PSEUDO-SALIENCY FOR HUMAN GAZE SIMULATION

PETER NICHOLAS CARUANA

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

SEPTEMBER 2022

ABSTRACT

Understanding and modeling human vision is an endeavor which can be; and has been, approached from multiple disciplines. Saliency prediction is a subdomain of computer vision which tries to predict human eye movements made during either guided or free viewing of static images. In the context of simulation and animation, vision is often also modeled for the purposes of realistic and reactive autonomous agents. These often focus more on plausible gaze movements of the eyes and head, and are less concerned with scene understanding through visual stimuli. In order to bring techniques and knowledge over from computer vision fields into simulated virtual humans requires a methodology to generate saliency maps. Traditional saliency models are ill suited for this due to large computational costs as well as a lack of control due to the nature of most deep network based models. The primary contribution of this thesis is a proposed model for generating pseudo-saliency maps for virtual characters, Parametric Saliency Maps (PSM). This parametric model calculates saliency as a weighted combination of 7 factors selected from saliency and attention literature. Experiments conducted show that the model is expressive enough to mimic results from state-of-the-art saliency models to a high degree of similarity, as well as being extraordinarily cheap to compute by virtue of being done using the graphics processing pipeline of a simulation. The secondary contribution, two models are proposed for saliency driven gaze control. These models are expressive and present novel approaches for controlling the gaze of a virtual character using only visual saliency maps as input.

"Would you tell me, please which way I ought to go from here?"

"That depends a good deal on where you want to get to." said the cat.

"I dont much care where." said Alice.

"Then it doesn't much matter which way you go."

"...So long as I get *somewhere*."

"Oh, you're sure to do that, if only you walk long enough."

– Lewis Carroll, Alices Adventures in Wonderland

ACKNOWLEDGEMENTS

First, I would like to express my deep gratitude and appreciation to my supervisor, Petros Faloutsos. As an instructor he inspired me to learn. As a supervisor he provided the environment and guidance for me to grow both academically and personally. I could not have asked for a better supervisor.

My sincere appreciation and thanks goes out to my colleague and friend, Melissa Kremer. Her support, mentoring and camaraderie helped me push through difficult times and finally bring my thesis to its completion.

I would also be remiss if I did not mention the invaluable advice, lessons and discussions I had with Professor John Tsotsos, Dr. Calden Wloka as well as all my colleagues in the Department of Electrical Engineering and Computer Science at York University over the last two years.

To my parents, my brother, and my dear friends. Thank you for your love, support and encouragement. And no, I don't expect you to read this entire thing.

Finally I would like to thank the Department of Electrical Engineering and Computer Science at York University for providing the financial support needed to complete this thesis work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Much work is being done in the field of computer vision to create saliency models, which are able to approximate how humans perceive stimuli (in the form of images) and output representations of human attention with respect to these stimuli. These models are constantly improving and well adapted to their problem domains. However, they are often computationally expensive and somewhat uninterpretable due to being developed largely as machine learning models. This leads to challenges when trying to utilize them in real-time settings, particularly in simulation. These concepts can be migrated into virtual settings through simple computational models of saliency which take advantage of the control and information available in simulated scenarios.

The goals of this work are to describe the process and methods required to A) generate pseudo-saliency map approximates in a real-time simulation in an efficient and neuro-physiologically plausible manner, and B) use those saliency maps for procedurally generating human gaze movements in a simulated humanoid character. This work specifically targets the area of casual pedestrians with minimal interactions. The driving belief behind these ideas is that basing virtual human gaze on real-time measures of visual attention allows for the generation of more realistic and interpretable gaze behaviours as well as opens the door for more exploration and application of research from the computer vision and vision sciences. By basing gaze control on visual input (in the form of saliency), there is a new freedom to be found in the compartmentalization of attention and top-down psychological factors away from the low-level visual-motor responses of humans to given interpreted stimuli. The psychological and biological factors for real humans are complex, as such modelling these factors is challenging. In this case, trying to model human vision must also contend with complexities such as goals, personality, intent etc. which can all impact the viewing behaviours of humans. As an example, a person walking down a city street they live on will likely place different importance on things like street signage versus someone unfamiliar with the area. Being able to easily define and apply factors such as this is extraordinarily valuable for those wishing to model virtual humans.

