PRG for the Robotic Dictaphone scenario, Generate (3rd) stage: variations of Algorithm 5.1 where the parts of the trajectories highlighted in red are the *connection* trajectories (between letter trajectories) that are being generated by the algorithm. For every sample, all cost parameters (ϕ_a, ϕ_f, ϕ_t, ϕ_o) are set to 1 except one cost parameter, where its value is defined by the value present in the sub-caption of the corresponding sub-figure. The word being written is ”puma”.

We first define the simulation environment. The simulation environment that we use is the Open Robotics Automation Virtual Environment (OpenRAVE) [48], which is an environment that allows for the deployment of kinematic planning algorithms in real-world autonomous robot applications. All simulations ran on a personal computer using a virtual environment running Ubuntu 14.04.5. The CPU is the AMD Ryzen 7 5800HS with 8 cores and 16 threads with a clock of 3.2 GHz. The computer has 16 GB of installed RAM and an SSD with 512 GB of storage. The graphics card is an NVIDIA GeForce RTX 3060 Laptop GPU.

The robotic agent used is the Baxter robot. Baxter has two independent force-sensing arms, each with 7 DOFs. Each wrist has a camera, accelerometer, and distance IR laser sensor. On top of the head display, there is a camera and $360°$-degree sonar. Baxter uses the Robot Operating System (ROS) framework to communicate between high-level programming and the hardware.

For the simulation setup, we put the Baxter robot in front of a table. We also substitute the right-hand grip for a support with a pen. Baxter will use the pen to write directly on the table. From an initial input modality, like the sound of a word, we pass that data through our framework that generates a possible and plausible single-stroke handwriting motion for the given word. Since the outputted motion is a 2D trajectory, we perform several transformations, making it compatible with the Baxter actuation system. First, we convert the movement into a spatial trajectory parallel with the z-axis. Then, we use

Table 5.5: Trajectory samples retrieved from running our full pipeline PRG, implemented for the Robotic Dictaphone scenario, when given as input the sound of the respective word, $\mathbf{x}_s$, or the images of the letters, $\mathbf{x}_i$. For the Represent and Generate stages we tested all implemented generative models: $R_{\text{CVAE}}(\mathbf{x}_s, \mathbf{x}_t)$, $R_{\text{AVAE}}(\mathbf{x}_s, \mathbf{x}_t)$, $R_{\text{AVAE}}(\mathbf{x}_i, \mathbf{x}_t)$, and $R_{\text{MUSE}}(\mathbf{x}_s, \mathbf{x}_i, \mathbf{x}_t)$.

Model	input	word		
		bell	cat	jump
$R_{\text{CVAE}}(\mathbf{x}_s, \mathbf{x}_t)$	$\mathbf{x}_s$			
$R_{\text{AVAE}}(\mathbf{x}_s, \mathbf{x}_t)$	$\mathbf{x}_s$			
$R_{\text{AVAE}}(\mathbf{x}_i, \mathbf{x}_t)$	$\mathbf{x}_i$			
$R_{\text{MUSE}}(\mathbf{x}_s, \mathbf{x}_i, \mathbf{x}_t)$	$\mathbf{x}_s$			
$R_{\text{MUSE}}(\mathbf{x}_s, \mathbf{x}_i, \mathbf{x}_t)$	$\mathbf{x}_i$			

Baxter's default inverse kinematics procedure to transform the spatial trajectory into the joint angles of the robotic arm, Baxter's right arm in this case, required to execute the given motion. We then pass the new description to the robot to run. See Figure 5.6 for an example.

