

Effective Non-Verbal Communication for Mobile Robots using Expressive Lights



Kim Baraka

CMU-RI-TR-16-12

Submitted in partial fulfillment of the requirements for the degree of
Master of Science in Robotics

Thesis committee:

Manuela Veloso, *chair*

Illah Nourbakhsh

Stephanie Rosenthal

Heather Knight

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

May 2016

Acknowledgements

First, I would like to thank Joydeep Biswas, Brian Coltin, Stephanie Rosenthal, and everyone else who contributed to making CoBot what it is today, and without whom this thesis would have certainly not been conceivable.

I would like to acknowledge Ana Paiva and the rest of the GAIPS group for their guidance in my user studies.

I also thank Richard Wang for sharing his knowledge, tips and tricks when working with CoBot.

I am grateful to the members of the CORAL group for their input on my research as well as the presentation of my work.

Thank you Walid Tamari for inspiring me to use programmable light strips on CoBot.

Finally, I express my sincere gratitude to my advisor Prof. Manuela Veloso, and the rest of my thesis committee, Prof. Illah Nourbakhsh, Dr. Stephanie Rosenthal and Heather Knight for their continuous feedback and their helpful insight throughout my research.

This research was partially supported by the FCT INSIDE ERI grant, FLT grant number 2015-143894, NSF grant number IIS-1012733, and ONR grant N00014-09-1-1031. The views and conclusions contained in this document are those of the author only.

Abstract

Mobile robots are entering our daily lives and are expected to carry out tasks with, for, and around humans in diverse environments. Due to their mobility and the diversity of their states while executing their tasks, revealing robot state information during task execution is crucial to enable effective human-robot collaboration, better trust in the robot, and more engaging human-robot social interactions. Verbal communication combined with on-screen display is currently the typical mechanism for communicating with humans on such robots. However, these communication mechanisms may fail for mobile robots due to spatio-temporal limitations. To remedy these problems, in this thesis, we use expressive lights as a primary modality to communicate to humans useful information about the robot’s state. Such lights are persistent, non-invasive, and visible at a distance, unlike other existing modalities, which they can complement or replace when impotent. Current light arrays provide us with a very large animation space, which we simplify by considering a handful of parametrized signal shapes that maintain great animation design flexibility. We present a formalism for light animation control and a mapping architecture from our representation of robot state to our parametrized light animation space. The mapping we propose generalizes to multiple light strips and even other expression modalities. We also show how this mapping can adapt, through a personalization algorithm, to temporal preferences of individuals engaging in long-term interactions with the robot. We implement our framework on CoBot, a mobile multi-floor service robot, and evaluate its validity through several user studies. Our study results show that carefully designed expressive lights on a mobile robot help humans better understand robot states and actions and can have a positive impact on people’s behavior in the real world.

Contents

List of Figures	xi
------------------------	-----------

List of Tables	xiii
-----------------------	-------------

1 Introduction	1
1.1 Motivation	1
1.2 Approach	2
1.3 Contributions	3
1.4 Reader's guide to the thesis	4
2 Related Work	5
2.1 Short survey of uses of lights	5
2.1.1 Lights for communication at a distance	5
2.1.2 Lights for revealing state information	6
2.1.3 Lights and aesthetics	6
2.2 Light as an expressive medium	6
2.2.1 Light control	6
2.2.2 Light expression space	7
2.2.3 Light animation semantics	7
2.3 Robot expression	8
2.3.1 Lights on robots	8
2.3.2 Other non-verbal modalities for robot expression	8
2.4 Personalization in HRI	8
3 Animating light sources	11
3.1 Light animation and animation space definitions	11
3.1.1 Light animation as a continuous intensity function matrix	11
3.1.2 Spatial layout	11
3.1.3 Animation space intensity functions	12
3.1.4 Animation tuple representation	13
3.2 Signal shape parametrization	14
3.2.1 Rectangle waveform	14
3.2.2 Triangle waveform	14
3.2.3 Sinusoidal waveform	15

3.2.4	Modulated waveform	15
3.2.5	Step function	16
3.2.6	Clipped ramp function	16
3.3	Animating a digital RGB LED strip	16
3.3.1	Light animation as a sequence of frames	17
3.3.2	Episodic animation control	17
3.4	Chapter summary	20
4	Mobile service robot state and its expressible elements	21
4.1	CoBot overview	21
4.1.1	CoBot tasks and services	21
4.1.2	CoBot user modalities	23
4.1.3	Robot motion modes	24
4.2	Robot state representation	24
4.2.1	Robot variables	24
4.2.2	State features and robot state	25
4.3	<i>What</i> part of robot state to express?	27
4.3.1	Expressible state tuples	28
4.3.2	Clustering expressible state tuples: expressible classes	28
4.4	Robot state / animation mapping	29
4.4.1	Mapping architecture	29
4.4.2	Expression of non-exclusive state features	30
4.5	Extension to multiple light strips / expression channels	32
4.6	Implementation of the mapping on a real robot	33
4.6.1	Hardware components	33
4.6.2	Control architecture	34
4.7	Chapter summary	35
5	Design and evaluation of the state/animation mapping	37
5.1	User study 1: Designing appropriate animations	37
5.1.1	Methodology	37
5.1.2	Preliminary Study	37
5.1.3	Participants	38
5.1.4	Survey design	38
5.1.5	Results and discussion	39
5.2	User study 2: Evaluating and generalizing the designed animations	40
5.2.1	Participants	42
5.2.2	Survey Design	43
5.2.3	Scenario descriptions	43
5.2.4	Multiple choice questions	44
5.2.5	Results	45
5.2.6	Discussion	47
5.3	Experiment: Impact on human behavior	48
5.3.1	Experimental procedure	48

5.3.2	Data filtering	48
5.3.3	Results and discussion	49
5.4	Chapter summary	49
6	Introducing agency in expression: personalization and adaptation in persistent interactions	51
6.1	Background	52
6.2	Formalism and user modeling	53
6.2.1	Problem setting	53
6.2.2	Modeling dynamic user preferences over time	53
6.3	Learning model parameters from user feedback	56
6.3.1	Profile “conservative”	56
6.3.2	Profile “consistent but fatigable”	57
6.3.3	Profile “erratic”	58
6.3.4	Action sequences generation	58
6.4	Results	58
6.5	Chapter summary and discussion	59
7	Conclusion	61
7.1	Summary and discussion of contributions	61
7.2	Future Work	62
	Bibliography	63
	Appendix A Protocol for Arduino “serial in” communication	67

List of Figures

1.1	Overview of the approach taken in this thesis (contributions are shown in red)	3
3.1	Examples of different spatial layouts for a set of lights	12
3.2	Episodic animation example: animation tuple as a function of time and corresponding intensity functions for each pixel of the strip	18
3.3	Summary of the animation framework described in this chapter	20
4.1	An example of symbiotic autonomy: CoBot getting help from a human at an elevator	22
4.2	CoBot performing different services: (A) an escort service; (B) a transport service	23
4.3	Turn detection check used for computation of feature ‘Turning’	27
4.4	Summary of the introduced robot state concepts	30
4.5	Architecture of the robot state / animation mapping	31
4.6	Flow diagram of the robot state / animation mapping	31
4.7	Mapping architecture for multiple expression channels	32
4.8	Two expressive light channels for multi-level expression on CoBot (left) and turn indicator snapshots for the low-level strip	33
4.9	Hardware interface design	34
4.10	Control diagram of the state animation interface	35
5.1	Animation pattern results	40
5.2	Snapshots of the winning animations for each scenario of user study 1, also used in the subsequent user studies	41
5.3	Comparison of the accuracies on each scenario across the two study conditions (error bars constructed using a 95% confidence interval; statistical significance shown used t-test)	46
6.1	Sample preferred of animation sequences for the user models presented	56
6.2	Simulation results showing sequences of actions taken by the agent and the corresponding reward sequences from a simulated user belonging to: (a) model “conservative”, (b) model “consistent but fatigable” and (c) model “erratic”	59

List of Tables

3.1	List of parametrized signal shapes considered for each $i_{jc_k}(t)$	13
5.1	Parameter values for the animation choices shown	39
5.2	Color results (color codes correspond to those used in the text)	42
5.3	Selected best animations for each scenario of user study 1	42
5.4	Scenarios used in user study 2	45
5.5	Proportion of people who helped the robot in the two experimental conditions across the three floors	49

Chapter 1

Introduction

1.1 Motivation

Mobile robots are entering our daily lives and are expected to carry out tasks with, for, and around humans in environments such as hospitals, supermarkets, hotels, offices, and shops. For effective operation of these robots in these human-populated environments, it is important that humans have an understanding of some of the processes, information, or decisions taking place on the robot. Due to their mobility and possible diversity of both their states and actions while evolving in a dynamic environment, *revealing information about the robot's state* over the course of its task execution is crucial to enable: (1) effective collaboration between humans and the robot, (2) better trust in the robot, and (3) more engaging human-robot social interactions. On the other hand, these robots are expected to be persistent, encountering the same users repeatedly, which urges us to consider the dynamics of long-term human-robot interactions as part of the design process of these robots.

Speech communication combined with on-screen display is the typical communication mechanism for communicating with humans on such robots. However, when it comes to mobile robots traveling long distances, these communication mechanisms may fail for different reasons. First, humans are not always in close proximity to the robot in which case the speech might be inaudible and the on-screen text not visible. Second, speech is transient in that its associated expressions only last the duration of a sentence. The *mobile* aspect of these robots, also shared by other embodied agents such as self-driving cars or aerial robots, hence calls for other methods of communication which are both persistent and perceivable at a distance.

To remedy these problems, in this thesis, we propose to use *expressive lights* primarily as a way to communicate useful information about the robot's state. Such lights are *visible at a distance*, provide a *persistent* visualization of state information, and are *non-invasive* as compared to other possible modalities such as loud non-verbal expressive sounds. By using such lights, we are effectively enabling the robot to *modify its appearance* as a method of communication with humans, which is a distinguishing feature for expression as compared to other modalities. Literature has shown that abstract dynamic visual cues [36], and more specifically dynamic lighting [42], have been shown to elicit interactive social responses.

These results potentially suggest that *expressive lights* on a robot are also likely to create more engaging interactions with humans, if they are communicating useful information, i.e., they are *informative*, and if they do so in a *legible*, i.e., readable, manner. These persistent lights might also serve as a complement to existing modalities of interaction which are transient (e.g., speech) or that require close proximity (e.g., on-screen text).

As a result of our literature review, two additional aspects of robot communication (regardless of the modality used) have been found to be lacking. First, expressive behaviors are theoretically dissociated from robot states; in other words, there lacks a framework connecting robot expressions to robot states that is easily portable to different platforms. Second, current robot communication has been found to be generally very rigid. Most robot expressions are scripted and do not allow for much flexibility, adaptation and personalization. This thesis will also attempt to address these two issues.

While revealing robot state through lights may have a direct functional role, we can also consider non-functional uses of lights. In particular, lights can be considered to be a part of the robot's appearance, which allows us to potentially change the latter over the days in order to adapt to personal preferences of users repeatedly interacting with the robot over long periods of time. This will be the focus of the last part of this thesis, in which we enable a robot to modify its appearance at appropriately timed intervals in order to maintain *engagement* (modeled through the form of rewards) of users with different profiles.

Throughout this thesis, we will focus our analysis on an *autonomous mobile service robot*, CoBot, which provides different types of services to people in a building across multiple floors. However, the ideas we present are made general enough to be easily ported to different platforms and choices than the ones made in this thesis. Nevertheless, CoBot constitutes an interesting platform for the purpose of expression of state information to humans through lights. Indeed, the ways in which CoBot interacts with humans are diverse: it can ask for help from humans when facing limitations (so-called *symbiotic autonomy* [40]), influence change in human behavior for the needs of its tasks, or provide useful information for the task at hand. The spectrum of information potentially communicated through expressive lights is hence greatly diverse and non-simplistic.

In summary, this thesis will be investigating the use of lights in two different ways. First, making elements of a robot's state visible to humans in an abstracted fashion through light animations which are *informative*, i.e., providing useful information for the situation at hand, and *legible* i.e., whose meaning is intuitive and doesn't require training. Second, we use lights for modulating the appearance of the robot, regardless of function, in repeated interactions with the same users. We focus on making these lights *engaging*, specifically, adapting to personal preference dynamics.

1.2 Approach

Figure 1.1 summarizes the approach taken to map robot state information to expressive light animations. We assume the robot has an available expression channel linked to its state (in our case expressive lights, whose expression space and control will be analyzed in depth). According to the nature of the expression channel, we express relevant state information in

a format which makes it easy to map to an expression on the expression channel (labeled ‘Expressible state information’ in Figure 1.1). We then investigate the details of the robot state / light animation mapping by running appropriate user studies. Finally, we give the robot agency in the selection of its light animations in the context of repeated interactions with the same user.

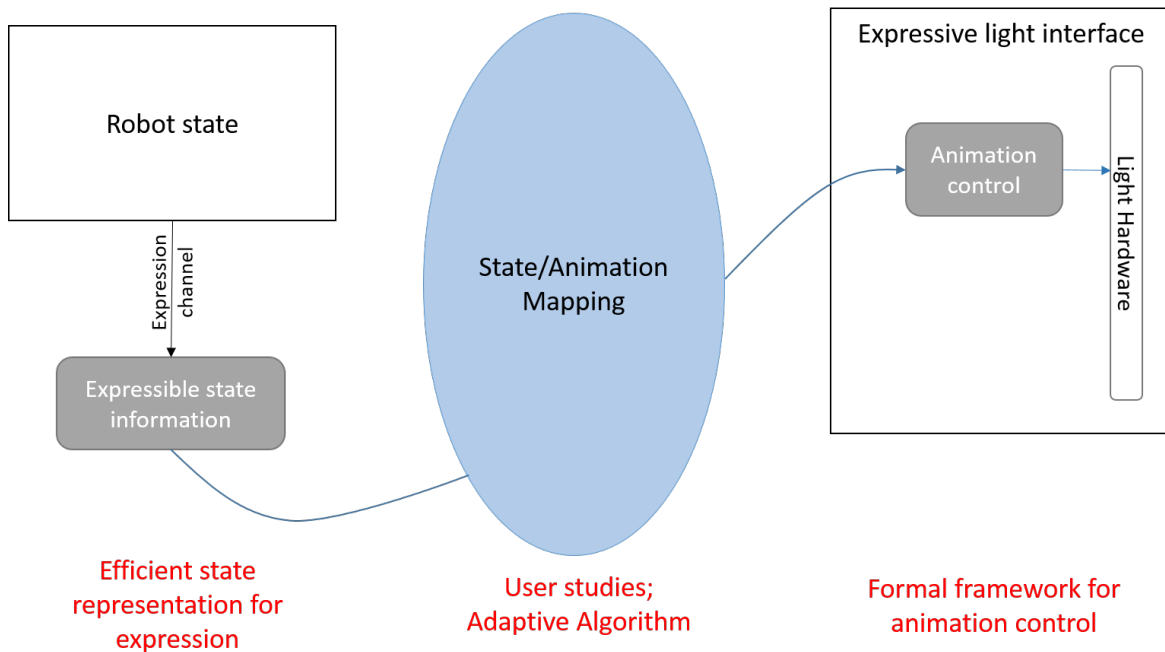


FIGURE 1.1 Overview of the approach taken in this thesis (contributions are shown in red)

1.3 Contributions

The contributions of this thesis are as follows:

- A *formal framework for animation control* focusing on addressable light strips and generalizable to other forms of light arrays,
- An *efficient state representation and mapping method* suited for expression of state information,
- *User studies* to investigate the design and impact of animating robot state through expressive lights, and
- An *adaptive algorithm* to personalize the change in these light expressions over repeated interactions for different *user profiles* that we introduce.

1.4 Reader's guide to the thesis

- Chapter 2 summarizes relevant literature on expressive lights, their use in technology and robots, general non-verbal robot communication, and personalization in Human-Robot Interaction (HRI).
- Chapter 3 lays out our formalism for light animation and specifically control of addressable RGB light strips.
- Chapter 4 presents a general framework for mapping a mobile robot's state to an expression such as a light animation.
- Chapter 5 reports three studies that were conducted to inform: (1) the design of appropriate light animations (parameter selection), (2) their evaluation and generalization to similar scenarios, and (3) their impact on human behavior in the real world.
- Chapter 6 demonstrates the integration of agency to light expression in the context of personalization of temporal dynamics in long-term human-robot interactions.

Chapter 2

Related Work

This thesis will be focusing on the use of lights as a communication medium for mobile robots. In this chapter, we therefore first present a short overview of general uses of lights in different applications and how they have previously been used as an expressive medium. We then focus our review on lights on robots and other existing expressive non-verbal modalities. We finally present a general overview of personalization in Human-Robot Interaction (HRI) which will be the subject of chapter 6.

2.1 Short survey of uses of lights

2.1.1 Lights for communication at a distance

Light signals have been widely used in the history of mankind to convey information at a distance or in low visibility environments, such as in aviation and maritime navigation [28], where the use of acoustic signals is not possible because of signal attenuation. However, most of these signals often need to be learned since they relied on codes (an extreme case being Morse code communication before radio technologies existed).

Nowadays, we see lights in our daily lives for communication at a distance, especially on roads thanks to indicators such as traffic lights which control traffic flow through a simple color code, or lights on cars such as flashers communicating upcoming driver actions, emergency flashers expressing an unusual or dangerous situation, break warning rear lights indicating the status of the brake, and headlights sometimes used by drivers to acknowledge or communicate with other cars or pedestrians.

Light also plays a role in some biological systems especially in animal communication. Through a process called bioluminescence [33], some animal species such as jellyfish, octopus, anglerfish, or fireflies are capable of emitting light to deceive, attract, or communicate different messages to other animals [1].

2.1.2 Lights for revealing state information

As discussed in the previous paragraph, cars are a good example where lights are used to reveal information about the car or the driver's state. Similarly, personal electronic devices and appliances often make use of light indicators with usually intuitive, walk-up-and-use patterns to convey information to the user. We see light indicators on all sorts of devices including cell phones, washing machines, toasters, laptops, cameras, battery chargers and more.

Such uses pose the problem of a *mapping* from a concrete state of the device to an abstract visualization into a light animation, which we will be investigating in chapter 5. We can exploit the fact that humans tend to generalize from their daily experience to get inspiration in our light animation design from standard patterns, codes and meanings associated with some of these existing light behaviors.

2.1.3 Lights and aesthetics

Because of their visual aesthetic appeal and wide flexibility in configuration and modularity, expressive lights have been featured extensively in contemporary art including interactive art installations [24], bridge art (e.g., the Pausch bridge at Carnegie Mellon ¹) or large scale visualizations of data such as the state of the Internet [22]. Expressive lights have also been used on wearable apparel [13].

Stage and scene lighting share common expressive features with indicator lights like color, intensity and time-varying patterns [15], but there the purpose is to illuminate rather to use the light source itself as an expressive communication modality. The same holds for modern programmable luminaires, mainly used for creating diverse moods in physical spaces [23].

2.2 Light as an expressive medium

Because of the way humans process visual information, which includes color and motion as “undoubtable attributes” guiding the human attention process [49], programmable multi-color lights which combine color and motion to create light animations are a good candidate for changing a robot's appearance in order to communicate different types of information to humans.