Portions of this document have been accepted for publication in [28] and [29]. These publications include content in Sections 2.2, 2.3, and Chapter 3. The Concepts in Chapters 3 and 4 were jointly developed between authors Peter Caruana, Melissa Kremer, Dr. Brandon Haworth, Dr. Mathew Schwartz, Dr. Mubbasir Kapadia, and Dr. Petros Faloutsos. Content in Chapter 4 has been submitted for review and is under consideration. Conceptualization and implementation of the work outlined in chapter 3.1 were done jointly between Peter Caruana and Melissa Kremer, with advice and guidance from Dr. Brandon Haworth, Dr. Mubbasir Kapadia and Dr. Petros Faloutsos. Dataset generation in chapter 3.2 was implemented by Melissa Kremer, with conceptualization done between Melissa Kremer and Peter Caruana. The parameter optimization process, details and evaluations were done by Peter Caruana. In Chapter 4, conceptualization was done primarily by Peter Caruana, with input from Dr. Brandon Haworth Dr. Mathew Schwartz and Dr. Petros Faloutsos. Implementation details and research were done by Peter Caruana.

Chapter 2 covers the theoretical background on human gaze, saliency and attention, fixation prediction and gaze control. It discusses physiological and psychological mechanisms which underlie how humans view and perceive stimuli. There is an emphasis on how these findings have been translated into work in the domain of computer vision, as well as simulation in the form of gaze control.

Chapter 3 explains the proposed approach to generate pseudo-saliency maps representing human visual attention in urban virtual settings, for the purposes of simulation. It discusses the factors which comprise the parametric model, as well as their origins in prior works. Additionally, this chapter explains and demonstrates how the proposed model can be optimized to mimic the output of state-of-the-art saliency models. The relative similarity of the optimized output is examined with respect to the SALICON saliency model as a baseline.

Chapter 4 presents two possible models for utilizing real-time pseudo-saliency maps for controlling the gaze of a virtual agent. The particle model interprets gaze as a physical particle being acted upon by attractive forces. These forces come from interpreting saliency maps as a potential field where areas of high salience are represented as ”wells” in the potential field thus drawing the particle towards them. The Probabilistic model treats saliency maps as their original interpretation, probability densities for fixations. Fixation points are drawn from the distribution, and the gaze of the observer is interpolated to look at these points.

Chapter 5 is the conclusion, and it summarizes the main points of this work as well as discusses possible future work.

CHAPTER 2

RELATED WORK

2.1 Human Gaze

2.1.1 Gaze Movements

Human gaze movements have been the subject of much study for well over a century. Over this time eye movements have been classified into seven categories. The standard set of eye movements generally agreed upon are (From Tsotsos et al.[62]):

1. **Saccades:** Voluntary jump-like movements that move the retina from one point in the visual field to another;

2. **Microsaccades:** Small, jerk-like eye movements similar to voluntary saccades, but with much smaller amplitudes from 2 to 120 arc-minutes;

3. **Vestibular-Ocular Reflex:** These movements stabilize the visual image on the retina by causing compensatory changes in eye position as the head moves;

4. **Optokinetic Nystagmus:** This stabilizes gaze during sustained, low-frequency image rotations at constant velocity;

5. **Smooth Pursuit:** Voluntary eye movements which track moving stimuli;
6. **Vergence:** Coordinated movements of both eyes, convergence for objects moving towards and diverging for objects moving away from the eyes;
7. **Torsion:** Coordinated rotation of the eyes around the optical axis dependent on head tilt and eye elevation;

There are quite a few different types of eye movements, each with its own complexities as well as implications for human gaze. For the scope of this work, the focus will be primarily on eye movements; saccades, microsaccades and smooth pursuits (and to a small degree, vestibular-ocular reflex); though it is beneficial to be familiarized with the full suite of eye movements human vision utilizes. Becker [4] summarizes much of the current understanding of saccadic control. A model for saccadic control can be quite complicated, as it is represented by a 4th order ODE however a 1st order approximation can suffice. The relationship between saccade duration and saccade amplitude can be written then as a linear relationship $D = D_0 + d \cdot A$; which can be used to interpolate gaze to fixation points, where:

- $D_0 \simeq 20 - 30$ ms
- $d \simeq 2 - 3$ ms/deg
- A is amplitude in degrees

In addition to eye movements, head movements are also a crucial part of gaze. Due to the slower and more limited nature of head movements, they tend to be less categorized. As a general rule, humans tend to simply align their heads with what they are looking at. According to Nakashima et al.[42], this is because a discrepancy between head and eye movements can cause interference in visual processing as well as a degradation in accuracy for localizing attentional focus and hand-eye coordination. Head movements are less erratic than saccades largely due to the greater moment of inertia of the head as compared to the eyes. Sudden fast head movements put a large strain on the human neck, and thus are generally avoided in casual free-viewing. One study [10], found that head movement duration can range between 200 to 800 ms after a series of saccades make up a gaze shift, with head rotation speeds increasing for larger gaze shifts. In contrast, a single saccade-fixation action typically lasts around 200 ms. In one study, Freedman [11] found that head movement duration averaged 450 ms, while saccades took around 200 ms in a single trial. From this same work, Freedman also observed a 100 ms delay of peak head velocity after peak eye velocity is reached. This is in line with what papers like Ruhland et al. [55] use when modelling head and eye movements, where they utilize a delay of 20 to 50 ms. To package this all nicely, head movements are slower and delayed, typically following along after a gaze shift. The combined set of eye and head movements will be referred to frequently as *gaze movements.*

A model which aims to emulate human gaze should be able to parameterize and replicate these kinds of gaze movements, or at least a sufficient subset of them. The large problem however with generalizing a control structure is that human gaze behaviours tend to be very diverse, and very idiosyncratic as noted by Ruhland et al. [55]. There are also subtle but measurable effects in eye movements, such as differences in vergence dependent on the motion being horizontal or vertical [13]. In light of these complexities, the scope of a model must be clearly defined. Gaze target selection draws from attention, with consideration of deliberate intent (e.g. "I want to look at that sign"). which then informs the corresponding gaze movement. For example, a slow-moving object of interest in view will elicit a smooth pursuit. However, if this target is moving beyond a certain speed, smooth pursuit becomes no longer possible and the human visual system will resort to "catch-up" saccades to keep track of the object. A model of gaze should be able to generate a wide range of plausible gaze movements given knowledge, or a map of how the person is attending to their world.

2.1.2 Memory and Inhibition of Return

Inhibition of return (or IOR) is described by Klein and Ivanoff [25] as a delayed response to stimuli in a peripheral location which was previously attended to or looked at. Originally found in Posner et al.'s [48] 1984 study, and later defined in their 1985 followup [49], inhibition of return's function appears to be orienting gaze towards novel locations which facilitates foraging and other search behaviours. It is fairly intuitive *why* humans do this, for example, if one were searching their office for a specific item it would make sense to avoid searching where they have already looked. Alternatively, if they were just trying to gather information about their environment, the same mechanism aids in information gathering. IOR typically appears in the literature when the stimulus event is not task-relevant, or there is no task given to the observer [25]. When test subjects were tasked with making saccadic movements which seemed most comfortable to them after viewing a brief stimulus, they most often would look away from the location of the stimulus. Across multiple studies, it was found that IOR is often encoded in environmental coordinates rather than retinal coordinates [25, 26]. This effect also appears in early IOR literature [49, 48]. Further studies have also shown that in some instances IOR appears to be encoded on an object basis [59, 60, 1, 14]. Both environmental location and object attachment as IOR encodings have strong experimental evidence supporting the existence of both strategies. The question then becomes, what are the circumstances in which either encoding occurs? Klein [25] suggests that this change in encoding depends on contextual factors such as whether the observer or objects in view are moving, as well as what the task and intent of the observer are. In early studies, IOR appeared as related to a reluctance of motor response to focus on particular locations, as opposed to inhibition or suppression of attention. However, studies have found IOR to occur in spatial tasks as well, not just in stimulus-response. These findings have shifted the general consensus that IOR does indeed occur on an attentional level as well as ocular motor response. The reasoning, again, appears to be contextual. For example, the types of stimuli along with the difficulty in discriminating the stimuli within the visual field both affect how IOR is introduced on the attentional level. The presence of IOR on attention is further supported by findings that IOR also appears in auditory [40, 41, 57] and tactile [59] modes. Results are consistent in demonstrating that inhibition of return's effect is to inhibit responses typically associated with stimuli. Narrowing down how IOR mechanisms will function is a difficult task, affected by many factors. Studies have found that IOR typically takes between 100 and 200 ms of cued saccade fixation to kick in, aligning with the reported latency between saccades at around 200 to 250 ms [12]. The effects can last several seconds, however, this can easily be affected by changes in the scene or task, as well as the observer's internal state.