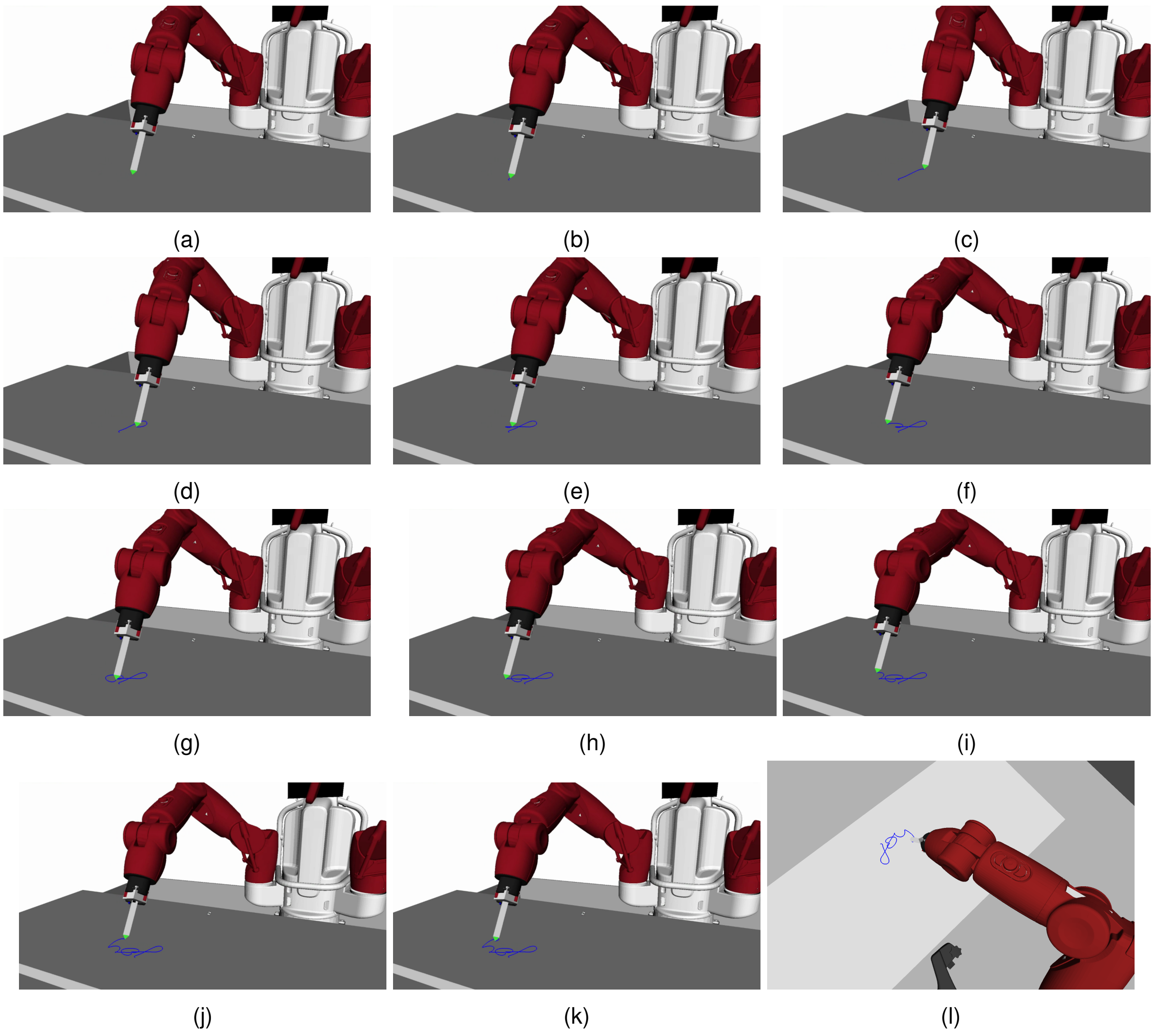


Figure 5.6: A 7-DOFs Baxter manipulator writing the word "joy" in a simulation environment on Open-RAVE. Figures 5.6a to 5.6k show incremental steps of Baxter writing the given word. Figure 5.6l shows the written word in another angle.

6 Conclusions and Future Work

Contents

In this chapter, we summarize the contributions of this thesis and discuss possible extensions of this work for future work.