2.2.1 Light control

Addressable LED technology has unlocked fine control of multiple light sources with possible color changes and gave rise to several control protocols of which DMX512 is probably the most popular, especially for large installations or multiple light devices ². Color Kinetics ³

¹<http://www.cmu.edu/randyslecture/bridge.html>

²<http://opendmx.net/index.php/DMX512-A>

³<http://www.colorkinetics.com/>

offers a wide variety of tools to design and control large-scale light installations through various animations.

Smaller scale options include addressable RGB LED strips on which will be focusing in this thesis. Each LED has a built-in microcontroller which controls the associated LED intensity as a function of time. By propagating serial communication packets throughout the strip, one can achieve appealing light animations using only one physical data communication pin⁴.

2.2.2 Light expression space

Light signals generally allow for a large expression space with several degrees of freedom. For example, light communication in insects have been found to use modulations in spectral composition, brightness, shape, size, and timing [33]. There has been efforts to standardize color spaces (RGB, HSV etc.) but not much work has been done when it comes to standardizing light patterns [21]. Also, parametrized abstract motifs have been used for spatial layout modulation [8].

Current LED technology offers control of brightness and color for individual light sources and allows for customizable layout options⁵, unlocking very flexible designs for light animations.

2.2.3 Light animation semantics

Because of the wide use of lights to convey concrete information, it seems that humans tend to associate specific *meanings* to different light animations.

For a single light source of a fixed color, different light patterns seem to convey diverse information about a personal device's operation [21]. In a traffic setting, flashing lights for instance seem to be associated with the idea of warning (such as a non-functional traffic or an emergency car flasher). Also, some studies have been conducted on the conspicuity of light patterns as a function of frequency, duration and contrast [18], but also on the perception of emergency warning signals especially in terms of color combinations [14]. Color has also been associated to the expression of emotions in different contexts, such as general perception of color [51], uses in clothing [13], or on virtual agents [37], [15].

Color theory [51] as well as learned color codes (e.g., traffic lights) provide a good starting point for the design of colored light animations carrying meaning (in our case related to robot state). However, it remains difficult to predict the appropriateness of colored animations for light sources extending in space beyond a single point (namely a light strip) and expressing meaning in relation to a complex machine such as a robot.

⁴<https://learn.adafruit.com/adafruit-neopixel-uberguide/advanced-coding>

⁵<http://www.colorkinetics.com/>

2.3 Robot expression

2.3.1 Lights on robots

The use of lights for non-verbal communication on robots remains rudimentary. Most of these uses do not have a direct functional role but rather focus on creating abstract impressions (such as “artificial subtle expressions” [27]), expressing emotions [25], or serving as very basic indicators (such as for battery level). Often, we see these light expressions dissociated from the robot’s state, such as for instance expressing people’s emotions in a cafe-style room on a Roomba robot [39]. To the best of our knowledge, the only instances of functional and state-related light communication in robots are for human-robot speech synchronization [17] and for communicating intent in robot navigation [44]. In [17], an animated LED is used to avoid utterance collisions in verbal human-robot communication by subtly blinking between the user’s speech end and the robot’s speech start. In [44], an array of LED’s are used to communicate direction of navigation on a quadcopter. This last work fits within our idea of expressing a part of the robot’s state through light animations. However, in the prior work, the expressed feature (directionality) remains a low-level one and the light expression has a low level of abstraction. In contrast, we will be focusing on higher-level features of the robot’s state related to the robot’s tasks in the (often unpredictable) environment.

2.3.2 Other non-verbal modalities for robot expression

Several non-verbal modalities have been considered for human-robot communication. Some of them such as eye gaze [2] or proxemics [9] are more suited for expressing an emotional or affective state of the robot. In this thesis, we are interested in expressing robot states that are related to tasks and there exists two main other non-verbal modalities which could be used for such expression. The first modality is expressive motion, which has been studied in different settings such as manipulators expressing goal and planning information [16] and for mobile robots expressing affective states [26]. The type of mapping problem in which we are interested in this thesis has been studied for a continuous input signal such as music or speech being mapped to motion sequences on a humanoid robot [52]. The second modality is expressive sounds which hasn’t been studied much for robotics applications and is more prevalent in electronic devices (e.g., phone notifications).

2.4 Personalization in HRI

In chapter 6, we introduce the personalization of light expressions according to different user profiles. Below we present previous related work on general personalization for human-robot interaction (HRI).

Apart from simple customization during usage [43], recent work has looked at autonomous personalization of HRI based on previous interactions with the same user. Examples include a snack delivering robot which uses data from past interactions to personalize the future interactions [29] or a humanoid robot learning different models for expressing

emotion through motion, which it is then able to use for personalizing expression of emotions [47]. Furthermore, the idea of self-initiative in a robot has been explored by learning ways of acting in the world depending on user verbal feedback on the current "state of the world" [35]. Finally, user modeling for long-term HRI, a focus of the current paper, has been looked at using archetypes of real users called *personas*, which encode traits of potential users in terms of interaction preferences [12]. Some authors also looked at ways of learning long-term behavior by identifying social primitives that are important when the novelty aspect of interaction vanishes [31] or matching personalities between robot and user [45]. However, these works focus more on the social aspect of the interaction rather than on the intelligence of the adaptation from a generic point of view, making their applicability and generalization poor in different types or modes of interaction. In the last chapter of this thesis, we will be decoupling the nature of the interaction options from the adaptation mechanism, which can then be tuned based on the nature of the interaction and the user's response to it.

Chapter 3

Animating light sources

The goal of this chapter is to provide a framework for animating a set of fixed light sources. This framework will help us design appropriate animations in a simple fashion and facilitate the process of mapping robot state space to light animation space. We begin by defining the concept of a light animation and reduce the dimensionality of the very large animation space by introducing a finite number of parametrized signal shapes. We then present the algorithms used to dynamically control a digital light strip according to the presented animation options. Our framework can be easily extended to other signal shapes or to multiple light strips.

3.1 Light animation and animation space definitions

3.1.1 Light animation as a continuous intensity function matrix

Definition 3.1. An animation $A(t)$, within a three-dimensional color space, of a set of n fixed point source lights is defined as a time-varying $n \times 3$ matrix of light intensities:

$$A(t) = \begin{pmatrix} i_{1c_1}(t) & i_{1c_2}(t) & i_{1c_3}(t) \\ i_{2c_1}(t) & i_{2c_2}(t) & i_{2c_3}(t) \\ \vdots & \vdots & \vdots \\ i_{nc_1}(t) & i_{nc_2}(t) & i_{nc_3}(t) \end{pmatrix} \quad (3.1)$$

where the rows represent the indices of the individual point source lights or *pixels* and the columns represent dimensions of the color space or *color channels* c_1 , c_2 and c_3 (e.g., RGB, XYZ [41]). The intensity values represent a percentage of an allowed maximum intensity, and hence:

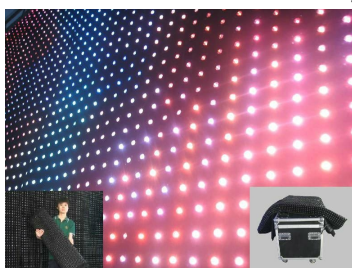
$$\forall t: 0 \leq i_{jc_k}(t) \leq 100 \quad j = 1, \dots, n; \quad k = 1, 2, 3$$

3.1.2 Spatial layout

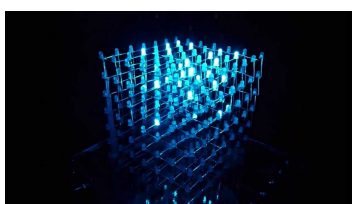
The animation matrix above does not capture the spatial layout of these pixels, which could be arranged in a linear, planar or three-dimensional fashion, as is shown in Figure 3.1.



(A) Light strip (Adafruit NeoPixel), retrieved from https://blog.adafruit.com/wp-content/uploads/2013/08/1507_LRG-600x461.jpg



(B) Light fabric (Huasun fabric LED screen), retrieved from <http://img.diytrade.com/cding/1951300/28703279/0/1348890009/>



(C) Light cube (Christopher Sauer), retrieved from <https://i.ytimg.com/vi/6bSUmBWXfK8/maxresdefault.jpg>

FIGURE 3.1 Examples of different spatial layouts for a set of lights

For the rest of this work, we will focus on linear light strips. They simplify the analysis and representation of light animations and allow for greater mounting flexibilities from a physical point of view. For linear strips, we let the pixel index (row index of the animation matrix) represent the position of the pixel on the strip, along a predefined direction.

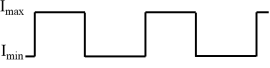
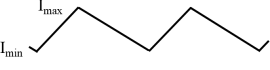


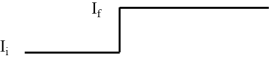
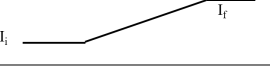
3.1.3 Animation space intensity functions

The space to which $A(t)$ belongs is very large, as each of the intensity functions $i_{jc_k}(t)$ can be any continuous function of t bounded between 0 and 100 and hence belongs to an infinite-dimensional space. As a result, we will only focus on a limited set of possible intensity functions $i_{jc_k}(t)$, which we group into classes. We call these classes *parametrized signal shapes*, summarized in Table 3.1. There are plenty of light animations (either in libraries for light strips ¹, or in art projects using lights [22]) whose goal is to create aesthetically pleasing

¹https://github.com/adafruit/Adafruit_NeoPixel

animations. These types of animations are usually sophisticated, fast-changing in space, time or color. We will be focusing on the simple signal shapes shown in Table 3.1 which, through efficient parametrization, will still provide us with great flexibility.

TABLE 3.1 List of parametrized signal shapes considered for each $i_{jc_k}(t)$

Periodic	Blink (rectangle waveform) 
	Fade in/out (triangle waveform [possibly asymmetric]) 
	Smooth fade in/out (sinusoidal waveform) 
	Irregular Blink (modulated waveform) 
Non-periodic	Abrupt intensity/color change (step function) 
	Slow intensity/color change (clipped ramp function) 

As a result of considering a handful of parametrized signal shapes, we have now reduced the dimensionality of the animation space drastically, and the analysis of this space is now simpler to work with.

3.1.4 Animation tuple representation

For compactness of representation, we define an *animation tuple*, which fully characterizes the animation behavior for a subset of the light pixels.

Definition 3.2. An *animation tuple* is defined as a tuple $\langle sh, \mathbf{p}_{sh}, j_{start}, j_{end} \rangle$, where:

- sh is a string identifier for the signal shape used in the animation. It can take on the following values: “rect” for rectangle waveform, “tri” for triangle waveform, “sinl” for sinusoidal waveform, “modw” for modulated waveform, “step” for step function, and “ramp” for clipped ramp function
- $\mathbf{p}_{sh} = (p_1^{sh}, p_2^{sh}, \dots, p_{m_{sh}}^{sh})$ is a vector of parameters associated with that particular signal shape, where m_{sh} is the number of parameters for shape sh
- j_{start} and j_{end} are the indices of the start and end pixel, respectively, to be animated. In other words, the animation described by sh and \mathbf{p}_{sh} is applied to pixels j_{start} up to j_{end} , inclusive

The behavior of a full light strip of is represented by a set of animation tuples $\{\langle \text{sh}_1, \mathbf{psh}_1, j_{\text{start}_1}, j_{\text{end}_1} \rangle, \dots, \langle \text{sh}_m, \mathbf{psh}_m, j_{\text{start}_m}, j_{\text{end}_m} \rangle\}$ such that:

$$\begin{aligned} \bigcup_{i=1}^m (\{j_{\text{start}_i}, \dots, j_{\text{end}_i}\}) &= \{1, \dots, n\} \\ \bigcap_{i=1}^m (\{j_{\text{start}_i}, \dots, j_{\text{end}_i}\}) &= \emptyset \end{aligned} \quad (3.2)$$

where n is the number of pixels on the strip. In other words, first, the pixel indices must cover the whole strip and, second, there should be no overlap in the specified pixel ranges. We call an animation tuple set satisfying conditions 3.2 a *complete animation tuple set*. It fully characterizes an animation of a whole light strip.

3.2 Signal shape parametrization

We now present a detailed description of each of the parametrized signal shapes shown in Table 3.1. For periodic signals of period T , we refer to the portion of the signal extending from $t = zT$ to $t = (z+1)T$ (where $z \in \mathbb{Z}$) as a *cycle*.

3.2.1 Rectangle waveform

The parameter vector \mathbf{p}_{rect} for this periodic waveform is composed of the following components:

- $(I_{c_1, \text{min}}, I_{c_2, \text{min}}, I_{c_3, \text{min}}) \equiv p_1^{\text{rect}}, (I_{c_1, \text{max}}, I_{c_2, \text{max}}, I_{c_3, \text{max}}) \equiv p_2^{\text{rect}}$: the minimum and maximum intensity values, respectively, of color channels c_1, c_2 and c_3 (in %)
- $T \equiv p_3^{\text{rect}}$: the period (in absolute time unit)
- $D_{\text{rect}} \equiv p_4^{\text{rect}}$: the fraction of the period in which the signal is maximal

A rectangle waveform $\text{rect}(t)$ on a color channel c_k is defined by:

$$\begin{aligned} \text{rect}(t) &= \begin{cases} I_{c_k, \text{max}} & 0 \leq t < D_{\text{rect}}T \\ I_{c_k, \text{min}} & D_{\text{rect}}T \leq t < T \end{cases} \\ \text{rect}(t) &= \text{rect}(t + zT) \quad z \in \mathbb{Z} \end{aligned} \quad (3.3)$$

3.2.2 Triangle waveform

The parameter vector \mathbf{p}_{tri} for this periodic waveform is composed of the following components:

- $(I_{c_1, \text{min}}, I_{c_2, \text{min}}, I_{c_3, \text{min}}) \equiv p_1^{\text{tri}}, (I_{c_1, \text{max}}, I_{c_2, \text{max}}, I_{c_3, \text{max}}) \equiv p_2^{\text{tri}}$: the minimum and maximum intensity values for each color channel (in %)

- $T \equiv p_3^{\text{tri}}$: the period (in absolute time unit)
- $D_{\text{tri}} \equiv p_4^{\text{tri}}$: the ratio of the rise time to the period

A triangle waveform $\text{tri}(t)$ on a color channel c_k is defined by:

$$\text{tri}(t) = \begin{cases} \frac{I_{c_k,\text{max}} - I_{c_k,\text{min}}}{D_{\text{tri}} T} t + I_{c_k,\text{min}} & 0 \leq t < D_{\text{tri}} T \\ \frac{I_{c_k,\text{max}} - I_{c_k,\text{min}}}{1 - D_{\text{tri}}} \left(-\frac{t}{T} + 1\right) + I_{c_k,\text{min}} & D_{\text{tri}} T \leq t < T \end{cases} \quad (3.4)$$

$$\text{tri}(t) = \text{tri}(t + zT) \quad z \in \mathbb{Z}$$

3.2.3 Sinusoidal waveform

The parameter vector \mathbf{p}_{sinl} for this periodic waveform is composed of the following components:

- $(I_{c_1,\text{min}}, I_{c_2,\text{min}}, I_{c_3,\text{min}}) \equiv p_1^{\text{sinl}}$, $(I_{c_1,\text{max}}, I_{c_2,\text{max}}, I_{c_3,\text{max}}) \equiv p_2^{\text{sinl}}$: the minimum and maximum intensity values for each color channel (in %)
- $T \equiv p_3^{\text{sinl}}$: the period (in absolute time unit)

A sinusoidal waveform $\text{sinl}(t)$ on a color channel c_k is defined by:

$$\text{sinl}(t) = \sin\left(\frac{2\pi}{T}t - \frac{\pi}{2}\right) \frac{I_{c_k,\text{max}} - I_{c_k,\text{min}}}{2} + I_{c_k,\text{min}} \quad (3.5)$$

3.2.4 Modulated waveform

This is a special kind of waveform combining several cycles from the three previous periodic signal shapes to create a “supercycle”. This “supercycle” is then periodically repeated. The parameter vector \mathbf{p}_{modw} for this periodic waveform is composed of the following components:

- $n_{\text{sub}} \equiv p_1^{\text{modw}}$: the number of subcycles in one supercycle.
- $\mathbf{v}_{\text{sup}} = (v_{\text{sub},1}, \dots, v_{\text{sub},n_{\text{sub}}}) \equiv p_2^{\text{modw}}$: a vector of n_{sub} shape identifier - shape parameters pairs $v_{\text{sub},i} = (\text{sh}_{\text{sub},i}, \mathbf{p}_{\text{sub},i})$, describing the light behavior in each subcycle.

A modulated waveform $\text{modw}(t)$ on a color channel c_k is defined by:

$$\text{modw}(t) = \begin{cases} \text{sh}_{\text{sub},1}(t) & 0 \leq t < T_1 \\ \vdots & \vdots \\ \text{sh}_{\text{sub},i}(t) & T_{i-1} \leq t < T_i \\ \text{sh}_{\text{sub},n_{\text{sub}}}(t) & T_{n_{\text{sub}}-1} \leq t < T_{n_{\text{sub}}} \end{cases} \quad (3.6)$$

$$\text{modw}(t) = \text{modw}\left(t + z \sum_{i=1}^{n_{\text{sub}}} T_i\right) \quad z \in \mathbb{Z}$$

where T_i represents the period of the i^{th} subcycle.

3.2.5 Step function

The parameter vector \mathbf{p}_{step} for this periodic waveform is only composed of the following component:

- $(I_{c_1,f}, I_{c_2,f}, I_{c_3,f}) \equiv p_1^{\text{ramp}}$: the final intensity value for each color channel (in %). The initial intensity value $(I_{c_1,i}, I_{c_2,i}, I_{c_3,i})$ (previous state of the strip) is not relevant for animating the strip from $t=0$ onward, so it is not included in the parameter vector.

A step function $\text{step}(t)$ on a color channel c_k is defined by:

$$\text{step}(t) = \begin{cases} I_{c_k,i} & t < 0 \\ I_{c_k,f} & t \geq 0 \end{cases} \quad (3.7)$$

3.2.6 Clipped ramp function

The parameter vector \mathbf{p}_{ramp} for this periodic waveform is composed of the following components:

- $(I_{c_1,i}, I_{c_2,i}, I_{c_3,i}) \equiv p_1^{\text{ramp}}$, $(I_{1,f}, I_{2,f}, I_{3,f}) \equiv p_2^{\text{ramp}}$: the initial and final intensity values for each color channel (in %)
- $t_{\text{rise}} \equiv p_3^{\text{ramp}}$: the rise time (in absolute time unit)

A clipped ramp function $\text{ramp}(t)$ on a color channel c_k is defined by:

$$\text{ramp}(t) = \begin{cases} I_i & t < 0 \\ \frac{I_f - I_i}{t_{\text{rise}}} + I_i & 0 \leq t < t_{\text{rise}} \\ I_f & t \geq t_{\text{rise}} \end{cases} \quad (3.8)$$

For all of these animations, note that, in a Red-Green-Blue (RGB) color space ($c_1 = R$; $c_2 = G$; $c_3 = B$), if the *color ratios* r_1 and r_2 listed below are constant as a function of time, then the animation is of a single color. If the ratio is not respected however, we would observe color changes throughout the animation. Color ratios r_1 and r_2 are defined as:

- $r_1 \equiv I_{R,\text{max}} : I_{G,\text{max}} : I_{B,\text{max}}$ and $r_2 \equiv I_{R,\text{min}} : I_{G,\text{min}} : I_{B,\text{min}}$ for the periodic signal shapes, and
- $r_1 \equiv I_{R,f} : I_{G,f} : I_{B,f}$ and $r_2 \equiv I_{R,i} : I_{G,i} : I_{B,i}$ for the non-periodic signal shapes

The case where $r_2 = 0 : 0 : 0$ results in a single color animation for any r_1 values.