The multitude of open questions as well as contextual changes makes it difficult to define an inhibition of return mechanism which can be used as part of a gaze control system. Factors like environment, agent factors, intent, and task all need to be taken into account to decide on things like which encoding to use etc. That is not to mention the open questions on specific mechanisms within IOR. There are many valid ways to implement IOR. In this work, the attempt is to focus on a subset of factors and contexts and suggest an IOR mechanism contingent on that, based on the previously mentioned literature. It is the hope that implementing gaze control systems based on attention and vision literature opens up possibilities to explore many of the open challenges yet unexplained.

2.2 Saliency and Attention

Human visual attention is a complex process which involves multiple cognitive and sensory systems. These systems use both extrinsic and intrinsic information to formulate visual perception [63]. There is much overlap in interpretations of saliency and visual attention. For the purposes of this work, the distinction is made between attention as it pertains to controlling gaze versus saliency which represents pre-attentive processing of where one is *likely* to attend. Saliency maps are the most common way to represent saliency in 2D static images. In the most common conceptualization of saliency maps, they describe where and to what an observer is likely to pay attention by mapping a distribution of all possible scan paths [6, 19, 21]. Put another way, a saliency map tells the probability that a person will look at a particular location. By understanding saliency maps in this way, they can represent potential targets for gaze shifts, with priority (or probability) given in order of decreasing salience [61].

A saliency map which is designed to replicate human cognitive processing should reflect outcomes in attention, implicitly encoding targets for fixations. Indeed, generating plausible representations of saliency should require consideration of the types of factors which impact and draw attention. Kümmerer et al. [34] found that high-level features (objects) are often good predictors for fixations, and therefore attention. Treue [61] noted that how humans perceive saliency appears to be an interplay between bottom-up features (in the form of visual stimuli) and top-down behavioural and attentional factors. Bottom-up saliency gets combined with top-down modulatory effects coming from visual attention. The effect is that stimuli are either strengthened or weakened on the basis of behavioural or semantic importance. Studies from Martinez et al. [39], Reynolds and Desimone [52], and Reynolds et al. [53] have demonstrated that attention modulates and enhances the contrast of stimulus in the visual field. In general, it is understood that humans also tend to direct more attention to things closer to the center of their field of view [9, 22, 38]. This effect is often referred to as a *center bias*, and is commonly included when considering models of saliency. However, Brefczynski and DeYoe [5] found neurological evidence that human attention can also be directed to other locations in the visual field beyond the center of gaze, showing the plausibility and biological basis for localized attention modulation. Other common features which have high salience include the speed of stimuli [24], human faces [22], and proximity [36]. The human visual system often optimizes gaze to aid in identification of targets. When viewing faces for example, one such study demonstrated that gaze is consistently aimed slightly below the target's eyes [47].

2.2.1 Rule-Based Saliency

The salience of visual stimuli can be thought of as measuring the stimulus relevance with respect to the human visual system. One method for producing saliency maps is by estimating stimuli salience with respect to a human observer in accordance with well-defined analytical rules. This concept has evolved over previous decades, beginning with Itti et al.'s seminal work [20]. Today referred to as the *IKN* model, they proposed a bottom-up visual attention model for 2D images to produce corresponding saliency maps. There have been numerous follow-up models over the years. Bruce and Tsotsos [6] proposed Attention based on Information Maximization (AIM), which is derived from the principle that attention seeks to select the most informative visual content. Around the same time, the DVA model from Hou and Zhang [17] is based on the rarity of visual features. As noted by Bruce and Tsotsos [7], it is difficult to directly evaluate the performance of rule-based saliency models, and there remains disagreement regarding the accuracy of results as compared to human eye-fixations from eye tracking studies.

2.2.2 Machine Learning-Based Saliency

More recent approaches for saliency map generation utilize deep neural networks. Most commonly, these models aim to implicitly encode pre-attentive processing from large sets of human eye tracking data. Notably, models such as Huang et al's SALICON [18] or Kümmerer et al.'s DeepGazeII [32] used eye tracking data as ground truth for training models to generate saliency maps to more closely match real humans. Qualitatively, these kinds of models tend to focus highly on objects, humans, and human faces. In Judd et al. [22], information from face detection was explicitly included in their models for visual saliency. It is a fair assumption to make that machine learning based models which use human-generated ground truth data will tend to assign salience to common features found in eye tracking studies [21, 34, 44].