6.1 Contributions

In this work, we addressed the problem of translating multimodal commands provided by different communication channels given by the human user to a single DMP movement executed by a robotic agent. We contributed with a novel three-stage pipeline that allows the processing, mapping, and generation of trajectory information, regardless of the communication channels employed by the human user to provide information to the agent. Our pipeline is robust to missing modalities. The execution of our pipeline starts by receiving any combination of input modalities. The first stage of our pipeline processes the received input modalities with their respective mapping function to extract valuable information and decompose, if necessary, the received data into a sequence of smaller representations that are more processable. Next, the representation stage receives the new representation sequence of the input modalities iteratively. It uses a generative model to encode the given representations into a latent representation that captures and correlates information from all given modalities. In the third stage, the same model decodes this latent vector to generate coherent, possibly new, modality data, in our case, the motion modality. Next, the sequence of motions is processed, performing additional transformations relevant to the task/problem to solve, and are then converted into a single DMP, ready to be executed by the robotic agent.

Our framework presents a new method to integrate multimodal sensory information into a robotic setting. One concern that immanently arises from it is how to perform cross-modal generation or cross-modal completion, focusing on manipulating all modalities and extracting relevant information necessary to facilitate the cross-generation and solve the corresponding task. By integrating a multimodal VAE in our pipeline, we present another possible way to correlate and combine different modalities for these types of HRI systems. The MVAE model can effectively associate the given modalities to a latent representation that can then be used to generate any modality, even if it was initially missing. The model is robust enough to perform this cross-generation even if the given input contains only a subset of all possible modalities, making it robust to missing modalities.

We also instantiate our pipeline in the context of a Robotic Dictaphone: the generation of robotic handwriting from textual information provided through the speech of the human user or visual information from the images of the characters to write. Our results show that our approach allows the generation of accurate handwritten samples that highly resemble human cursive writing, regardless of the number and nature of the communication channels employed by the human user or generative model used, working even under uncertainty where some modalities are missing. Our pipeline is agnostic both to the nature of the task and the communication channels employed by the human user.

Finally, we confirm the usability of our framework in a simulation environment where we first use our framework to generate the handwriting motion given the speech relative to the word to write, as explained above, and then use the Baxter robot to execute the given motion. Thereby, we show a practical implementation of our framework for the Robotic Dictaphone scenario.

6.2 Future Work

One possible and straightforward direction to continue this thesis is the application of the proposed framework to other HRI situations. It would be interesting to implement the developed framework in scenarios with different characteristics, restrictions, goals, and evaluation procedures. Thus, making it possible to evaluate our framework in more depth and understand its role and usefulness. We could also test if the usefulness of our framework directly depends on the category of the task. For example, we can examine how well our implementation operates in tasks that have different objectives to optimize: like a task that performs a systematic work; tasks where we need smooth coordination between the robot and human; or when we need an implementation that works even with high-levels of uncertainty (multiple modalities missing). Another essential aspect worth testing is how well the framework works and scales in other scenarios, especially when various modalities are involved. In Section 1.3.2, we gave two possible scenarios as examples: a grasping task and indoor navigation of a robot.

Another line of research worth exploring is to extend our framework to have memory. In this new setting, the generated motions from the representation stage would be conditioned on the given command, from the input modalities, and, also, on the previously generated motion. With this improved architecture, we would implicitly create a level of abstraction which reduces the discrepancy of how humans think and how the framework operates. To accomplish this, we would have to try several approaches combining RNNs with multimodal VAEs.

Focusing on the robotic dictaphone scenario, we argue that it would be compelling to apply the framework on the real world using a real Baxter robot, proving its practicality even more. Another critical aspect of testing would be creating a user test to ask random users to distinguish between word samples generated by our framework and actual human samples. The proposed user test would be another way of evaluating our framework quantitatively.

From the discussion above, upgrading the generative model architecture to merge multimodal VAEs with RNNs would implicate that the framework would work on word level instead of character level. Such improvement brings benefits. One of them is that there is no need to process the motion trajectories after the generation process since the model learns by itself how to accommodate the motion of each letter to form the corresponding word. We could also try use the sound information directly as another modality to the generative model without pre-processing it beforehand.