3.3 Animating a digital RGB LED strip

The animation model described in the previous sections is useful for design and visualization purposes. However, this model presents us with some limitations in practice when working with digital addressable LED strips. For the rest of this chapter, we assume we are working in a RGB color space.

3.3.1 Light animation as a sequence of frames

Most light strips nowadays are digital LED strips and therefore have a finite refresh rate f_{refresh} . This means that the intensity functions $i_{jc}(t)$ ($c = R, G, B$) are actually synchronized discrete signals $i_{jc}[l] = i_{jc}(l \cdot \Delta t)$ where $\Delta t = \frac{1}{f_{\text{refresh}}}$. Moreover, the color levels are quantized (usually into 256 levels) for each Red, Green or Blue color channel. The combined discrete $i_{jc}[l]$ values form the discrete-valued matrix $\hat{A}[l]$, which we call an *animation frame*.

Definition 3.3. An *animation frame* $\hat{A}[l]$, at time step l , for a 256-level RGB digital LED strip containing n pixels is defined as:

$$\hat{A}[l] = \begin{pmatrix} i_{1R}[l] & i_{1G}[l] & i_{1B}[l] \\ \vdots & \vdots & \vdots \\ i_{nR}[l] & i_{nG}[l] & i_{nB}[l] \end{pmatrix} \quad (3.9)$$

where the rows represent the pixel indices and the columns represent the R, G and B color channels. The intensity values are discretized and quantized such that:

$$\forall l \in \mathbb{N}: i_{j,c}[l] \in \mathbb{N}, 0 \leq i_{j,c}[l] \leq 255 \quad j = 1, \dots, n; \quad c = R, G, B$$

3.3.2 Episodic animation control

The animations as described in the previous paragraph start at time step $l = 0$ and extend arbitrarily in time. However, if these lights are going to be used to express a dynamic process, such as the varying state of a robot, then frequent switches from one animation to another will be needed, hence the notion of episodic animation introduced next.

Definition 3.4. An *animation episode* is defined as a fixed portion of a particular animation. Animation episodes can be arbitrarily defined; for the signal shapes we considered, we define an episode to be:

- A single signal cycle, for periodic signal shapes, and
- The portion of the signal needed to transition from the initial to the final intensity value, for non-periodic signal shapes.

For a complete animation tuple set containing different animations for different pixels, each with its own episode, we define the complete animation episode to be lasting as long as the shortest of the individual episodes.

Based on the above definition, we propose an *episodic animation control algorithm*, which given a dynamic process DP of time-varying complete animation tuple sets, generates a sequence of animation episodes, as shown in Figure 3.2. For illustration purposes, we assume that pixels 1 through n have the same animation, so we only show the corresponding animation tuple (the complete animation tuple set in this case is a singleton containing that tuple). The algorithm used to achieve this behavior is summarized in Algorithm 1. At the

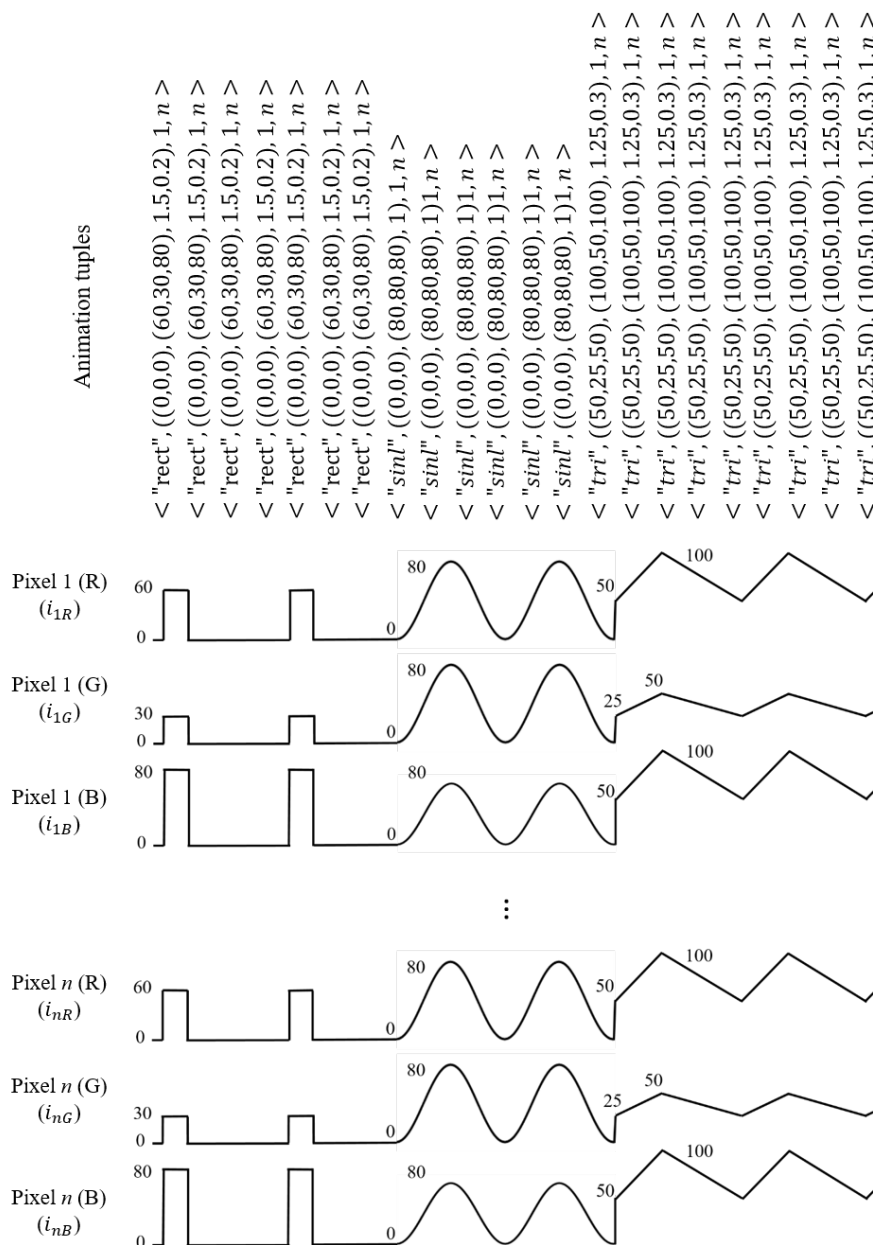


FIGURE 3.2 Episodic animation example: animation tuple as a function of time and corresponding intensity functions for each pixel of the strip

beginning of each episode, the algorithm gets the current animation tuple set value from DP and animates the strip with one episode of the animation corresponding to that tuple set. We assume that the choice of episode length is small compared to the dynamics of DP, in other words the minimum switching time for the animation tuple set values is greater than any single animation episode.

Algorithm 1 Episodic animation control algorithm RunAnim

```

1: procedure RUNANIM(DP)
2:   tuplenow ← NULL
3:   StartTimer(t)                                ▷ t globally stores the time elapsed in seconds
4:   while true do
5:     tupsetprev ← tupsetnow
6:     tupsetnow ← FetchAnimTuple(DP)
7:     if tupsetprev ≠ tupsetnow then ResetTimer(t)
8:     Episode(tupsetnow)
9:   procedure EPISODE( $\{\langle sh_1, \mathbf{psh}_1, j_{start_1}, j_{end_1} \rangle, \dots, \langle sh_m, \mathbf{psh}_m, j_{start_m}, j_{end_m} \rangle\}$ )
10:     $\{e_1, \dots, e_m\} \leftarrow \text{GetEpisodeLengths}(\mathbf{psh}_1, \dots, \mathbf{psh}_m)$ 
11:     $e_{\min} \leftarrow \min_i e_i$ 
12:    while  $t < e_{\min}$  do
13:       $\hat{A} \leftarrow \text{GetFrame}(\{\langle sh_1, \mathbf{psh}_1, j_{start_1}, j_{end_1} \rangle, \dots, \langle sh_m, \mathbf{psh}_m, j_{start_m}, j_{end_m} \rangle\}, t)$ 
14:      UpdateStrip( $\hat{A}$ )                                ▷ Updates the strip with the current frame
15:    procedure GETFRAME( $\{\langle sh_1, \mathbf{psh}_1, j_{start_1}, j_{end_1} \rangle, \dots, \langle sh_m, \mathbf{psh}_m, j_{start_m}, j_{end_m} \rangle\}, t$ )
16:      for  $i \leftarrow 1, \dots, m$  do
17:        for  $j \leftarrow start_i, \dots, end_i$  do
18:           $\hat{A}_{j,1:3} \leftarrow (\text{round}(\frac{255}{100} \cdot i_{j,R}(t)), \text{round}(\frac{255}{100} \cdot i_{j,G}(t)), \text{round}(\frac{255}{100} \cdot i_{j,B}(t)))$ 

```

Let d_{frame} be the total delay between two animation frame updates. d_{frame} comprises several sources of delays including the time to compute the frame and the time to refresh the strip. Note that for a specific animation, $d_{frame} = \Delta t = \frac{1}{\text{Refresh Rate}}$. Since these delays can differ from frame to frame, it is best to keep track of time using a timer and sample the original $A(t)$ curves at the corresponding time values, as shown in the “Episode” procedure of Algorithm 1, to get the corresponding discrete and quantized frames $\hat{A}[l]$, computed by the “GetFrame” procedure.

As a final note, we can easily extend our framework to more than one light strip to be independently animated. For N light strips, each will have its own dynamic process $DP_i, i = 1, \dots, N$ to be animated. Let P_1, \dots, P_N be independent control process, one for each strip. Each P_i will be independently running Algorithm 1 on DP_i to episodically control light strip i .

3.4 Chapter summary

In this chapter, we presented a formal framework for representing and controlling light animations, summarized in Figure 3.3. We first introduced a representation of light animations as complete sets of animation tuples. We then focus on the control of RGB LED strips and present an episodic animation control algorithm to enable the translation of a dynamic process of complete animation tuple sets into a sequence of frames on a light strip. Our framework (Arduino code available online ²) is not platform-specific and can be used by any device using the communication protocol summarized in Appendix A.

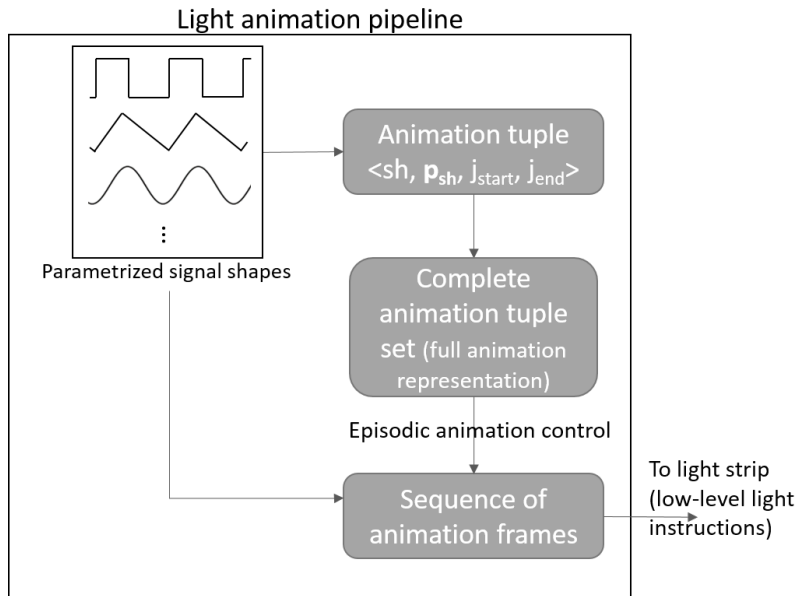


FIGURE 3.3 Summary of the animation framework described in this chapter

²<https://github.com/kobotics/LED-animation>

Chapter 4

Mobile service robot state and its expressible elements

In this chapter, we first introduce a representation of robot state that is suited for expression of elements of robot state on a given expression channel (expressive lights in our case). Our analysis focuses on CoBot, a collaborative mobile service robot with diverse capabilities. We then discuss the process of mapping the robot state information to light animations with respect to the formalism presented in the previous chapter. We illustrate the ideas presented by selecting *informative* elements of CoBot’s state to be displayed to humans using expressive lights. We then discuss the generalizability of our mapping framework to multiple light strips / modalities expressing different aspects of the robot’s state. We end the chapter by providing details about how the mapping framework was implemented on the CoBot robot with two light strips expressing task-related and navigation-related aspects of the robot’s state, respectively.

4.1 CoBot overview

In this thesis, we use CoBot, a collaborative mobile service robot, as an example to illustrate the ideas presented. CoBot can perform a set of services to humans in a building across multiple floors. Building locations (rooms, kitchens, and elevators), as well the navigation map of the building, are known to the robot. CoBot robustly navigates autonomously [46] from location to location while avoiding obstacles during its navigation, whenever possible, or stopping in front of unavoidable obstacles such as humans obstructing its path. When facing limitations (such as pressing the button of an elevator or loading/unloading an object on/from the robot), the robot asks for help from humans. This is the main idea behind *symbiotic autonomy* [40], which enables a robot to overcome its limitations, as shown in Figure 4.1.

4.1.1 CoBot tasks and services

The *tasks* performed by CoBot are of one of two types:



FIGURE 4.1 An example of symbiotic autonomy: CoBot getting help from a human at an elevator

- *Navigation tasks* involve navigating from a start location to a goal location according to a predefined navigation map on which a path planning algorithm is run, and
- *Human interaction tasks* involve asking for human help (referred to as an ‘ask’ task) or waiting for human input, such as confirmation, service initiation, or dismissal (referred to as a ‘wait’ task).

Tasks can be combined to form *services*. The three services offered by CoBot and considered in this thesis are the following:

- **Go-to-Room** service, in which the robot goes from its current position to a goal location.
- **Item-transport** service, in which the robot transports an item in its basket from a start location to a goal location.
- **Escort** service, in which the robot escorts a person from a start location (typically an elevator) to a goal location.

Figure 4.2 shows CoBot performing some of these services.

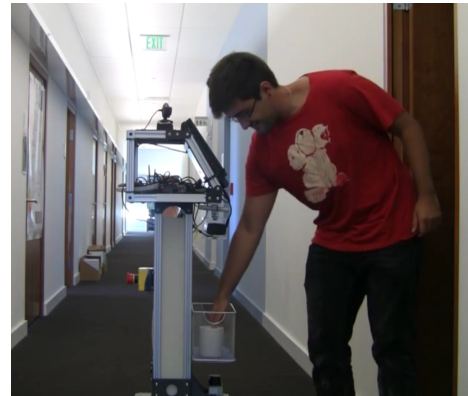
An example of a service broken down into individual tasks is shown below.

Example 4.1. Sample service: Transport an object from room 7002 (on the 7th floor) to 3201 (on the 3rd floor):

- Task 1: *Navigate* from current location to service start location (room 7002).
- Task 2: *Wait* for human to put the object to be transported in basket, and get confirmation.



(A) CoBot escorting a person



(B) CoBot transporting a coffee mug to an office

FIGURE 4.2 CoBot performing different services: (A) an escort service; (B) a transport service

- Task 3: *Navigate* from room 7002 to the 7th floor elevator.
- Task 4: *Ask* for human assistance to take the elevator to the 3rd floor.
- Task 5: *Navigate* inside the elevator.
- Task 6: *Ask* for human assistance to know when correct floor is reached.
- Task 7: *Navigate* out of the elevator.
- Task 8: *Navigate* from the 3rd floor elevator to service goal location (room 3201).
- Task 9: *Wait* for human to collect object and get service completion confirmation.

4.1.2 CoBot user modalities

Robot services can be requested in three different ways, corresponding to three different input modalities:

- Through the robot's touch screen: A Graphical User Interface (GUI) enables scheduling services from CoBot itself by letting users choose the service type and its required parameters. It can also be used to interrupt the execution of task if needed.
- Through the speech interface [38]: The GUI also includes a button that enables speech interaction with the robot. The user can issue service commands using simple structured language that the robot understands.
- Through a web interface: People in the building can schedule a robot service in a time window through a web interface.

4.1.3 Robot motion modes

Mobility constitutes an important distinguishing feature of CoBot compared to other types of robots. We can distinguish three different motion modes in which the robot can be, summarized below.

- **Moving:** in this mode, the robot is executing a navigation task successfully. We distinguish these two cases:
 - the robot moves *with someone* (escort service),
 - the robot moves *on its own* (all other tasks).
- **Stopped:** in this mode, the robot is not moving, which can be due to several reasons. We also distinguish two cases:
 - the robot is *intentionally* stopped, either because it is idle, or because it is performing an interaction task.
 - the robot is *forced* to be stopped, because of some unexpected event such as the presence of an obstacle or internal failure. In the presence of an obstacle, the robot says “Please excuse me” to incite any human obstacles to move away. If it is blocked for more than a few minutes, it will send an email to the developers for help.

In the next section we present a formalism for CoBot’s state. The formalism presented is however made general enough to easily apply to other robots with different types of services and capabilities.

4.2 Robot state representation

4.2.1 Robot variables

Definition 4.2. The *robot variables* represent any relevant quantity (discrete or continuous) that the robot maintains through its software. We categorize robot variables into *service* variables (related to the robot’s services and tasks and the planning associated with them), *execution* variables (related to task execution in the environment), and *internal* variables (related to the internals of the robot’s software and hardware).

The robot variables that we consider for CoBot are the following:

- *Service variables:*
 - the current service type `service` (‘go-to-room’, ‘transport’, or ‘escort’),
 - the current task type `task` (‘navigate’, ‘ask’, or ‘wait’),
 - the service path plan `path-plan` (list of position vertices (x , y , floor\#)),
 - the service start location `start-loc`, and

- the service goal location `goal-loc`.
- *Execution variables*:
 - the current robot location `loc`,
 - the current robot speed `speed`
 - a boolean indicator `path-blocked` for whether the robot's path is blocked, causing the robot to stop,
 - the duration of the path blockage `block-time` (None if `path-blocked` is false),
 - a boolean indicator `GUI-interrupt` for whether the robot's task was interrupted from the GUI, and
 - the duration of the interruption from the GUI `interrupt-time` (None if `GUI-interrupt` is false).
- *Internal variables*:
 - the list `sens-status` of sensor statuses (1 for normal / 0 for faulty),
 - the list `act-status` of actuator statuses (same),
 - the software status `soft-status` (1 for normal / 0 for important error),
 - The gravity of the fault if it occurs `fault-level` (e.g., on a scale from 1 to 5).
 - a boolean indicator `charging` for whether the robot is charging, and
 - the percentage battery level `batt-level`

Note that the value of some of these variables might be undefined depending on the situation the robot is in; in that case, we assign a value of None to those variables with undefined values.