Bibliography

[1] J. Ruiz-del-Solar, P. Loncomilla, and N. Soto, “A survey on deep learning methods for robot vision,” *CoRR*, vol. abs/1803.10862, 2018. [Online]. Available: http://arxiv.org/abs/1803.10862

[2] Y. Lu, Z. Xue, G.-S. Xia, and L. Zhang, “A survey on vision-based uav navigation,” *Geo-spatial Information Science*, vol. 21, no. 1, pp. 21–32, 2018. [Online]. Available: https://doi.org/10.1080/10095020.2017.1420509

[3] M. Malik, M. K. Malik, K. Mehmood, and I. Makhdoom, “Automatic speech recognition: a survey,” *Multimedia Tools and Applications*, vol. 80, no. 6, pp. 9411–9457, Mar 2021. [Online]. Available: https://doi.org/10.1007/s11042-020-10073-7

[4] O. Kroemer, S. Niekum, and G. Konidaris, “A review of robot learning for manipulation: Challenges, representations, and algorithms,” *Journal of Machine Learning Research*, vol. 22, no. 30, pp. 1–82, 2021. [Online]. Available: http://jmlr.org/papers/v22/19-804.html

[5] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, “Learning dexterous in-hand manipulation,” *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[6] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixao, F. Mutz *et al.*, “Self-driving cars: A survey,” *Expert Systems with Applications*, vol. 165, p. 113816, 2021.

[7] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, “A survey of autonomous driving: Common practices and emerging technologies,” *IEEE access*, vol. 8, pp. 58 443–58 469, 2020.

[8] V. Villani, F. Pini, F. Leali, and C. Secchi, “Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications,” *Mechatronics*, vol. 55, pp. 248–266, 2018.

[9] B. Scassellati, H. Admoni, and M. Matarić, “Robots for use in autism research,” *Annual review of biomedical engineering*, vol. 14, pp. 275–294, 2012.

[10] F. S. Melo, A. Sardinha, D. Belo, M. Couto, M. Faria, A. Farias, H. Gamboa, C. Jesus, M. Kinarullathil, P. Lima *et al.*, "Project inside: towards autonomous semi-unstructured human–robot social interaction in autism therapy," *Artificial intelligence in medicine*, vol. 96, pp. 198–216, 2019.

[11] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, "Continuum robots for medical applications: A survey," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1261–1280, 2015.

[12] S. Chandra, R. Paradeda, H. Yin, P. Dillenbourg, R. Prada, and A. Paiva, "Do children perceive whether a robotic peer is learning or not?" in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, 2018, pp. 41–49.

[13] I. Leite, C. Martinho, and A. Paiva, "Social robots for long-term interaction: a survey," *International Journal of Social Robotics*, vol. 5, no. 2, pp. 291–308, 2013.

[14] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy, "Approaching the symbol grounding problem with probabilistic graphical models," *AI magazine*, vol. 32, no. 4, pp. 64–76, 2011.

[15] J. E. Laird, K. Gluck, J. Anderson, K. D. Forbus, O. C. Jenkins, C. Lebiere, D. Salvucci, M. Scheutz, A. Thomaz, G. Trafton *et al.*, "Interactive task learning," *IEEE Intelligent Systems*, vol. 32, no. 4, pp. 6–21, 2017.

[16] H. Ahmadi, M. S. Hoseinzadeh, A. Ekhlasi, and M. Latifi, "Voice commands classification in order to control robot movement," Feb 2021. [Online]. Available: https://figshare.com/articles/conference_contribution/Voice_commands_classification_in_order_to_control_robot_movement/13712842/2

[17] S. Lobov, V. Mironov, I. Kastalskiy, and V. Kazantsev, "Combined use of command-proportional control of external robotic devices based on electromyography signals." *Medical Technologies in Medicine/Sovremennye Tehnologii v Medicine*, vol. 7, no. 4, 2015.

[18] K. Gundogdu, S. Bayrakdar, and I. Yucedag, "Developing and modeling of voice control system for prosthetic robot arm in medical systems," *Journal of King Saud University - Computer and Information Sciences*, vol. 30, no. 2, pp. 198–205, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157817300216

[19] R. Huang and G. Shi, "Design of the control system for hybrid driving two-arm robot based on voice recognition," in *IEEE 10th International Conference on Industrial Informatics*, 2012, pp. 602–605.

[20] M. A. V. J. Muthugala and A. G. B. P. Jayasekara, "Enhancing human-robot interaction by interpreting uncertain information in navigational commands based on experience and environment," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 2915–2921.

[21] S. Chandra, P. Dillenbourg, and A. Paiva, "Children teach handwriting to a social robot with different learning competencies," *International Journal of Social Robotics*, vol. 12, no. 3, pp. 721–748, Jul 2020. [Online]. Available: https://doi.org/10.1007/s12369-019-00589-w

[22] K. Endo, M. Kanoh, and T. Nakamura, "Teaching handwriting using robot and onomatopoeia," in *2015 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, 2015, pp. 484–490.

[23] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2014.

[24] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper/2014/file/d523773c6b194f37b938d340d5d02232-Paper.pdf

[25] M. Vasco, H. Yin, F. S. Melo, and A. Paiva, "How to sense the world: Leveraging hierarchy in multimodal perception for robust reinforcement learning agents," 2021.

[26] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 02 2013. [Online]. Available: https://doi.org/10.1162/NECO_a_00393

[27] M. Ginesi, N. Sansonetto, and P. Fiorini, "Overcoming some drawbacks of dynamic movement primitives," *Robotics and Autonomous Systems*, vol. 144, p. 103844, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889021001299

[28] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 02 2013. [Online]. Available: https://doi.org/10.1162/NECO_a_00393

[29] ——, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[30] H. Yin, F. Melo, A. Billard, and A. Paiva, "Associate latent encodings in learning from demonstrations," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, pp. 3848–3854, Feb. 2017.

[31] M. Suzuki, K. Nakayama, and Y. Matsuo, "Joint multimodal learning with deep generative models," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=BkL7bONFe

[32] M. Wu and N. Goodman, “Multimodal generative models for scalable weakly-supervised learning,” in *Advances in Neural Information Processing Systems*, 2018, pp. 5575–5585.

[33] Y. Shi, S. N, B. Paige, and P. Torr, “Variational mixture-of-experts autoencoders for multi-modal deep generative models,” in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019, pp. 15 692–15 703. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/0ae775a8cb3b499ad1fca944e6f5c836-Paper.pdf

[34] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, “Learning and generalization of motor skills by learning from demonstration,” in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 763–768.

[35] B. Nemec and A. Ude, “Action sequencing using dynamic movement primitives,” *Robotica*, vol. 30, no. 5, p. 837–846, 2012.

[36] T. Kulvicius, K. Ning, M. Tamosiunaite, and F. Worgötter, “Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting,” *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 145–157, 2012.

[37] D. Yongda, L. Fang, and X. Huang, “Research on multimodal human-robot interaction based on speech and gesture,” *Computers & Electrical Engineering*, vol. 72, pp. 443–454, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790618315222

[38] I. Rodomagoulakis, N. Kardaris, V. Pitsikalis, E. Mavroudi, A. Katsamanis, A. Tsiami, and P. Maragos, “Multimodal human action recognition in assistive human-robot interaction,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 2702–2706.

[39] H. Liu, T. Fang, T. Zhou, and L. Wang, “Towards robust human-robot collaborative manufacturing: Multimodal fusion,” *IEEE Access*, vol. 6, pp. 74 762–74 771, 2018.

[40] H. Wang, X. Li, and X. Zhang, “Multimodal human-robot interaction on service robot,” in *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, vol. 5, 2021, pp. 2290–2295.

[41] K. Noda, H. Arie, Y. Suga, and T. Ogata, “Multimodal integration learning of robot behavior using deep neural networks,” *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 721–736, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889014000396

[42] D. Rao, M. D. Deuge, N. Nourani–Vatani, S. B. Williams, and O. Pizarro, “Multimodal learning and inference from visual and remotely sensed data,” *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 24–43, 2017. [Online]. Available: https://doi.org/10.1177/0278364916679892

[43] Q. Luo, J. Wu, and M. C. Gombolay, “A generalized robotic handwriting learning system based on dynamic movement primitives (dmps),” *CoRR*, vol. abs/2012.03898, 2020. [Online]. Available: https://arxiv.org/abs/2012.03898

[44] H. Yin, P. Alves-Oliveira, F. S. Melo, A. Billard, and A. Paiva, “Synthesizing robotic handwriting motion by learning from human demonstrations,” in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, ser. IJCAI’16, 2016, p. 3530–3537.