4.2.2 State features and robot state

We build upon the robot variables defined in the previous section to generate states in which the robot may find itself. The robot state is determined using state features, as defined below.

Definition 4.3. Robot *state features* are discrete (usually high-level) aspects of the robot's state. They are represented as *logical expressions over robot variables*. The state features we consider are only those who are relevant to humans that potentially interact with the robot such as users, humans in the navigation environment, and developers.

The state features for CoBot considered in this thesis are listed below:

- 'Escorting': $(\text{service} = \text{'escort'}) \wedge (\text{task} = \text{'navigate'})$.
The robot is in the process of escorting the user.

- ‘Blocked by an obstacle’: $(\text{path-blocked} = \text{True}) \wedge (\text{task} = \text{navigate})$. An obstacle is impeding the navigation task progress.
- ‘Asking for help at an elevator’: $(\text{task} = \text{'ask'}) \wedge (\text{isElevator}(\text{loc}))$ (where $\text{isElevator}(a)$ returns True if location a is an elevator and False otherwise). The robot is waiting to get help at an elevator.
- ‘Charging’: $(\text{charging} = \text{True})$. The robot is connected to power and charging its battery.
- ‘Interrupted by user’: $\neg (\text{task} = \text{None}) \wedge (\text{GUI-interrupt} = \text{True})$. The robot has been interrupted by a user through the GUI.
- ‘Waiting for object loading’: $(\text{service} = \text{'transport'}) \wedge (\text{task} = \text{'ask'}) \wedge (\text{loc} = \text{start-loc})$. The robot is waiting for the object to be transported at the service start location.
- ‘Stopped because of faulty component’: $(\text{contains1}(\text{sens-status})) \vee (\text{contains1}(\text{act-status})) \vee (\text{contains1}(\text{soft-status}))$ (where $\text{contains1}(a)$ returns True if list a contains at least one 1 and returns False otherwise). One or more execution-time fault(s) occurred in the software or hardware of the robot (e.g., the LiDAR sensor gets reading errors when the robot navigates in areas with a lot of sunlight).
- ‘Waiting for dismissal’: $(\text{task} = \text{'ask'}) \wedge (\text{loc} = \text{end-loc})$. The robot is waiting for the user to dismiss it by pressing its “Done” button on its touch screen.
- ‘Navigating’: $(\text{task} = \text{navigate})$. The robot is performing a navigation task.
- ‘Turning’: $(\text{distanceFromNextVertex}(\text{loc}, \text{path-plan}) < d_{th}) \wedge (|\text{nextTurnAngle}(\text{loc}, \text{path-plan})| > \alpha_{th})$ (where d_{th} and α_{th} are thresholds for the distance to the next vertex in path-plan and for the upcoming turning angle; $\text{distanceFromNextVertex}(\cdot, \cdot)$ returns the distance between the current location loc and the next vertex in the path-plan and $\text{nextTurnAngle}(\cdot, \cdot)$ returns the upcoming turn angle α shown in Figure 4.3). The robot is about to take a turn in its navigation path.

Note that state features are not necessarily mutually exclusive: more than one state feature could be true at the same time. Since this might pose a problem for the purpose of expression (we can only visualize one or at most a few of these features at a time), we will handle this issue in section 4.4.

Definition 4.4. The robot *state* is defined as the set of all state features that are true at a given time t . Since the state of the robot is constantly changing as a function of time, the state is a dynamic process $S(t) = \{s_1, s_2, \dots, s_{m_t}\}$ where $s_{1..m_t}$ are the state features true at time t .

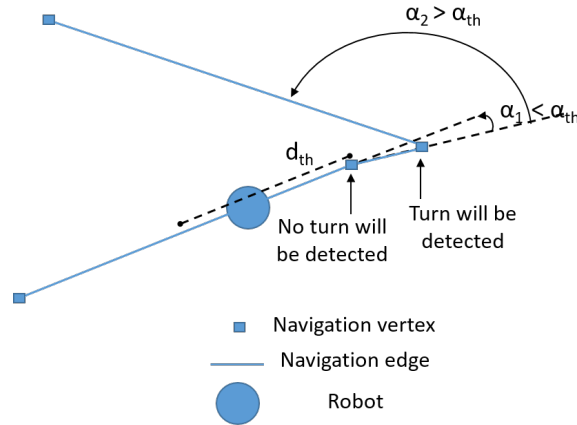


FIGURE 4.3 Turn detection check used for computation of feature ‘Turning’

4.3 What part of robot state to express?

So far, in this chapter, we have looked at robot state dissociated from robot expression. Even though state features have been defined to be relevant to users (and hence would gain at being expressed to them in some way), the actual nature of the medium used for expression hasn’t been taken into account yet in our analysis.

Definition 4.5. We denote by *expression channel* any communication medium (verbal or non-verbal) that can be potentially used to express elements of a robot’s state. The expression channel we consider is *expressive lights*.

In particular, in this section, we are interested in two aspects of state representation for the purpose of expression:

- **Modulation:** State features are a discrete representations of high-level, user-relevant elements of the robot’s state. However, this representation is rigid and doesn’t allow for modulation (no possible expression of “microstates” within the state feature or of continuous quantities that might be relevant when a particular state feature is true). For this reason, in this section we will be introducing what we call “expressible state tuples”, a data structure which takes into account modulation of expressive behaviors (such as light animations) according to modulation quantities.
- **Simplification:** For a given expression channel, there is a trade-off between complexity of the expression vocabulary (total number of different expressions used) and legibility of expression (how readable or intuitive these expressions are). For better legibility, it might be useful to group or cluster some expressible elements of states together into classes which are expressed using the same expression. For this reason, in this section we will be introducing what we call “expressible classes”.

4.3.1 Expressible state tuples

We introduce *expressible state tuples*, which are tuples containing all state information relevant to a particular robot situation and expressible on a given expression channel.

Definition 4.6. An *expressible state tuple* on a given expression channel is defined as a tuple $\langle s, \mathbf{v}_s \rangle$, where s is a state feature and \mathbf{v}_s is a vector of *modulating variables*, relevant for expression of state feature s on the considered expression channel. These additional variables can either be robot variables (defined in section 4.2.1) or variables computed from robot variables, referred to as computed variables.

To illustrate the concept of an expressible state tuple, here are a few examples based on the state features listed in section 4.2.2:

- $\langle \text{'Escorting'}, \text{percentDone}(\text{loc}, \text{path-plan}) \rangle$, where $\text{percentDone}(\cdot, \cdot)$ computes the percent distance traveled along the path plan (progress so far towards goal location); this expressible state tuple could be translated into some sort of progress indicator visible to the escorted user.
- $\langle \text{'Blocked by an obstacle'}, \text{block-time} \rangle$; this expressible state tuple could be translated into a blockage indicator which gets more noticeable as the blockage time increases.
- $\langle \text{'Charging'}, \text{batt-level} \rangle$; this expressible state tuple could be translated into a visual indicator changing as the percentage battery level increases.
- $\langle \text{'Turning'}, (\text{nextTurn}(\text{loc}, \text{path-plan}), \text{speed}) \rangle$ (where $\text{next-turn}(\cdot, \cdot)$ is a function returning 'left' or 'right' according to the direction of the upcoming navigation turn); this expressible state tuple can be translated into a turn indicator showing the upcoming turn direction.

4.3.2 Clustering expressible state tuples: expressible classes

For decreased complexity and increased legibility of expressions, we introduce in this section classes of expressible state tuples, or *expressible classes* for short. The clustering is dependent on the expression channel considered (different expression channels might require different levels of abstraction depending on their complexity/legibility tradeoff). We cluster in the same class those expressible state tuples which possess semantic similarities, allowing us to express them in a similar fashion through the expression channel. The expressible classes suggested below were designed for expressive lights as our expression channel; for other expression channels, as noted above, the classification may vary, even for the same expressible state tuples.

Based on the expressible state tuples listed in the previous subsection, we propose the following expressible classes:

- Class **'Progressing through a process with known goal'**: There are different kinds of processes that the robot goes through in which the goal is known. The progress on

such processes could be expressed in the same way across different kinds of processes such as when the robot is escorting a person to a specific location or when it is charging its battery towards the maximum charge level. For both of the escorting and charging cases, the additional variable in the expressible state tuple represents a percentage completion.

The expressible state tuples corresponding to this expressible class are:

```
< 'Escorting', percentDone(loc,path-plan) >
< 'Charging', batt-level >
```

- Class **'Interrupted during task execution'**: There are different ways in which the robot's task execution gets interrupted. From the perspective of expressive lights, it doesn't matter knowing why the interruption occurred (e.g., because of an obstacle, a faulty robot component, or a user-initiated interruption through the GUI) as much as communicating a state of interruption. (Other expression channels such as screen display or voice could eventually complement the expression with additional information.)

The expressible state tuples corresponding to this expressible class are:

```
< 'Blocked by an obstacle', block-time >
< 'Interrupted by user', interrupt-time >
< 'Stopped because of faulty component', fault-level >
```

- Class **'Waiting for human input'**: As described previously, there are several situations for which the robot is waiting for some sort of human input (corresponding to tasks of type 'wait' or 'ask'): the robot can be waiting for a task to be initiated by a user or confirmed at completion time, or it can be waiting for help in cases where it uses symbiotic autonomy to overcome some of its limitations.

The expressible state tuples corresponding to this expressible class are:

```
< 'Asking for help at an elevator', None >
< 'Waiting for object loading', None >
< 'Waiting for dismissal', None >
```

4.4 Robot state / animation mapping

Now that we have discussed our representation of the robot state both on its own and in relation to an expression channel, we look at the problem of mapping expressible classes to specifically light animations, whose formalism was presented in chapter 3.

4.4.1 Mapping architecture

The mapping we define between expressible classes (state information) and animation tuples (light animation information) is divided into two parts:

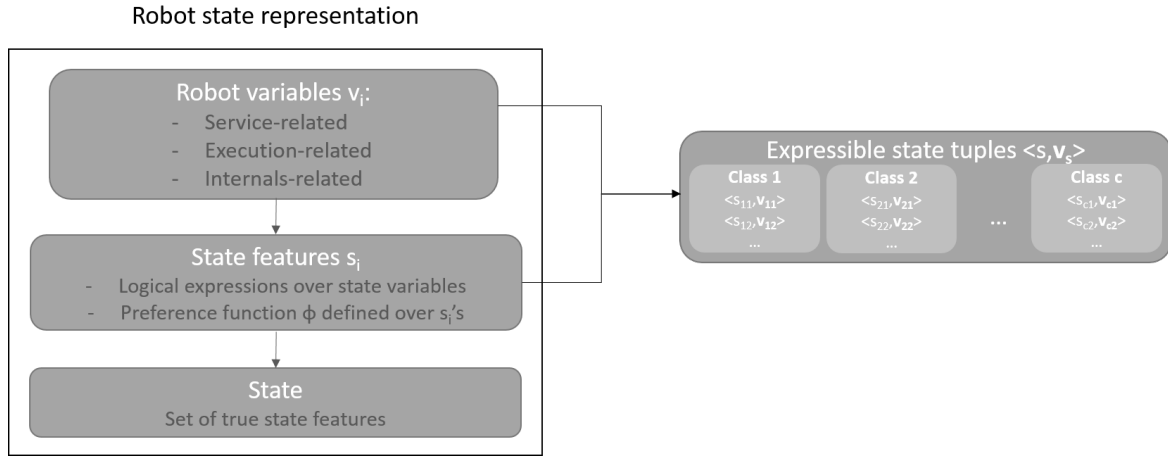


FIGURE 4.4 Summary of the introduced robot state concepts

- **Animation determination:** The signal shape sh as well the default parameters \mathbf{p}_{sh} and j_i ($i = \text{start, end}$) are determined by the 'feature' part of expressible state tuple, which is by definition discrete and communicates well the presence of distinct robot states.
- **Animation modulation:** We allow modulation in the animation by modifying the value of the parameters (for a fixed signal shape) based on the value(s) of the 'variable' part of the expressible state tuple.

Figure 4.4 summarizes the mapping architecture, and Figure 4.6 shows the flow of the mapping process for robot state all the way to the light animation module.

4.4.2 Expression of non-exclusive state features

As can be observed in the way state features are defined, there is no constraint on their mutual exclusivity, which means that there could be two or more features which are true at the same time, and hence more than one expressible tuple competing to be expressed on the same channel. If we assume that only a single expressible state tuple can be expressed on a single channel then we need a way of selecting one out of the set of expressible state tuples, which we call state preference function.

Definition 4.7. A *state preference function* for a given expression channel is a function $\phi : P(F) \rightarrow F$ where F is the set of state features and $P(F)$ is the power set of F . ϕ selects one preferred state feature given a set of true state features.

In practice, there might be sets of mutually exclusive features, which can reduce the domain of ϕ . Also, it might be possible to create a strict total order on the state features, which greatly simplifies the representation of ϕ , as is shown in the example below, but it might generally not be the case.

Example of preference ordering on sample features:

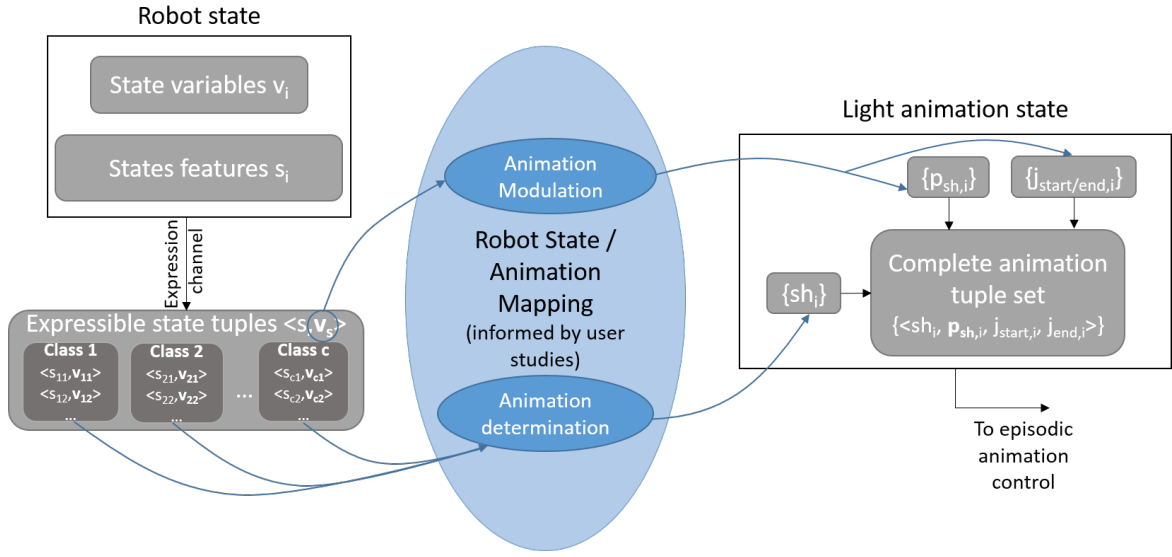


FIGURE 4.5 Architecture of the robot state / animation mapping

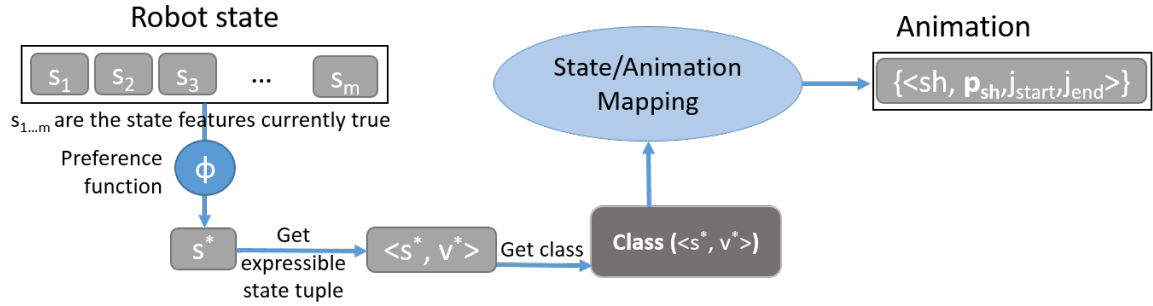


FIGURE 4.6 Flow diagram of the robot state / animation mapping

Consider the three features ‘Escorting’ = f_1 , ‘Blocked by an obstacle’ = f_2 , and ‘Asking for help at an elevator’ = f_3 .

If the robot is blocked by an obstacle (f_2 is true), then it should express it even if it currently escorting a person (f_1 is true). (f_2 and f_1 are mutually exclusive since the robot cannot be blocked by an obstacle while it is statically waiting at an elevator).

Similarly, if the robot is escorting a person (f_1 is true) but across multiple floors, it will encounter situations where it is asking for help at an elevator (f_3 is true) while performing the escort task. In such situation, we prefer f_3 over f_1 since the service cannot continue if the ‘ask’ task is not completed, which is hence more important.

Therefore we have the following total order: $f_2 > f_3 > f_1$, where $x > y$ indicates that x is preferred over y .

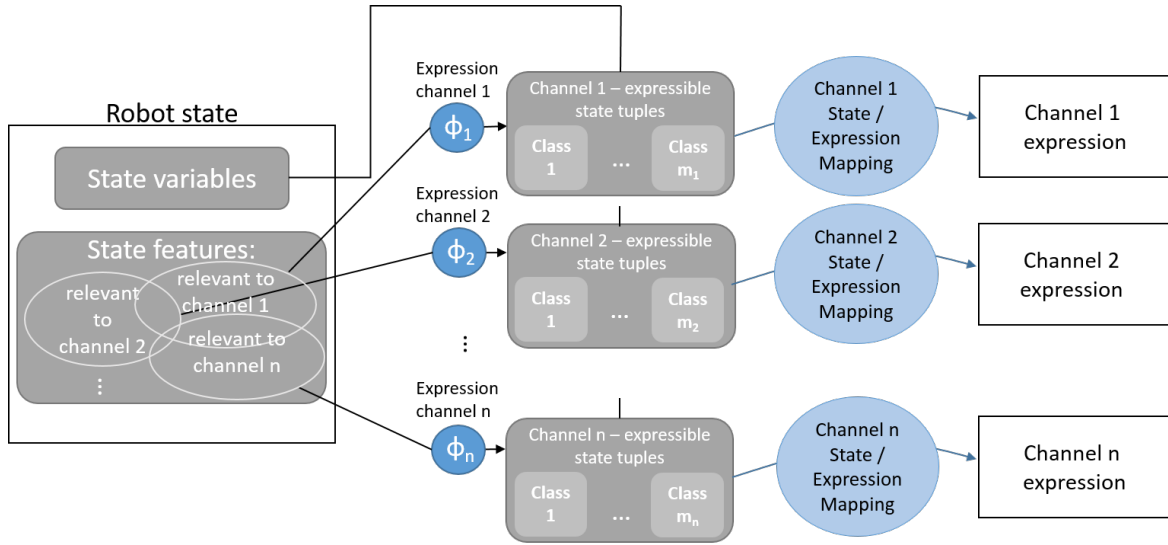


FIGURE 4.7 Mapping architecture for multiple expression channels

4.5 Extension to multiple light strips / expression channels

The mapping method introduced above for a single channel can easily be extended to handle multiple expression channels. Even though our analysis focuses on two channels of the same modality (expressive lights), the architecture presented is general enough to be applied to other modalities such as speech, expressive motion, etc.

Mapping architecture

Figure 4.7 shows our mapping architecture when more than one light strip, or more generally more than one expression channel, is present. Each channel has its relevant state features (possibly repeated), preference function, expressible state tuples and expressible classes.

Example using two light strips for multi-level expression

To demonstrate the extensibility of our concept to more than one expression channel, we consider one light strip for higher-level state information (related to tasks and services) expression and another light strip for lower-level state information (related to navigation parameters). Implementation details on hardware mounting and animation control can be found in the next section. The two strips can be seen in Figure 4.8. The high level strip animations will be informed by user studies in the next chapter. For the low-level strip, we considered a turn indicator animation, which lights either the left or the right part of the strip depending on the turn the robot is about to take. The rest of the strip changes color as a function of speed (red for fast, orange for medium speed and green for low speed). The corresponding expressible tuple for this strip's expression is: $\langle \text{'Turning'}, (\text{next-turn}(\text{loc}, \text{path-plan}), \text{speed}) \rangle$.

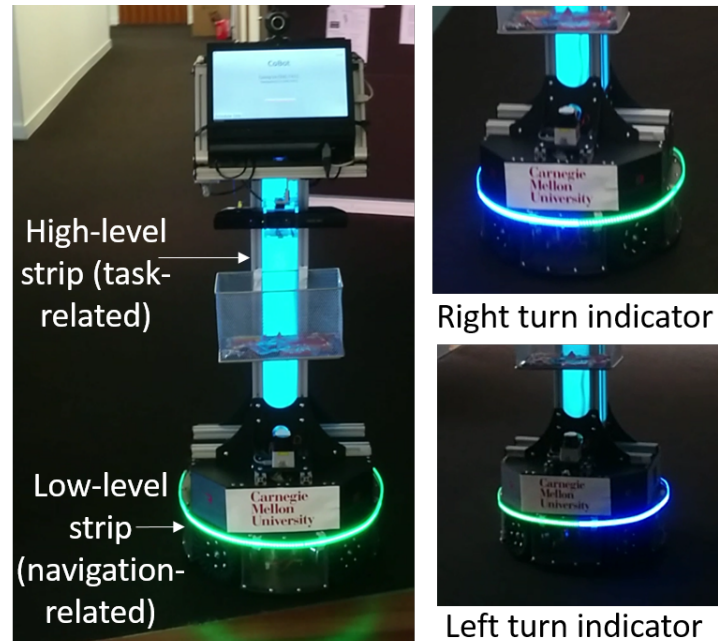


FIGURE 4.8 Two expressive light channels for multi-level expression on CoBot (left) and turn indicator snapshots for the low-level strip

4.6 Implementation of the mapping on a real robot

The framework discussed above for mapping robot state to light animations has been implemented on one of our CoBots and has been robustly running for more than a year whenever the robot is deployed. In this section we provide some details about the hardware and software components used to robustly integrate the light expression on CoBot.

4.6.1 Hardware components

For our light sources, we used two programmable, fully addressable NeoPixel LED strips¹ with 91 and 144 pixels respectively mounted on the robot's body and around its base respectively, as can be seen in Figure 4.9. Acrylic diffusers were added around the body LED strip to achieve omnidirectional visibility. Compared to other options like luminous fabrics or LED panels, linear strips are both simple in structure and flexible to adopt different mounting alternatives on CoBot. The NeoPixels strip moreover provides high light intensity thanks to its density of 144 LEDs/m (35 Watts/m max) which makes it suited for good visibility in luminous areas such as indoor bridges or other areas with glass windows.

The light strips are controlled by Arduino Uno microcontrollers² (one per strip). The data pin of the strips are connected to a digital output pin of the corresponding Arduino. The

¹<https://www.adafruit.com/products/1507>

²<http://store-usa.arduino.cc/products/a000066>

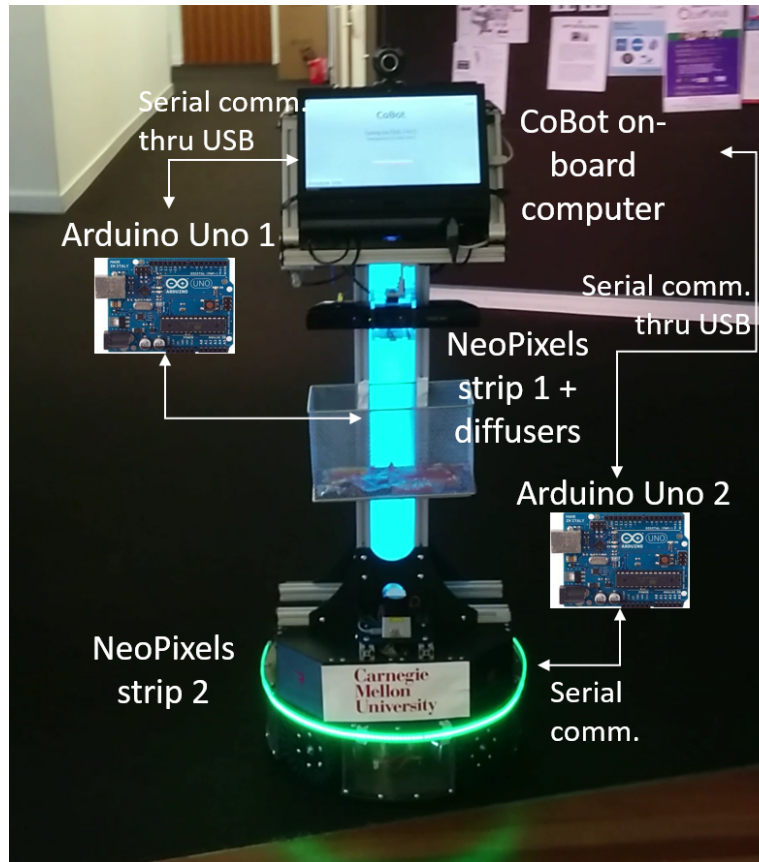


FIGURE 4.9 Hardware interface design

Arduino in turn is connected to the CoBot on-board computer through a USB cable used for serial communication.

4.6.2 Control architecture

The light interface control architecture is summarized in Figure 4.10. A Robot Operating System (ROS) node running on the robot itself keeps track of the robot variables (by subscribing to the corresponding ROS topics or calling the corresponding ROS services) and computes the corresponding expressible state tuples. It then maps these to animation tuples for each strip, which are sent to the corresponding microcontroller using serial communication. The protocol used for communication between the ROS node and the microcontroller can be found in Appendix A. Based on the animation tuples received, the microcontroller controls the light strip by running the episodic animation algorithm described in chapter 3. The flow of data from the ROS node to the microcontroller is synchronized with the animation episodes because serial communication is only reliable when the microcontroller is not sending any data out to the strip. We ensure such synchronization by a simple procedure similar to a handshake at the end of each animation episode. The actual data sent out to the LED strip on

the digital pin of the Arduino is determined by the Adafruit NeoPixel library³ and uses serial packets that the WS2811-based pixels understand.

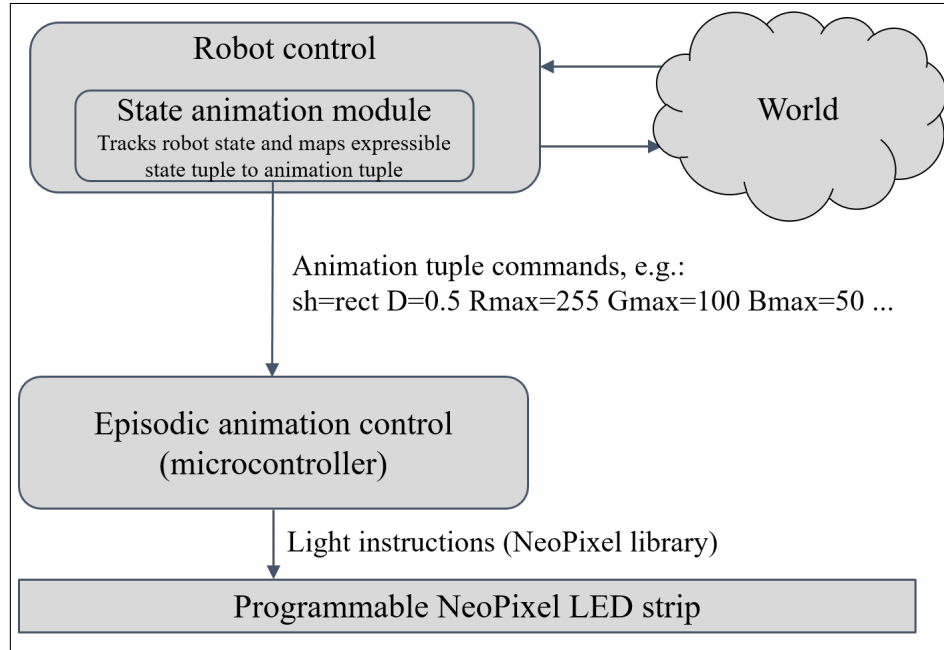


FIGURE 4.10 Control diagram of the state animation interface

4.7 Chapter summary

In this chapter, we focused on extracting *informative* elements of a robot's state to be expressed through an expression channel (in our case, through light animations). We presented a general representation of a mobile service robot's state that is suited for expression. We then showed how to map the extracted elements of the robot's state to an expression, in particular a light animation as defined in the previous chapter. We extended our method to handle more than one expression channel, which is illustrated using two light strips on CoBot expressing different types of state information (high-level task-related versus low-level navigation-related information). Our process, although it has focused on CoBot, is general enough to apply to other platforms, and other expression channels than lights.

³https://github.com/adafruit/Adafruit_NeoPixel

Chapter 5

Design and evaluation of the state/animation mapping

The previous chapter was concerned with question of *what* to express, i.e., selecting appropriate state information that is useful and relevant for a given situation. In this chapter, we focus on the question of *how* to express the selected state information. In order to answer this question, we present three studies. The first study [3] is concerned with the design of appropriate animations for some of the expressible state tuples discussed in chapter 4, i.e. selecting the appropriate animation parameters discussed in chapter 3 according to the different scenarios considered. The second study [4] evaluates the *legibility* of the animations resulting from our design study, as well as their generalizability to expression of state tuples in the same class (refer to chapter 4). The third study is a small experiment which proves that the presence of these animated lights on the robot can actually influence people's behavior to help the robot perform better at its tasks.

5.1 User study 1: Designing appropriate animations

5.1.1 Methodology

In order to select suitable parameters for the animations presented above, we conducted a study with a video-based survey. Participants were first given detailed description about the situation of the robot in each scenario and then asked to watch videos showing the robot in each of the scenarios defined above, while answering a survey through the form of a spreadsheet.

5.1.2 Preliminary Study

A preliminary study was conducted with the people who have the most expertise for our purposes, namely the CoBot developers. Eight developers participated in the survey, and submitted their choices. To validate our design choices, we recruited 30 more people to

include in the study. The results across both studies were consistent. The extended study is described next.

5.1.3 Participants

A total of 38 participants took part in this study. 61% study or work in a robotics-related field, 18% are in a design-related field and 21% are in an engineering-related field. Ages range from 19 to 50 years with an average of around 25 years. 18% are from North America, 32% are from Europe, 29% are from the Middle East and 21% are from Asia. 68% are male and 32% are female.

5.1.4 Survey design

Participants were asked to give their input on three aspects of the animation: animation pattern, speed and color. For each scenario, 3 different animation patterns (corresponding to signal shapes + dynamics parameters) were shown with the same neutral color (soft blue). Nuances of 3 different speeds were also shown within each pattern. The participants were asked to select the one that they thought would fit best the robot's expression purposes in the given scenario. Participants were also shown 6 possible light colors (in the form of a static image of the robot) and were asked to select the most appropriate for each scenario. We make the reasonable assumption that the choice of color for the animation is independent of the actual animation selected, which helps reduce the amount of choices to be shown. Indeed, while animation pattern and speed both relate to modulations in time and intensity, color seems to be much less intertwined to the other two. Furthermore, according to color theory [50], color on its own plays a strong role in expression. Next, we list and justify the choices of animation patterns shown to the participants.

- Scenario "waiting": A regular blinking animation (Blink); a siren-like pattern; a rhythmic (non-regular) blinking animation. We believe these to be good candidates for grabbing attention because of the dynamic aspect, the warning connotation and the non-regular pattern respectively.
- Scenario "blocked": A faded animation (that we call "Push") that turns on quickly and dies out slower (giving the impression of successively pushing against an obstacle); an "aggressive" blink (fast blink followed by slow blink); a simple color change at the time the robot gets blocked. We believe these to be good candidates for inciting the human to move away from the path.
- Scenario "progress": A bottom-up progress bar where lights gradually fill from top to bottom proportionally to the distance from the goal; a top-down progress bar where lights fill from the top towards the bottom; a gradual change from an initial color to a final color, again proportionally to the distance from goal.

The parameter values associated with these animations are summarized in Table 5.1. In addition to the animation summarized in the table, the following colors were shown for each scenario as static images of the lighted robot: Red (R), Orange (O), Green (G), Soft Blue (B), Dark Blue (B') and Purple (P).

TABLE 5.1 Parameter values for the animation choices shown

Scenario "Waiting"			Scenario "Blocked"			Scenario "Progress"	
wv	D	T (s)	wv	D	T (s)	$disp$	u_{disp}
Blink			Push			prog_bar	bottom_up
	0.5	2/1.6/0.6		0.25	1.5/1/0.5	prog_bar	top_down
Siren			Aggressive Blink			color_change	-
	0.5	2/1.6/0.6		0.5	2/1.6/0.6		
Rhythmic Blink			Color change				
	0.5	3/2.5/1.5		1	-		

5.1.5 Results and discussion

Table 5.3 shows the selected best choices, which were consistent between the preliminary and the extended study. Fig. 5.1 and Table 5.3 show the distribution of the results in the extended study. In the following discussion, p-values are obtained from a Chi-Square goodness-of-fit test against a uniform distribution.

In Fig. 5.1, we show the results for the animation pattern. For the scenario "waiting" ($p = 0.0137$), among the participants who chose the winning animation "Siren", 64% chose the slower speed, 29% the medium speed and 7% the faster speed. For the scenario "blocked" ($p = 0.0916$), among the participants who chose the winning animation "Push", 27% chose the slower speed, 40% the medium speed and 33% the faster speed. Note that the static color change was the least preferred animation pattern for this scenario, which aligns with Bertin's result stating that motion (in our case a non-static animation) being one of the most effective visual features for attention grabbing [7] (for the "waiting" scenario which relies even more on attention grabbing, all three of our designed patterns were designed to be in motion). For the scenario "progress" ($p = 1.10^{-6}$), the participants chose the bottom-up progress bar animation. All p-values obtained are below 0.10, which indicates a strongly non-uniform distribution of preferences for each scenario, and this can clearly be seen in Fig. 5.1.

The results for colors, summarized in Table 5.2 similarly show a clear preference for one option in each case. For instance, soft blue was selected for the "waiting" scenario. This result supports the statement in [13] that cold colors are better than warm colors at grabbing attention. Also, red was selected as the best color for the "blocked" scenario. This is consistent with the fact that red is often perceived as demanding [50] or stimulating [13], which are both desirable in this scenario. Even though the Red/Green color combination was the most voted for in the "waiting" scenario, this study did not account for interference

between animations for different scenario. As it turns out from real-world deployments, since the color red was associated with the “blocked” scenario, it confused people to also see it in the “progress” scenario. We hence replaced the background color with a faint dark blue, but still against a bright green progress bar.

The results of the study show that some animation design alternatives can be eliminated, while a small set can be considered valid. Although there is generally a clear preference for one of the choices in each scenario, shown in Figure 5.2, the study was informative of the distribution of preferences, which enables us to possibly generate animations probabilistically instead of only committing to a single one. Also, the scenarios we looked at are quite generic and are commonly encountered in interactions involving a social robot and a human. However, before extrapolating our results to other platforms, we need to ensure that other factors (e.g. strip size or placement, light diffusion mechanism ...) do not influence the perception of the expression. These results can still however serve as a starting point for the design of future social robotic systems which use lights as a means of communication.

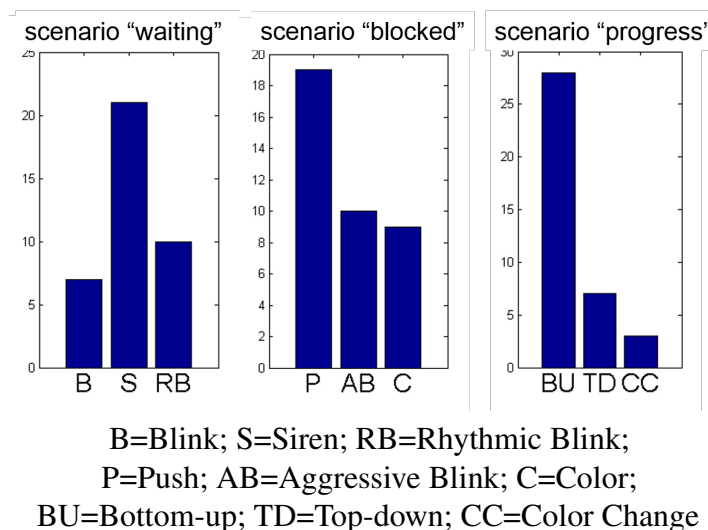
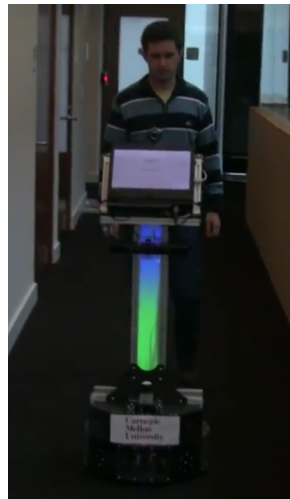


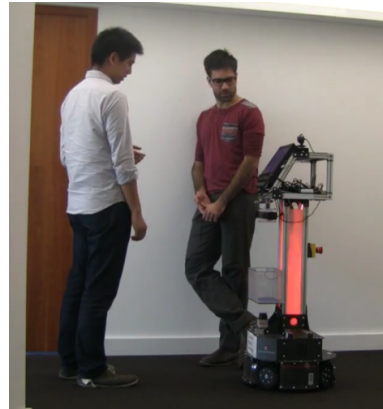
FIGURE 5.1 Animation pattern results

5.2 User study 2: Evaluating and generalizing the designed animations

In order to evaluate the effectiveness of the chosen expressive light animations, we conducted an online survey in which participants watched videos of a robot performing tasks from afar. At the end of each video, participants were asked to hypothesize about the robot’s current *state*, but also about its *actions* (specifically reasons for performing a specific action such as stopping or being unresponsive). Questions were in a multiple choice format, with four possible answers. Half of the participants saw the robot performing tasks with its expressive lights on (referred to as the “Lights on” condition), and half saw the robot with the lights off



(A) Green ascending progress bar
on an escort task



(B) Flashing red “push” animation
for path obstructions





(C) Slow soft blue “siren” to call
for human help

FIGURE 5.2 Snapshots of the winning animations for each scenario of user study 1, also used in the subsequent user studies

TABLE 5.2 Color results (color codes correspond to those used in the text)

Scenario "waiting"					
R	O	G	B	B'	P
13%	13%	13%	39%	16%	6%
Scenario "blocked"					
R	O	G	B	B'	P
53%	29%	5%	0%	10%	3%
Scenario "progress" (top 6)					
R/G	B/P	B'/G	O/G	O/B	P/B
27%	12%	12%	8%	8%	8%

TABLE 5.3 Selected best animations for each scenario of user study 1

Scenario	Animation and parameters			
"waiting"	Soft blue "Siren" with period 2s			
	wv	D	T (s)	Color
		0.5	2	Soft Blue
"blocked"	Red "Push" with period 1.5s			
	wv	D	T (s)	Color
		0.25	1.5	Red
"progress"	Bottom-up progress bar			
	$disp$	u_{disp}	In. Color	Fin. Color
	prog_bar	bottom_up	Red	Green

(referred to as the “Lights off” condition). Participants were randomly assigned to one of the two experimental conditions. We analyzed participants’ hypothesis choices to demonstrate that those who saw the robot with the lights were more accurate and gained a higher level of trust in robots from watching the videos.