[45] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 449–12 460, 2020.

[46] D. Llorens, F. Prat, A. Marzal, J. M. Vilar, M. J. Castro, J. C. Amengual, S. Barrachina, A. Castellanos, S. España, J. A. Gómez, J. Gorbe, A. Gordo, V. Palazón, G. Peris, R. Ramos-Garijo, and F. Zamora, “The ujipenchars database: a pen-based database of isolated handwritten characters,” in *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC’08)*, N. C. C. Chair), K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, and D. Tapias, Eds. Marrakech, Morocco: European Language Resources Association (ELRA), may 2008, pp. 2647–2651, http://www.lrec-conf.org/proceedings/lrec2008/.

[47] H. Yin, P. A. Oliveira, F. S. Melo, A. Billard, and A. Paiva, “Synthesizing robotic handwriting motion by learning from human demonstrations,” in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, S. Kambhampati, Ed., IJCAI. New York: AAAI Press, International Joint Conferences on Artificial Intelligence, 2016, pp. 3530–3537. [Online]. Available: https://www.ijcai.org/proceedings/2016

[48] R. Diankov, “Automated construction of robotic manipulation programs,” Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, August 2010. [Online]. Available: http://www.programmingvision.com/rosen_diankov_thesis.pdf

A

Generative Models' Architectures

We present the architectures used for all generative models: $R_{\text{CVAE}}(\mathbf{x}_s, \mathbf{x}_t)$ (Table A.1), $R_{\text{AVAE}}(\mathbf{x}_s, \mathbf{x}_t)$ (Table A.2), $R_{\text{AVAE}}(\mathbf{x}_i, \mathbf{x}_t)$ (Table A.3) and $R_{\text{MUSE}}(\mathbf{x}_s, \mathbf{x}_i, \mathbf{x}_t)$ (Tables A.4 and A.5).

Table A.1: Architecture of $R_{\text{CVAE}}(\mathbf{x}_s, \mathbf{x}_t)$ and additional parameters.

Encoder	$(\mathbf{x}_m, \mathbf{x}_s) \in \mathbb{R}^{200+62}$
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 16; FC 16
Decoder	$(\mathbf{z}, \mathbf{x}_s) \in \mathbb{R}^{16+62}$
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 200
Latent dims	$16 + 62$
Batch size	128
Optimizer	Adam, $lr = 1 \exp^{-4}$

Table A.2: Architecture of $R_{\text{AVAE}}(\mathbf{x}_s, \mathbf{x}_t)$ and additional parameters.

Encoder ($\mathbf{x}_s$)	$\mathbf{x}_s \in \mathbb{R}^{62}$
	Word Embeding 256
	FC 256, LeakyReLU
	FC 256, LeakyReLU
	FC 16; FC 16
Decoder ($\mathbf{x}_s$)	$\mathbf{z}_s \in \mathbb{R}^{16}$
	FC 256, LeakyReLU
	FC 256, LeakyReLU
	FC 256, LeakyReLU
	FC 62
Encoder ($\mathbf{x}_m$)	$\mathbf{x}_m \in \mathbb{R}^{200}$
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 16; FC 16
Decoder ($\mathbf{x}_m$)	$\mathbf{z}_m \in \mathbb{R}^{16}$
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 200
Latent dims	$\mathbf{z}_s = 16$, $\mathbf{z}_m = 16$
Batch size	128
Optimizer	Adam, $lr = 1\exp^{-4}$
α (from (2.7))	1

Table A.3: Architecture of $R_{\text{AVAE}}(\mathbf{x}_i, \mathbf{x}_t)$ and additional parameters.