5.2.1 Participants

A total of 42 participants, of which 14 were male and 28 were female, took part in the study. Ages ranged from 20 to 67 ($M = 32.4$, $SD = 13.7$). Out of the 42 participants, 33 live in the United States; the rest live in different countries across Asia and Europe. Even though computer usage was relatively high amongst participants (31 out of 42 used computers 30+ hours per week), experience with robots was generally low. Only 5 out of the 42 participants reported having worked with robots before, and 20 reported that they have never seen a robot in person before (3 participants had seen our particular robot, CoBot, before taking the survey). Finally, we ensured that none of the participants were colorblind, since our light animations included color and it could have an effect on our results.

5.2.2 Survey Design

Our online video-based survey comprised nine video scenarios of CoBot acting in our environment followed by a multiple choice question asking participants to choose a hypothesis of what the robot was doing. Four plausible hypotheses about the robot's state/actions were presented as choices for each video, of which one had to be selected. Participants were also asked to rate their confidence in their answer on a 5-point Likert scale. The video order, as well as the choices for each answer, were randomized to avoid any order effects.

Each of the video scenarios was recorded using our autonomous robot with lights on and lights off. Although the robot was acting autonomously, the videos were replicated as close as possible for the two conditions. We can reasonably assume that the only notable difference between the two videos for a given scenario is the presence or absence of lights on the robot. The videos did not include any robot speech or any visible information on the robot's screen.

After viewing all nine videos, some relevant background and related information, including trust questions about this particular robot and robots in general, was also collected. Additionally, we recorded the time taken for completing the survey and made sure everyone responded within reasonable time limits (no disqualifications).

5.2.3 Scenario descriptions

The nine scenarios shown in the videos were specifically chosen based on actual tasks that the robot performs while it is deployed in our buildings. We focused our scenarios on the same three common *scenario classes* studied in our prior work – “progressing through a process”, “blocked”, and “waiting for human input”. For each scenario class, we produced three distinct scenarios in which the robot's state or actions are ambiguous, which are summarized in Table 5.4 and described below.

The “**progressing through a process**” scenarios represent the robot taking actions for a long duration. For each of these scenarios, the progression was modeled as the light expression of a progress bar (see subsection 3.1). The scenarios chosen to represent this class are:

- *Navigation task with human presence (P1)*: A person participates in the Escort Task in which they are accompanied to their goal location.
- *Speech task (P2)*: The person asks a question to the robot, which provides no immediate answer, as it is searching the web for the required information. The video ends before the robot responds. When present, the lights show the progress on the web query task.
- *Charging (P3)*: The robot is charging inside the laboratory, with no clear view of the power plug. When present, the lights show the battery level increasing progressively (video sped up 10 times).

The “**blocked**” scenarios represent the robot being interrupted in its navigation by obstacles of different kinds. The important blockage is supported by the fast red flashing light (see subsection 3.1). The scenarios chosen to represent this class are:

- *Human obstacle facing the robot (B1)*: The robot is stopped in its navigation by a person standing in a narrow corridor, facing the robot.
- *Human obstacles looking away from the robot (B2)*: The robot is stopped in its navigation by a person standing in a narrow corridor, facing away from the robot.
- *Non-human obstacle (B3)*: The robot, navigating down a narrow corridor, detects a person walking towards it and changes its navigation path to avoid the person. As a result, it finds itself in front of a branch of plant, which it considers as an obstacle, causing it to stop.

The “**waiting for human input**” scenarios represent the robot stopped waiting for different types of actions to be taken by a human. For each of these scenarios, the robot is waiting patiently as represented by the slow flashing blue light (see subsection 3.1). The scenarios chosen to represent this class are:

- *Waiting for help at an elevator (W1)*: The robot is stopped in front of the elevator, waiting for someone to press the elevator button and let it in. People are passing by, ignoring the robot’s presence.
- *Object loading (W2)*: The robot is stopped in the kitchen area, facing a counter on which we can see a cup of coffee. Next to the counter area, a person is washing the dishes, presumably unaware of the robot’s presence.
- *Confirming task completion (W3)*: The robot is stopped in front of an office door, with coffee in its basket. A person shows up from inside the office and takes the coffee. The robot doesn’t react to the person’s action and remains still. The person looks at the robot with a confused look on their face.

For each scenario, when lights are present, the default animation on the robot (when no expression is desired) is a static soft blue color.

5.2.4 Multiple choice questions

After viewing each video, the participants were given choices to explain the robot’s state or actions. As discussed earlier, each of the scenarios can be ambiguous to a person viewing CoBot from afar either because of lack of contextual information or because of mixed signals in the robot’s behavior. The corresponding answer choices for each video scenario were specifically chosen to reflect many of the possible hypotheses that could correspond to the robot’s behaviors. Given our prior work, we theorize that the light expressions will reduce the uncertainty that people have in understanding robot’s behavior, leading to more accurate answers to our multiple choice questions.

Question examples Some examples of questions and choices in the survey are:

In the video above, why did the robot stop? (a) The robot recognizes the person, who was expecting it, (b) The robot sees the person as an obstacle, (c) The robot needs help from the person, (d) The robot is inviting the person to use its services. (Scenario B1)

TABLE 5.4 Scenarios used in user study 2

Scenario class	Progress to a goal (P)	Blocked (B)	Waiting for human input (W)
Scenario 1	Navigation task with human presence (P1)	Human obstacle facing the robot (B1)	Symbiotic autonomy (elevator button) (W1)
Scenario 2	Speech task (P2)	Human obstacles looking away from the robot (B2)	Object loading (W2)
Scenario 3	Battery charging (P3)	Non-human obstacle (B3)	Confirming task completion (W3)

In the video above, why is the robot not moving after the person has taken the coffee? (a) It is waiting for the person to confirm the task is over, (b) It has nothing to do, (c) It is low on battery, (d) It is trying to get inside the room but the door is too narrow. (scenario W2)

5.2.5 Results

Responses to the survey multiple choice questions in the nine scenarios were coded in a binary fashion – three answers were coded as wrong and one answer was coded as the correct answer. The resulting dependent variable *accuracy* was modeled as binary categorical. Additionally, we coded the responses to our questions about robot *trust* (5-point Likert scale). We analyzed the effects of our independent variables – experimental *condition* (binary categorical variable “Lights on” and “Lights off”) and *scenario* (nine categories) – on the dependent variables. While our scenarios had a range of difficulty resulting in a range of accuracies, our light animations have a statistically significant effect across all scenarios on participant’s accuracy. The participants who saw the robots with lights on also indicated an increase in their overall trust in robots more than those who saw the robot with lights off.

Participant accuracy

In order to analyze our categorical dependent variable *accuracy*, we used a McNemar’s chi-square test in a combined between- and within-subject design. The “Lights on/off” *condition* is our between-subject variable. All nine video *scenarios* were shown to all participants and therefore is a within-subject variable. The *participant* is modeled as a random variable within the model as each person may be more or less accurate in general. The McNemar’s chi-square tested whether the participants’ answers depend on the presence/absence of lights, video scenario, and/or the interaction effects of both the lights and video scenario together.

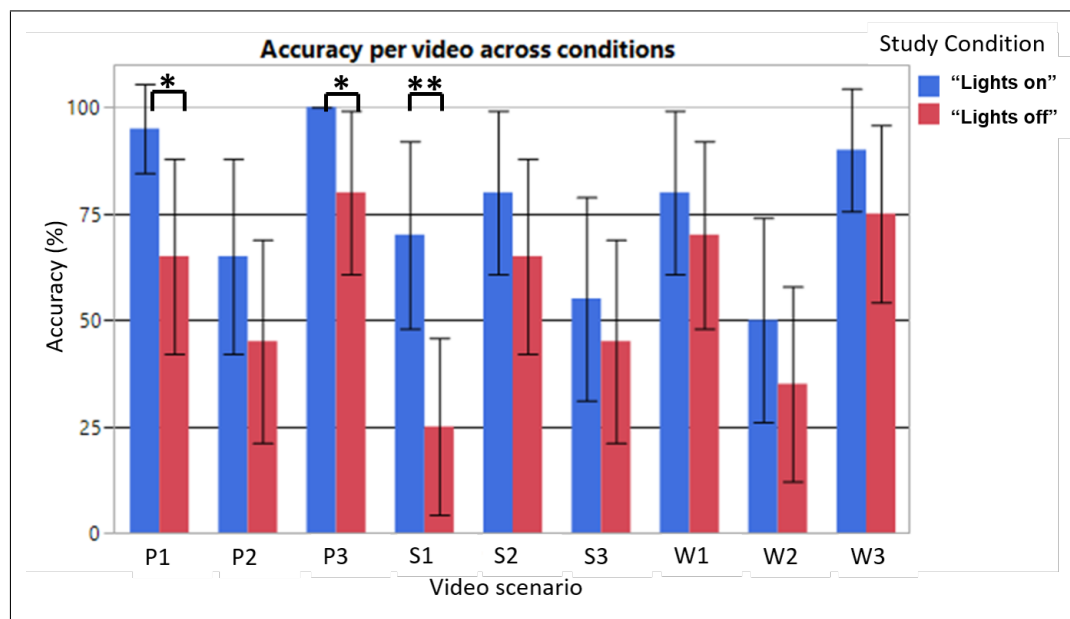


FIGURE 5.3 Comparison of the accuracies on each scenario across the two study conditions (error bars constructed using a 95% confidence interval; statistical significance shown used t-test)

Our results indicate that there is a statistically significant difference in the accuracy based on the presence/absence of lights (“Lights on” $M = 75.66\%$, $SD = 18.20\%$; “Lights off” $M = 56.08\%$, $SD = 19.16\%$, $p < 0.0007$). The accuracy was significantly higher for participants who saw the lights. Additionally, there is a statistically significant difference in participants’ accuracy based on the video scenario (see Figure 5.3 for means and standard deviations, $p < 0.0001$) (i.e., some videos were harder to determine the robot’s state/actions than others for each participant). However, there was no statistically significant effect by the interaction of the light condition and the video scenario ($p = 0.7$), indicating that the increased effectiveness of the “Lights on” condition was the same across scenarios. Based on these results, we conclude that while the choice of a correct robot state/actions hypothesis does depend on the scenario in which humans see the robot, the tested light animations universally help increase their accuracy.

Figure 5.3 shows the average accuracy of the participants for each scenario and each light condition. The error bars represent a 95% confidence interval of the mean. We note that the “Lights on” condition accuracies (shown in blue) are universally higher than the “Lights off” accuracies (shown in red). Additionally, the graph clearly shows our result that the video scenarios have different average accuracy, but the accuracy change between conditions per video scenario is not reflective of the scenario.

Participant trust in robots

On average, participants reported that their trust in robots had increased after watching the videos shown in the survey. (To the question: “Do you agree with the following statement? ‘After watching these videos, I will not trust robots as much as I did before.’”, participants

in both conditions answered above 3 over 5 on average on a 5-point Likert scale, where 1 meant “Strongly Agree” and 5 meant “Strongly Disagree”.) The reported increase in their trust in robots was significantly more pronounced for participants in the “Lights on” condition ($M = 4.29$, $SD = 0.90$) compared to those in the “Lights off” condition ($M = 3.52$, $SD = 0.87$) ($t(40) = \pm 2.02$ two-tailed, $p = 0.008$).

However, there was no statistically significant difference between the two conditions in the reported absolute level of trust in both CoBot and in robots in general ($t(40) = \pm 2.02$ two-tailed, $p > 0.05$), only in the change in trust discussed above did the results differ across conditions.

Participant confidence

we analyzed the average reported answer confidence over the 9 videos and found that it was slightly higher in the “Lights on” condition as compared to the “Lights off” condition (2.9 versus 3.17 respectively on a 5-point Likert scale). However, this difference was not statistically significant ($t(40) = \pm 2.02$ two-tailed, $p > 0.05$). Similarly, no statistical significance was found when comparing reported confidence on a per video basis.

5.2.6 Discussion

The significant effect of scenario on response accuracy shows that some questions were harder than others. This was not unexpected, since some scenarios were more ambiguous than others. The four answer choices presented for each question were designed by us and it was inevitable that some choices were more obvious to choose or to eliminate depending on the scenario and the question asked.

Furthermore, it is to be noted that all of our videos intentionally lacked obvious contextual clues. Lack of such clues is a usual situation when encountering a mobile robot like CoBot. Visitors often encounter the robot for the first time and interact with it with no knowledge at all about its capabilities, current state, or expectations from humans. Another example is when CoBot is stopped waiting for help (e.g., at an elevator). In such cases, looking at the robot from afar does not give much insight about what the robot’s state, unless other visual cues are present.

Moreover, our three animations, originally designed for a specific scenario, are shown to generalize well to other similar scenarios. Some scenarios (like P3 and W3) even outperform the scenario used for the design of the light expressions. We attribute the success of such generalizability to the abstraction used in these expressions.

Finally, the fact that no significant changes in participants confidence was observed between the two conditions might be due to the fact that participants in one condition were unaware of the other condition which might have made all confidence reports average out around the mean score. It might be interesting to see if these results change in a within-subject study of the same type. The confidence people have on hypothesizing about a robot’s state and actions is in fact an important factor to have in mind when designing robotic systems, especially safety-critical ones (such as for example self-driving cars).

5.3 Experiment: Impact on human behavior

In order to evaluate the impact of the robot body lights on people's behavior with respect to the robot, we conducted a small observational study to evaluate whether the presence of lights influenced people to help CoBot at an elevator by pressing the elevator button.

5.3.1 Experimental procedure

CoBot was placed in front of the elevator with the following text on its screen: "Can you please press the elevator button for me to go up?". We had two experimental conditions:

- Condition 'Lights on': in this condition, the lights were on and displaying our designed animation for the 'elevator waiting' scenario.
- Condition 'Lights off': in this condition, the lights were off and the only indication that the robot needed help pressing the elevator button was the on-screen text.

The experiment was run over three different floors of our building (labeled GHC6, GHC7, and GHC8), which had slightly different elevator area architectures and different people. Both conditions were run over each floor, with the order of the conditions randomized and counterbalanced across floors.

Our dependent variable is whether or not the person provides help to the robot. It is coded as a binary variable 'help', where '1' means help was provided and '0' means no help was provided.

5.3.2 Data filtering

Because the human dynamics around elevator areas can be complex, we had to filter our data according to some criteria, listed below:

- We filtered out people who passed across the robot from the back, since it would be hard for them to see what the screen said.
- If there was at least one person waiting at the elevator, any behavior from the passing people was discarded since the presence of people at the elevator might have interfered with people's willingness to help the robot (e.g., diffusion of responsibility effects).
- People who were not only passing across the elevator but planned to get on it were filtered out, unless they purposefully pressed the button in the reverse direction they were going to help the robot (i.e., pressing the 'up' button for the robot and the 'down' button for themselves).
- If the 'up' elevator button was already pressed and people showed a clear intent of pressing it again without actually pressing it, they were counted in.

Floor	Lights on	Lights off
GHC7	7/10	1/10
GHC6	4/10	3/10
GHC8	6/10	3/10
Total (%)	56.7%	23.3%

TABLE 5.5 Proportion of people who helped the robot in the two experimental conditions across the three floors

5.3.3 Results and discussion

Table 5.5 shows the results for the two floors. We observe that the presence of lights has a statistically significant impact on humans helping CoBot ($t(58) = \pm 2.00$ two-tailed, $p = 0.007$). We attribute this difference to two factors: first, the fact that the animation for this scenario is legible as shown in our legibility study; second, the fact that the animation was effective at grabbing people's *visual attention* in addition to the other contextual clues that the robot is in need of help such as the on-screen text and the position and orientation of the robot close to the elevator. While only 23.3% of people provided help to CoBot with lights off, 56.7% provided help when the lights were on. The effect was more or less pronounced according to the floor, which we may attribute to different factors such as familiarity with the robot, architectural differences between the floors that might affect visibility, and time of the day.

5.4 Chapter summary

In this chapter, we have presented three user studies which evaluate the behavior of CoBot's body light strip.

The first study informed our design of light animations for three different scenarios (waiting at an elevator, blocked by a human obstacle, and escorting a person). We asked experts to provide their top choice from a 'catalog' of available light animations, speeds, and colors.

The second study validated the legibility of the designed animations, as well as their generalizability to scenarios similar to the ones used in the design process. It mainly showed that the presence of lights has a significant impact on people's understanding of the robot in diverse situations.

The third study was a small real-world experiment which proved that these lights can have a real impact on people's behavior with regard to the robot. It showed that people were more likely to help the robot press the elevator button when the lights were displaying our 'waiting for human help' animation.

Chapter 6

Introducing agency in expression: personalization and adaptation in persistent interactions

So far we have focused on using expressive lights as a way to reveal a robot state. By doing so, used these lights for a functional purpose pertaining to providing humans with a better understanding of the robot during task execution. We have shown how these lights can be used in a way that is both *informative* and *legible*. We now look at a third aspect of these lights, *engagement*. To show how they can be engaging, we explore in this chapter another use of these lights whereby no intentional meaning is associated with them but where they are instead used as a complement for the main interaction with the user. More concretely, consider a mobile service robot visiting a person's office on a daily basis (say delivering an object, bringing coffee etc.), the question we are asking ourselves in this chapter is: how should the appearance of the robot (in our cases light animations on the robot) change along the time dimension over repeated interactions with the same user? Our main assumption is that different users will show different satisfaction patterns with a robot changing its appearance as a function of the number of times it has visited a particular user.

We make the following assumptions:

- (1) A robot repeatedly interacts with different humans whose identity is known by the robot.
- (2) Every time the robot encounters a given user, it chooses one out of a set of fixed possible actions, corresponding to light animations, to interact with them.
- (3) The user has a method of providing feedback for the interaction through a score or rating.

In this chapter, we present a method that enables an agent, such as a mobile service robot, to learn online action sequences which maintain high reward over time [5]. Our formulation of this problem, being general enough, is not specific to light animations, but can be applied to any other valid expressions or interactions with similar functionality, such as different speech with the same meaning, different motions with the same start and goal, etc.

In practice, social interactions can be much more complex and require much more context to help adaptation, however there is a wide range of interaction types which do not play a direct functional social role but act more as a complement to the main interaction. As mentioned earlier, we use interaction with lights on CoBot, but our method can be applied to other types of interaction modalities. A finite set of predefined light animations is used for personalized interaction. The robot, being able to accurately navigate and localize itself accurately in our buildings, can identify users (e.g. by their office number or by recognizing an associated digital signature), hence enabling the personalization of light animations while servicing that user. At the end of each interaction, the touch screen interface may be used to rate the interaction (e.g. by the use of a slider), providing the rewards on which our algorithm relies. Other possible reward collection mechanisms can include facial expression analysis or other sensors such as wearable sensors.

Long-term user preferences are however from being static or homogeneous, which is not accounted for in traditional recommender systems. Indeed, being exposed to the same type of interaction for a long time might develop boredom or fatigue for some, while others might value it for its predictability. To summarize, general static preferences change from individual to individual, but preference dynamics are equally important in a long-term setting. In this chapter, we propose to learn, for a given user, both sets of preferred actions and time-related quantities which would dictate the preferred dynamics of interaction.

This chapter is divided into three main parts. In the first part, we introduce three user models which capture different possible profiles in relation to the appreciation of “change” in the way the robot interacts with the user. These are formulated in terms of evolution of the reward from the user as a function of the possible sequences of actions used when interacting with that user. In the second part, we present our algorithms to learn the model parameters, assuming we are given the user model. In the third part, we show the applicability of this work to our mobile service robot, CoBot, which is deployed in our buildings and uses expressive lights for improved interaction with humans.

6.1 Background

In the problem we consider, we will assume that user data are limited to rewards at every interaction (in the form of a rating submitted by the user), making it comparable to a recommender system learning user preferences and suggesting new items [10]. However, the algorithms used in such systems do not take into account the dynamics of preferences (boredom, habituation, desire for change etc.). In the field of automatic music playlist generation, the concepts of diversity and serendipity have been mentioned [11]. However, no viable solution has yet been proposed to address this problem. Also, the idea of exploration in music recommender systems has been studied [48], but it does not apply to our problem since we assume the number of actions to be relatively small. In a robotics application, the need for adaptive interaction that takes into account habituation has been recently formulated for empathetic behavior [31] (in this chapter, we take a more general approach). Going back to the problem of preference dynamics, our problem can formally be compared to the restless multi-armed bandit problem where rewards are non-stationary and which is generally known

to be P-SPACE hard [32]. In this work, we restrict the rewards to evolve according to one of three models, which makes the problem of learning the model parameters easier to solve.

6.2 Formalism and user modeling

In this section, we start by presenting the formulation of the problem at hand and move on to introduce three models of dynamic preferences corresponding to three different possible profiles of users.

6.2.1 Problem setting

Time is discretized into steps $i = 1, 2, 3, \dots$, where each time step represents an encounter between the robot and the user. We assume that the encounters are of an identical nature or serve the same functional role (for example the robot is always delivering an object to the user's office). Also, for simplicity, we do not worry about the actual time interval between two consecutive steps (these could be for example different days or different times within the same day). At every time step, we assume the robot has to choose one out of a set of n possible actions corresponding to light animations. In the context of light animations, the different actions represent different animations in terms of speed, color or animation patterns. Let $\mathbf{A} = \{a_1, \dots, a_n\}$ represent the set of possible actions. After every encounter, we assume that the user provides a rating $r^{(i)}$ to the interaction where $r^{(i)} \in [0; 10]$. The reward is assumed to be corrupted by additive white Gaussian noise: $r = \bar{r} + \varepsilon$ where $\varepsilon \sim N(0, \sigma^2)$. The noise can come from the following sources: (1) inaccurate reporting of the user's true valuation, (2) mistakes when using the user interface (e.g. slider) to report the reward and (3) failure to remember previous ratings resulting in inconsistent ratings.

Our goal is to learn, for a specific user (with a specific reward model), which action to take next given the history of actions and rewards. The problem can hence be compared to the Multi-Armed Bandit problem where a single player, choosing at each time step one to play one out of several possible arms and gets a reward for it, aims to maximize total reward (or equivalently minimize total regret) [32]. In our case, the rewards are stochastic and non-stationary and the arms or actions, corresponding to the different light animations, are relatively few.

6.2.2 Modeling dynamic user preferences over time

Human preference dynamics have mainly been studied by market researchers including variety-seeking [30] [6], boredom and novelty seeking [20] consumer behaviors. In this section, we introduce three user models corresponding to different profiles of users interacting with a mobile service robot repeatedly. Our models are full mappings from action sequences to reward sequences for a particular user profile. Our three user profiles were chosen in such a way to span well enough the spectrum of possible profiles for users encountering a robot frequently (say on a daily basis). Our models are vaguely inspired by variations along the "openness" dimension of the five-factor model in psychology [19]. These models we crafted

take into account both properties of preferred actions sets and time-related quantities dictating the evolution of rewards depending on the action sequences. Figure 6.1 shows sample ideal sequences for lights on our robot, CoBot, for each of the three models on different days in which the robot visits a person's office to deliver coffee. For the three models presented below, we use \mathbf{A}_{pref} to denote the set of preferred actions (regardless of the sequence of actions in which they fall).

Model 1: The “conservative”

This type of user wants to stick to one option denoted by a^* , but appreciates surprises from time to time at some frequency. A surprise means taking for one time step an action $a \neq a^*$ in a set of preferred “surprise actions” $A_{\text{surp}} \subset \mathbf{A}$. When a^* is repetitively shown in a sequence (we call sequences of the same action homogeneous sequences), the reward \bar{r} starts out as a constant (r_{max}) and after T time steps starts decreasing, due to boredom, with a linear decay rate α until it reaches r_{min} , after which it remains constant. For homogeneous sequences of the non-preferred actions (i.e. actions in $A \setminus \{a^*\}$), the reward starts at a value $r_{\text{non-pref}}$ and decreases exponentially to zero with time (indicating that the user very quickly gets bored) with some decay rate β . In summary, the model parameters are:

- a^* : the action with the maximum value of $E[\bar{r}]$. A homogeneous sequence of a^* actions is referred to from now on as a **p-sequence**.
- $\mathbf{A}_{\text{pref}} = \{a^*\}$
- \mathbf{A}_{surp} : set of actions suitable for surprises, defined as $\{a : E[\bar{r}_a] > r_{\text{th}}\}$, where r_{th} is a threshold value.
- T : optimal length of the homogeneous sequence of the preferred action, after which the user starts getting bored. If the robot always alternates between p-sequences and surprises, T can also be seen as a between two consecutive surprises. T is assumed to be a random variable uniformly drawn in a window $[T_{\text{min}}, T_{\text{max}}]$ every time a new p-sequence is started.
- α : linear reward decay rate in a p-sequence whose length exceeds T .
- r_{max} : constant reward for p-sequences of length less than or equal to T .
- r_{min} : lower clipping value for reward in p-sequences. A good value for is 5, which means that the user is neither rewarding nor punishing the robot for taking their preferred action for too long.
- $r_{\text{non-pref}}$: initial reward value when starting a homogeneous sequence that is not a p-sequence. If the previous homogeneous sequence is a p-sequence, $r_{\text{non-pref}}$ is a function of the length of the p-sequence l as follows: if $l \geq T_{\text{min}}$ we assume that the user is expecting a surprise which will provide some maximal reward $r_{\text{non-pref,max}}$. When $l < T_{\text{min}}$, we expect the surprise to be disliked, so we decrease the surprise

reward linearly: $r_{\text{non-pref}} = r_{\text{non-pref,max}} \cdot (1 - \frac{T_{\min}-l+1}{T_{\min}})$. If the previous homogeneous sequence is not a p-sequence, $r_{\text{non-pref}}$ is a constant $r_{\text{non-pref,base}}$.

- β : exponential reward decay rate for a homogeneous sequence that is not a p-sequence.

Model 2: The “consistent but fatigable”

This type of user values consistency in actions taken but needs shifts from time to time. It is the profile where there always needs to be an uninterrupted routine but this routine has to be changed after some time. The user has a set of preferred actions which he expects to see in long sequences. These sequences alternate between the different actions after some time spent sticking with one of the actions. We assume the same model of boredom used in the previous section, namely the reward starts decaying linearly for the preferred actions after some time interval T . There is no surprise factor associated with this model since we assume that the user does not appreciate surprises.

The parameters of this model are the following (no description provided means the parameters are the same as in the “conservative” model):

- $\mathbf{A}_{\text{pref}} = \{a_1^*, \dots, a_m^*\}$, where $m \geq 2$. p-sequences in this model are defined to be homogeneous sequences formed using one action in \mathbf{A}_{pref} .
- T : optimal length of a p-sequence, after which the user starts getting bored. T is assumed to be a random variable uniformly drawn in a window $[T_{\min}, T_{\max}]$ every time a new p-sequence is started.
- α , r_{\max} and r_{\min} : idem
- $r_{\text{non-pref}}$: initial reward value when starting a homogeneous sequence that is not a p-sequence. A constant in this model.
- β : decay rate of the reward for a homogeneous sequence that is not a p-sequence.

Model 3: The “erratic”

This type of user is mainly interested in change, in both action selection and time-related parameters. They have no clear preferences over the possible actions but require the actions to change according to some average rate not restricted to a window as in model 1 and 2. We assume that at every step the user has some fixed probability p_{sw} to desire a switch to a different action, independently of anything else. Hence the optimal length T of homogeneous sequences follows the distribution: $p(T = t) = (1 - p_{\text{sw}})^{t-1} p_{\text{sw}}$ (for $t \geq 1$), whose average $\mu_T = 1/p_{\text{sw}}$, making μ_T a sufficient statistic. Similar to previously, the reward decreases linearly after T time steps in a homogeneous sequence.

6.3 Learning model parameters from user feedback

Now that we have presented the three user models that we consider, we look at the problem of learning their parameters from user reward sequences. Once these parameters become known, we can then generate personalized sequences of actions maximizing cumulative reward for a specific user. In what follows, we assume that the model to which a particular user belongs to is known a priori. In practice, this can be achieved by prompting the user to select one profile which described them best, or through a set of questions similar to a personality test.

Note that although we have previously raised the problem of dealing with the non-Markovian aspect of user preferences (meaning that the reward of a certain action depends on the history of previous actions), in the way we have modeled the user profiles in the previous section, the model parameters encode the preference dynamics. These parameters are assumed to be unchanged as time goes by, hence we have effectively turned the dynamic problem into a Markovian one. Next, we describe the learning procedure for each of the user profiles introduced.

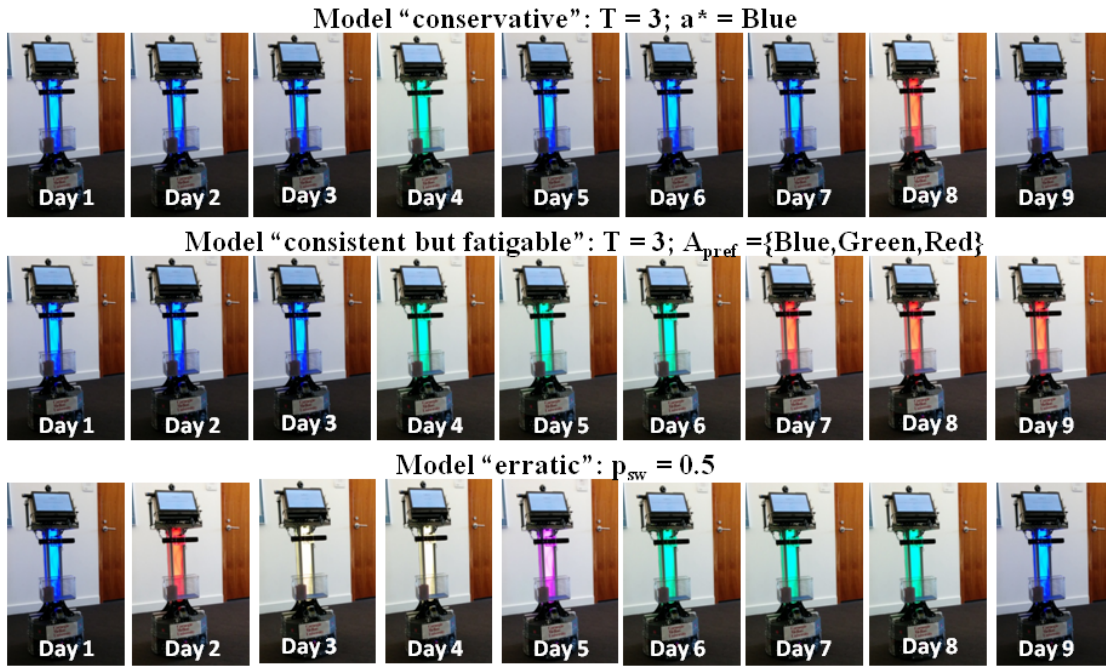


FIGURE 6.1 Sample preferred of animation sequences for the user models presented

6.3.1 Profile “conservative”

In order to learn the parameters of this model, we divide the learning process into two phases: one phase for learning preference sets and the other for learning the time-related parameters. The parameters the agent performing the actions needs to learn are: a^* , A_{surp} , T_{min} and T_{max} .

Phase 1: Learning preference sets In this preliminary phase, actions are uniformly drawn from \mathbf{A} until each action is taken at least n_{th} times, where n_{th} depends on the noise variance estimate $\tilde{\sigma}^2$ and on our target confidence value (for all practical purposes, we use $n_{\text{th}} = 10$). Note that randomization of the sequence of actions to be taken is crucial in this phase since the previous actions can have an influence on the reward of the current action and we would like to dilute this effect. Once we have an average reward estimate for each action, we select a^* to be the action with the maximum estimated reward and \mathbf{A}_{surp} to be the set of all actions whose reward estimates exceed the set threshold r_{th} , where the value of r_{th} has to ensure that $|\mathbf{A}_{\text{surp}}| \geq 1$. It is assumed that the set of best actions to be used for surprises will score high in this phase as well.

Phase 2: Learning time-related parameters In order to learn the two parameters of interest T_{\min} and T_{\max} , the agent first learns to estimate the mean and variance of T (μ_T and σ_T respectively) and uses them to infer the parameter values. To achieve this, the agent follows p-sequences until a need for surprise is detected (more details below). A surprise is restricted to taking an action in \mathbf{A}_{pref} for one time step following a p-sequence. After a surprise, the agent reverts back to following a p-sequence until another surprise is decided upon.

The learning procedure goes as follows: when in a p-sequence of actions, if a downward trend in reward is detected, show a surprise chosen uniformly from \mathbf{A}_{pref} . Since the reward is noisy, a smoother is needed to filter out high frequency noise in the data. We use an exponentially weighted moving average (EWMA) [34] with fixed sample size s , combined with a threshold detector, to detect a downward trend in the reward of the p-sequence. The threshold used in the threshold detector depends on the estimated noise variance in the reward $\tilde{\sigma}^2$. Every time a downward trend is detected, we record the estimated T value associated with the p-sequence. Once enough surprises are chosen, we would have accurate enough estimates of μ_T and σ_T , which can be used to find the time-related parameters as follows:

$$\tilde{T}_{\min, \max} = \tilde{\mu}_T \mp \frac{\sqrt{12\tilde{\sigma}_T^2 + 1} - 1}{2}$$

Note that there is a lag associated with the moving average trend detector. This lag is equal to half the sample size $\lfloor \frac{s}{2} \rfloor$ and $\tilde{\mu}_T$ needs to be adjusted to account for it. Also, for a small number of data points, we might be overestimating σ_T . Hence we set $\tilde{\sigma}_T$ to be half the estimate of the standard deviation in the values of T . This way we impose a more conservative restriction on the values of T which will ensure that $[\tilde{T}_{\min}, \tilde{T}_{\max}] \subset [T_{\min}, T_{\max}]$.

6.3.2 Profile “consistent but fatigable”

Similar to that of the previous model, the learning procedure is still separated into the two phases. However, as far as action selection is concerned, since there is no surprise but only a change factor in this model, the termination of a p-sequence of a_i consists in starting a new p-sequence with an action chosen uniformly in $\mathbf{A}_{\text{pref}} \setminus \{a_i\}$. The first phase for learning preference sets uses the same procedure as before, except that once the average reward

estimates are obtained, we set \mathbf{A}_{pref} to be the set of animations with a reward estimate above r_{th} (possibly different than the one used in the “conservative” model). Here again, the threshold value should be set such that the cardinality m of \mathbf{A}_{pref} is at least 2. The second phase for learning time-related parameters is similar to the one used in the previous model.

6.3.3 Profile “erratic”

For this type of profile, no sets of preferred actions need to be learned since we assume that the user has no clear preferences between the different actions. Hence, the only parameter to learn is the probability of switching p_{sw} . The action selection algorithm is identical to the “consistent but fatigable” model, with $\mathbf{A}_{\text{pref}} = \mathbf{A}$. μ_T can also be estimate as before, and once a good estimate is obtained, we infer our parameter p_{sw} as follows: $\tilde{p}_{\text{sw}} = \frac{1}{\tilde{\mu}_T}$.

6.3.4 Action sequences generation

The learning phase stops when the parameters are learned with some target confidence value. The latter comes in our case mainly from the error rate of the EWMA and depends on the various parameters including noise variance. Once the parameters are learned, appropriate stochastic sequences can be generated according to the estimated parameter values. For models “conservative” and “consistent but fatigable”, we uniformly draw a value of T in the estimated window. For model “erratic”, we follow the same action with probability $1 - p_{\text{sw}}$ and uniformly switch to another action with probability p_{sw} . In this exploitation phase, the feedback requirements can be reduced or eliminated, since we have all the parameters needed to generate optimal sequences which will maximize the cumulative reward for the given user. In practice, occasional user feedback (e.g. asking for a reward) can be used to confirm the model and parameters. We will not provide more detail about this exploitation phase since the focus of this work is on the learning aspect. However, notice that in the learning phase we are already generating sequences which are not too far from optimal.

6.4 Results

In this section, we present a few results showing our implementation of our simulated user’s preference dynamics and our algorithm’s ability to learn the different model parameters. Figure 6.2 shows the evolution of the learning process for single instances of the three models. We consider 8 possible actions arbitrarily labeled 1 through 8. Phase 1 of the learning algorithm can be clearly distinguished in the first two models, after which the algorithm learns the set \mathbf{A}_{pref} ($\{a_4\}$ for model “conservative” and $\{a_2, a_4, a_6\}$ for model “consistent but fatigable”). Once it identifies the preferred sets, the algorithm is also able to adapt to the preferred action dynamics. Notice that whenever there is a notable decrease in the reward, a change is performed, whether creating a temporary “surprise” (a), changing to another steady option (b) or creating erratic change (c).