Encoder ($\mathbf{x}_i$)	$\mathbf{x}_i \in \mathbb{R}^{1\times 28\times 28}$
	4×4 Conv 64, stride 2, pad 1, LeakyReLU
	4×4 Conv 128, stride 2, pad 1, LeakyReLU
	4×4 Conv 256, stride 2, pad 1, LeakyReLU
	FC 512, LeakyReLU
	FC 16; FC 16
Decoder ($\mathbf{x}_i$)	$\mathbf{z}_i \in \mathbb{R}^{16}$
	FC 512, LeakyReLU
	FC 2304, LeakyReLU
	4×4 ConvTranspose 128, stride 2, pad 1, out padding 1, LeakyReLU
	4×4 ConvTranspose 64, stride 2, pad 1, LeakyReLU
	4×4 ConvTranspose 1, stride 2, pad 1, Sigmoid
Encoder ($\mathbf{x}_m$)	$\mathbf{x}_m \in \mathbb{R}^{200}$
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 16; FC 16
Decoder ($\mathbf{x}_m$)	$\mathbf{z}_m \in \mathbb{R}^{16}$
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 200
Latent dims	$\mathbf{z}_i = 16$, $\mathbf{z}_m = 16$
Batch size	128
Optimizer	Adam, $lr = 1\exp^{-4}$
α (from (2.7))	1

Table A.4: Architecture of $R_{\text{MUSE}}(\mathbf{x}_s, \mathbf{x}_i, \mathbf{x}_t)$ low level and additional parameters. We assume that $\mathbf{x}_s = \mathbf{x}_1$, $\mathbf{x}_i = \mathbf{x}_2$ and $\mathbf{x}_m = \mathbf{x}_3$.

Encoder ($\mathbf{x}_s$)	$\mathbf{x}_s \in \mathbb{R}^{62}$
	Word Embeding 256
	FC 256, LeakyReLU
	FC 256, LeakyReLU
	FC 8; FC 8
Decoder ($\mathbf{x}_s$)	$\mathbf{z}_s \in \mathbb{R}^{8}$
	FC 256, LeakyReLU
	FC 256, LeakyReLU
	FC 256, LeakyReLU
	FC 62
Encoder ($\mathbf{x}_i$)	$\mathbf{x}_i \in \mathbb{R}^{1\times 28\times 28}$
	4×4 Conv 64, stride 2, pad 1, LeakyReLU
	4×4 Conv 128, stride 2, pad 1, LeakyReLU
	4×4 Conv 256, stride 2, pad 1, LeakyReLU
	FC 512, LeakyReLU
	FC 8; FC 8
Decoder ($\mathbf{x}_i$)	$\mathbf{z}_i \in \mathbb{R}^{8}$
	FC 512, LeakyReLU
	FC 2304, LeakyReLU
	4×4 ConvTranspose 128, stride 2, pad 1, out padding 1, LeakyReLU
	4×4 ConvTranspose 64, stride 2, pad 1, LeakyReLU
	4×4 ConvTranspose 1, stride 2, pad 1, Sigmoid
Encoder ($\mathbf{x}_m$)	$\mathbf{x}_m \in \mathbb{R}^{200}$
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 8; FC 8
Decoder ($\mathbf{x}_m$)	$\mathbf{z}_m \in \mathbb{R}^{8}$
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 200
Latent dims	$\mathbf{z}_s = 8$, $\mathbf{z}_i = 8$, $\mathbf{x}_m = 8$
Batch size	64
Optimizer	Adam, $lr = 1\exp^{-4}$
α_1 (from (2.8))	1
α_2 (from (2.8))	1
α_3 (from (2.8))	1
λ_1 (from (2.8))	1
λ_2 (from (2.8))	1
λ_3 (from (2.8))	50
δ (from (2.11)	1

Table A.5: Architecture of $R_{\text{MUSE}}(\mathbf{x}_s, \mathbf{x}_i, \mathbf{x}_t)$ high level and additional parameters.

Encoder (top level, equal for all modalities)	$\mathbf{c} \in \mathbb{R}^8$
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 8; FC 8
Decoder (top level, equal for all modalities)	$\mathbf{z}_\pi \in \mathbb{R}^8$
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 512, LeakyReLU
	FC 8
Latent dims	$\mathbf{z}_\pi = 8$
Optimizer	Adam, $lr = 1\exp^{-4}$
γ_1 (from (2.9))	10
γ_2 (from (2.9))	10
γ_3 (from (2.9))	10
β (from (2.9)	1