The simulation was run over 350 time steps with the following parameter values for illustrative purposes. $T_{\text{min}} = 20$ and $T_{\text{max}} = 30$ for the first two models and $p_{\text{sw}} = 0.08$ for

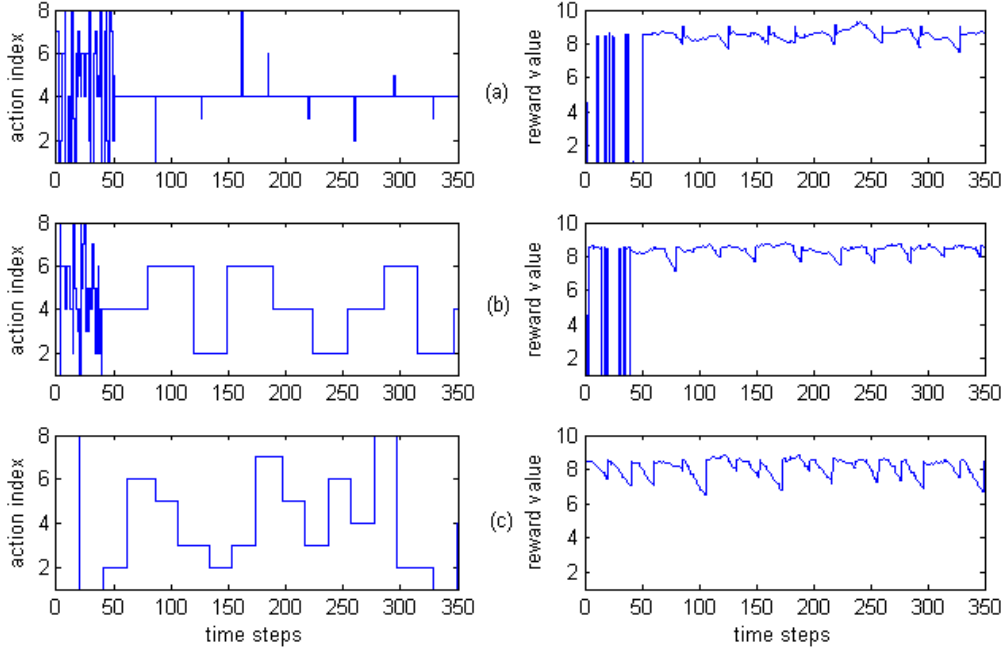


FIGURE 6.2 Simulation results showing sequences of actions taken by the agent and the corresponding reward sequences from a simulated user belonging to: (a) model “conservative”, (b) model “consistent but fatigable” and (c) model “erratic”

the third model. The noise variance σ^2 was set to 0.05. Here are a few results over 1,000 randomized trials.

-**Model “conservative”**: % error in $\tilde{\mu}_T$: 3.5% ; $\tilde{T}_{\min} = 20.94$; $\tilde{T}_{\max} = 30.81$. After rounding, the estimated interval is contained in the true interval.

-**Model “consistent but fatigable”**: % error in $\tilde{\mu}_T$: 2.67% ; $\tilde{T}_{\min} = 21.72$; $\tilde{T}_{\max} = 26.95$. The rounded estimated interval is contained in the true interval.

-**Model “erratic”**: $\tilde{\mu}_T = 12.67$, therefore $p_{\text{sw}} = 0.079$ (1.3% error).

The algorithm was able to learn all the parameters with reasonable accuracy.

6.5 Chapter summary and discussion

In this chapter, we have explored the potentially *engaging* aspect of expressive lights on a robot. More specifically, we looked at personalization of interactions through lights by adapting the choice of animations shown on a particular encounter such that it follows the preference dynamics of the encountered user. We first presented three models for dynamic long-term user preferences, which are mappings from action sequences to reward sequences capturing aspects of boredom and appreciation for change or surprise. Given which model a specific user belongs to, we contributed an algorithm which enables the robot to learn the model parameters using the sequence of rewards given by the user. Our results show that the agent is able to learn the parameters of the model reasonably well and in a relatively

short number of time steps for all three models. Our algorithm is robust to noise, but further experiments are needed to evaluate the degradation in performance as the noise increases. In the future, we plan to enable the robot to also learn which model the user belongs to from the reward sequences themselves. Also, allowing a mixture of the three models with weights to be learned (although it is not exactly clear whether it is a viable idea in a social setting) could diversify the space of sequences generated and alleviate the problem of forcing the users to categorize themselves. Furthermore, requesting a reward from the user at each encounter might have a negative social impact; to overcome this, we could investigate efficient sampling methods to gather rewards in a sparse manner while maintaining accuracy in the learning process.

Chapter 7

Conclusion

7.1 Summary and discussion of contributions

This thesis has focused on enabling robots to effectively communicate information about their state through the use of expressive lights. We focused on three main aspects of such communication modality in relation to robot state, summarized below.

- We came up with **informative** expressions of the robot's state by: (1) *selecting* from it user-relevant elements as a function of situation and context, and (2) *mapping* them to the elements of our expression channel, namely expressive lights.
- We designed these expressions (namely, light animations) to be **legible** such that they require minimal or no training to be understood by first-time users. We informed our design choices with both *design* and *evaluation* user studies.
- We showed how we could enable expressions to be **engaging** by introducing *personalization and adaptation in repeated interactions* with the same user. We accounted for aspects of user preference dynamics such as *surprise effect, fatigue, and taste for change*.

Our technical contributions are the following:

- A *formal framework for light animation control* which divides the animation space into a finite set of *signal shapes* with associated parameters related to *dynamics, color, intensity, and spatial indexing*. An *episodic animation control algorithm* is used to render animatable structures (*complete animation tuple sets*) into *animation frames* on a digital LED strip. Although the analysis is focused on addressable light strips, it easily generalizes to other forms of light arrays,
- An *efficient representation for robot state information* which builds upon *state variables* and *state features* to form structures suited for expression which we call *animation tuples*,

- A *design study* to investigate the design (parameter selection) of animating robot state through expressive lights, and
- An *evaluation study* which shows that expressive lights on a robot can increase understanding of robot states and actions, and
- An *adaptive algorithm* to personalize the change in these light expression over repeated interactions for different *user profiles* that we introduce.

7.2 Future Work

The natural extension of this thesis is its application to truly multimodal robot expression. With multiple modalities, there is a problem of distributing expression across different heterogeneous modalities with different capabilities and limitations. Also, investigating aspects of redundancy and complementarity will be important since modalities, relating to the same system, cannot be treated as completely independent. Along these lines, there is also the problem of synchronization between different modalities, especially when two modalities use the same physical resource (e.g., sound and speech, functional and expressive motion).

Additionally, in the last chapter of this thesis we assumed that we were given rewards from users. In the future, we would like to investigate suitable ways to measure these rewards in practice and possibly extend them to higher dimensions if more than one reward gathering mechanism is in place (for example, facial expression analysis to measure surprise, and ratings on the robot's screen to estimate preference sets regardless of the time component). Furthermore, if the action of gathering a reward is assumed to be costly, then an efficient sampling method needs to be devised to reduce the number of rewards needed to learn the model parameters while maintaining high estimation accuracy and close to optimal policies.

Moreover, we would like to test whether the three user models we present actually capture any aspect of human preferences when interacting with a service robot. Apart from user studies for validating or rejecting some of our assumptions, some existing validated models of consumer might provide us with good insight and need to be further studied (e.g., models used in the toy industry).

Finally, the personalization aspect we started tackling in this thesis opens doors to a multitude of research questions when one considers more complex social interactions such as with subpopulations with special needs for which personalization plays a crucial role to enable smooth interactions with a robot. In particular, we are interested in children with Autistic Spectrum Disorder (ASD) who may show drastically different behavioral patterns according to the level of the disorder. We believe that adapting the choice of modalities to be used for interactions with such individuals will also play an important role in the personalization.

Bibliography

- [1] (n.d.). Light communication of animals. <https://prezi.com/tvoish8wdpq9/light-communication-of-animals/>. [Online; accessed 11-April-2016].
- [2] Alves-Oliveira, P., Di Tullio, E., Ribeiro, T., and Paiva, A. (2014). Meet me halfway: Eye behaviour as an expression of robot's language. In *2014 AAAI Fall Symposium Series*.
- [3] Baraka, K., Paiva, A., and Veloso, M. (2016a). Expressive lights for revealing mobile service robot state. In *Robot 2015: Second Iberian Robotics Conference*, pages 107–119. Springer.
- [4] Baraka, K., Rosenthal, S., and Veloso, M. (2016b). Enhancing Human Understanding of a Mobile Robot's State and Actions using Expressive Lights. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. (Under review).
- [5] Baraka, K. and Veloso, M. (2015). Adaptive Interaction of Persistent Robots to User Temporal Preferences. In *Proceedings of ICSR'15, the International Conference on Social Robots*, Paris, France.
- [6] Berné, C., Múgica, J. M., and Yagüe, M. J. (2001). The effect of variety-seeking on customer retention in services. *Journal of Retailing and Consumer Services*, 8(6):335 – 345.
- [7] Bertin, J. (1983). *Semiology of graphics: diagrams, networks, maps*.
- [8] Betella, A., Inderbitzin, M., Bernardet, U., and Verschure, P. F. (2013). Non-anthropomorphic expression of affective states through parametrized abstract motifs. In *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on*, pages 435–441. IEEE.
- [9] Bethel, C. L. (2009). Robots without faces: non-verbal social human-robot interaction.
- [10] Bobadilla, J., Ortega, F., Hernando, A., and Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, 46:109–132.
- [11] Bonnin, G. and Jannach, D. (2015). Automated generation of music playlists: Survey and experiments. *ACM Computing Surveys (CSUR)*, 47(2):26.
- [12] Campos, J. and Paiva, A. (2011). A personal approach: the persona technique in a companion's design lifecycle. In *Human-Computer Interaction–INTERACT 2011*, pages 73–90. Springer.

- [13] Choi, Y., Kim, J., Pan, P., and Jeung, J. (2007). The considerable elements of the emotion expression using lights in apparel types. In *Proceedings of the 4th international conference on mobile technology, applications, and systems*, pages 662–666. ACM.
- [14] De Lorenzo, R. A. and Eilers, M. A. (1991). Lights and siren: A review of emergency vehicle warning systems. *Annals of emergency medicine*, 20(12):1331–1335.
- [15] De Melo, C. and Paiva, A. (2007). Expression of emotions in virtual humans using lights, shadows, composition and filters. In *Affective Computing and Intelligent Interaction*, pages 546–557. Springer.
- [16] Dragan, A. (2015). *Legible Robot Motion Planning*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.
- [17] Funakoshi, K., Kobayashi, K., Nakano, M., Yamada, S., Kitamura, Y., and Tsujino, H. (2008). Smoothing human-robot speech interactions by using a blinking-light as subtle expression. In *Proceedings of the 10th international conference on Multimodal interfaces*, pages 293–296. ACM.
- [18] Gerathewohl, S. J. (1957). Conspicuity of flashing light signals: Effects of variation among frequency, duration, and contrast of the signals. *J. Opt. Soc. Am.*, 47(1):27–29.
- [19] Goldberg, L. R. (1990). An alternative" description of personality": the big-five factor structure. *Journal of personality and social psychology*, 59(6):1216.
- [20] Ha, J. and Jang, S. (2015). Boredom and moderating variables for customers' novelty seeking. *Journal of Foodservice Business Research*, 18(4):404–422.
- [21] Harrison, C., Horstman, J., Hsieh, G., and Hudson, S. (2012). Unlocking the expressivity of point lights. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1683–1692. ACM.
- [22] Holmes, K. (n.d.). The mood of the chinese internet lights up the facade of beijing's water cube. <http://motherboard.vice.com/blog/video-the-great-mood-building-of-china>. [Online; accessed 11-Feb-2016].
- [23] Hoonhout, J., Jumpertz, L., Mason, J., and Bergman, T. (2013). Exploration into lighting dynamics for the design of more pleasurable luminaires. In *Proceedings of the 6th International Conference on Designing Pleasurable Products and Interfaces*, pages 185–192. ACM.
- [24] Jones, D. N. (2014). Interactive light art show 'congregation' opens at market square. <http://www.post-gazette.com/local/city/2014/02/22/Interactive-light-art-show-opens-at-Pittsburghs-Market-Square/stories/201402220081>. [Online; accessed 11-Apr-2016].
- [25] Kim, M.-g., Sung Lee, H., Park, J. W., Hun Jo, S., and Jin Chung, M. (2008). Determining color and blinking to support facial expression of a robot for conveying emotional intensity. In *Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on*, pages 219–224. IEEE.

- [26] Knight, H. and Simmons, R. (2014). Expressive motion with x, y and theta: Laban effort features for mobile robots. In *Robot and Human Interactive Communication, 2014 RO-MAN: The 23rd IEEE International Symposium on*, pages 267–273. IEEE.
- [27] Kobayashi, K., Funakoshi, K., Yamada, S., Nakano, M., Komatsu, T., and Saito, Y. (2011). Blinking light patterns as artificial subtle expressions in human-robot speech interaction. In *RO-MAN, 2011 IEEE*, pages 181–186. IEEE.
- [28] Langmuir, I. and Westendorp, W. F. (1931). A study of light signals in aviation and navigation. *Journal of Applied Physics*, 1(5):273–317.
- [29] Lee, M. K., Forlizzi, J., Kiesler, S., Rybski, P., Antanitis, J., and Savetsila, S. (2012). Personalization in hri: A longitudinal field experiment. In *Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on*, pages 319–326. IEEE.
- [30] Leigh McAlister, E. P. (1982). Variety seeking behavior: An interdisciplinary review. *Journal of Consumer Research*, 9(3):311–322.
- [31] Leite, I. (2015). Long-term interactions with empathic social robots. *AI Matters*, 5(1):13–15.
- [32] Liu, H., Liu, K., and Zhao, Q. (2013). Learning in a changing world: Restless multiarmed bandit with unknown dynamics. *Information Theory, IEEE Transactions on*, 59(3):1902–1916.
- [33] Lloyd, J. E. (1971). Bioluminescent communication in insects. *Annual review of entomology*, 16(1):97–122.
- [34] Lucas, J. M. and Saccucci, M. S. (1990). Exponentially weighted moving average control schemes: properties and enhancements. *Technometrics*, 32(1):1–12.
- [35] Mason, M. and Lopes, M. (2011). Robot self-initiative and personalization by learning through repeated interactions. In *Human-Robot Interaction (HRI), 2011 6th ACM/IEEE International Conference on*, pages 433–440. IEEE.
- [36] Mutlu, B., Forlizzi, J., Nourbakhsh, I., and Hodgins, J. (2006). The use of abstraction and motion in the design of social interfaces. In *Proceedings of the 6th conference on Designing Interactive systems*, pages 251–260. ACM.
- [37] Nijdam, N. A. (2009). Mapping emotion to color.)(Eds.): *Book Mapping emotion to color* (2009, edn.), pages 2–9.
- [38] Perera, V., Soetens, R., Kollar, T., Samadi, M., Sun, Y., Nardi, D., van de Molengraft, R., and Veloso, M. (2015). Learning task knowledge from dialog and web access. *Robotics*, 4(2):223–252.
- [39] Rea, D. J., Young, J. E., and Irani, P. (2012). The roomba mood ring: An ambient-display robot. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 217–218. ACM.

- [40] Rosenthal, S., Biswas, J., and Veloso, M. (2010). An Effective Personal Mobile Robot Agent Through Symbiotic Human-Robot Interaction. In *Proceedings of AAMAS'10, the Ninth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Toronto, Canada.
- [41] Schanda, J. (2007). *Colorimetry: understanding the CIE system*. John Wiley & Sons.
- [42] Seittinger, S., Taub, D. M., and Taylor, A. S. (2010). Light bodies: exploring interactions with responsive lights. In *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*, pages 113–120. ACM.
- [43] Sung, J., Grinter, R. E., and Christensen, H. I. (2009). Pimp my roomba: designing for personalization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 193–196. ACM.
- [44] Szafir, D., Mutlu, B., and Fong, T. (2015). Communicating directionality in flying robots. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 19–26. ACM.
- [45] Tapus, A., Țăpuș, C., and Matarić, M. J. (2008). User—robot personality matching and assistive robot behavior adaptation for post-stroke rehabilitation therapy. *Intelligent Service Robotics*, 1(2):169–183.
- [46] Veloso, M., Biswas, J., Coltin, B., and Rosenthal, S. (2015). CoBots: Robust Symbiotic Autonomous Mobile Service Robots. In *Proceedings of IJCAI'15, the International Joint Conference on Artificial Intelligence*, Buenos Aires, Argentina.
- [47] Vircikova, M., Pala, M., Smolar, P., and Sincak, P. (2012). Neural approach for personalised emotional model in human-robot interaction. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–8. IEEE.
- [48] Wang, X., Wang, Y., Hsu, D., and Wang, Y. (2014). Exploration in interactive personalized music recommendation: a reinforcement learning approach. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 11(1):7.
- [49] Wolfe, J. M. and Horowitz, T. S. (2004). What attributes guide the deployment of visual attention and how do they do it? *Nature reviews neuroscience*, 5(6):495–501.
- [50] Wright, A. (2009). The colour affects system of colour psychology. In *AIC Quadrennial Congress, 2009*.
- [51] Wright, B. and Rainwater, L. (1962). The meanings of color. *The Journal of general psychology*, 67(1):89–99.
- [52] Xia, G., Tay, J., Dannenberg, R., and Veloso, M. (2012). Autonomous robot dancing driven by beats and emotions of music. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, pages 205–212.

Appendix A

Protocol for Arduino “serial in” communication

The Arduino receives commands to start an animation on the light strip, defined by the parameters specified.

Protocol: command syntax:

[param1]=[val1] [param2]=[val2] [param3]=[val3] ... \n

will start a new animation with the parameters specified. If required parameters are not specified, the default values will be used. The order of parameters does not matter.

List of parameters and values:

- wv: wave shape (signal shape)
 - 1: Square (requires D, T, R, G and B)
 - 2: Triangular (requires D, T, R, G and B)
 - 3: Modulated periodic (requires nSub (max is 5), wv1 ... wv[nSub], D1 ... D[nSub], T1 ... T[nSub], R, G and B) nSub is the number of subperiods in one period and wv[i] is either 1 or 2.
 - 4: Static color/intensity change (requires R, G and B)
 - 5: Static color/intensity change in space (requires nSeg (max 3), R1, G1, B1 ... R[nSeg], G[nSeg], B[nSeg], frac1, ... frac[nSeg]) nSeg is the number of different color segments to be included, starting from lower part of strip. frac[i] is the fraction of the strip (0-1) occupied by segment i (frac[i]’s should sum to 1).
- D: dynamics parameter (0-1)
- T: period (in ms)
- R,G,B: max color intensity values (0-255) for each channel

Total list of parameters with type and default values: wv (int) (2), D (double) (0.5), T (int) (1000), R (int) (0), G (int) (255), B (int) (255), nSub (int) (2), wv1 (int) (1), wv2 (int) (1), wv3 (int) (1), wv4 (int) (1), wv5 (int) (1), D1 (double) (0.5), D2 (double) (0.5), D3 (double) (0.5), D4 (double) (0.5), D5 (double) (0.5), T1 (int) (500), T2 (int) (500), T3 (int) (500), T4 (int) (500), T5 (int) (500), nSeg (int) (2), R1 (int) (0), G1 (int) (255), B1 (int) (255), R2 (int) (0), G2 (int) (255), B2 (int) (255), R3 (int) (0), G3 (int) (255), B3 (int) (255), frac1 (double) (0.5), frac2 (double) (0.5), frac3 (double) (0).

Example:

(Make sure to add an endl at the end of the command)

Regular Red Blinking with period 1s: wv=1 D=0.5 T=1000 R=255 G=0 B=0

Note: Failure to respect the syntax will result in turning all the pixels off and disabling the serial communication with the Arduino board.

The full Arduino code which takes in the commands above as a serial input can be found at the following repository <https://github.com/kobotics/LED-animation>