2.2.3 Pseudo-Saliency

In traditional terms, saliency models are meant to run on images of the real world and are calculated from information within the images given to them. Pseudo-saliency, by contrast, is an attempt to mimic ideas from saliency and use them in the context of a virtual scene. By pre-embedding and calculating saliency in a simulation, one can use the concept of saliency for a variety of real-time tasks without actually worrying about visual processing. In some sense, this approach is closer to a replication of human processing which is not entirely dependent on the colours of stimuli we see, but on a complex understanding of objects, their movement, importance etc. Some approaches do not represent saliency visually, such as Oyekoya et al.'s work [43] or their follow-up [27]. These methods use a weighted sum of various factors to calculate a saliency score for all objects within view of a character. Factors include speed, orientation, proximity and angular speed. These models are simple and allow for easy target selection through ranked ordering of targets. The simplicity and effectiveness of such parametric models directly inspired the Parametric Saliency Maps model described in this work. Ağıl et al. [2] proposed a very similar model in order to compute points of interest, however also tried to include social factors such as considering grouped entities, gaze copying and more. Their function for calculating saliency is the same as Oyekoya et al., however, includes some additional factors. Notably, they include what they called *attractiveness*; a value representing the importance inherently tied to an entity. While simple, this idea is quite powerful as it allows for simple intuitive tuning of the model towards particular targets. For example, a uniformed officer or firefighter would likely be much more interesting and draw a person's attention than your average pedestrian in a crowd. Similarly, the ability to decrease saliency on an entity level may facilitate an IOR mechanism.

2.2.4 Saliency Metrics

Noted by Bylinskii et al.[8], Comparing and scoring saliency maps is notoriously difficult. It is easy to define metrics, and indeed one does not have to search hard to find tens of different metrics. Interpreting results is not always a straightforward task. This leads to challenges when comparing results with respect to ground truth as well as between models themselves. Different metrics tend to penalize and reward different things, making comparisons often unfair. For this reason, it is highly recommended by experts in the field [8, 33, 54] to use multiple metrics for evaluation. Bylinskii et al. [8] in particular showed that the choice of metric combinations needs to be diverse as well as complementary, as many metrics can often contradict each other. Kullback-Liebler divergence (KL) measures how different two probability distributions are, with a score of 0 when distributions are identical. It is often used as a metric in machine learning algorithms such as generative adversarial networks (GANs), but it is also often used in the saliency literature to measure the dissimilarity of saliency maps [35, 51, 58]. Kümmerer et al.[33] and Riche et al.[54] both recommend that Kullbach-Liebler divergence be used in combination with some other location-based score, such as similarity score (SIM) or correlation coefficient (CC). Bylinskii et al. showed that KL-Divergence and SIM complement each other quite well and are good candidates for a set of metrics.

For discrete distributions $P(x), Q(x)$ defined on probability space χ, Kullback-Liebler divergence (KL) is defined as:

$$D_{KL}(P||Q) = \sum_{x \in \chi} P(x) \log \left(\frac{P(x)}{Q(x)} \right) \tag{2.2.1}$$

Similarity score (SIM) is defined as:

$$SIM(P, Q) = \sum_{x \in \chi} \min(P(x), Q(x)) \tag{2.2.2}$$

2.3 Fixation Prediction

Fixation prediction refers to models and algorithms which aim to replicate the mechanisms for saccadic selection in humans and output prediction for a fixation point and/or a series of fixation points. Active fixation prediction from Wloka et al. [64] aims to generate a temporal series of fixation positions within a 2D image. Connecting each fixation in temporal order generates a scan path. They accomplish this using a tiered saliency approach, which blends a coarse feature map on the periphery of the view with a highly detailed saliency map located at the point of fixation. In combination with a temporal inhibition of return (IOR) mechanism, this method is able to generate plausible scan paths on images which are highly similar to the structure of ground truth scan path data. A notable takeaway from this approach is the importance that selective attention suppression in the periphery plays in making local micro-saccadic point predictions. Additionally, the inclusion of IOR is highly important, as noted by Klein [26] that IOR is consistently found to be present in studies on fixation and saccadic movements. A more recent example of fixation prediction comes from the Deepgaze III model from kümmerer et al. [31], where they trained a deep neural network to predict and generate temporal fixation location predictions from fixation density maps (saliency maps) for free-viewing of natural images. The model is of particular interest due to the greater importance placed on scene content for predicting the next fixation points, over previous scan path history. The limitation there would be that this would de-emphasize IOR as an important mechanism in preventing subsequent predictions in nearby areas. Many findings summarized by Zetsche [65] conclude that saccadic selection in humans; and to a similar degree in primates, favours regions with greater visual structure within images and the visual field. Comparing data from real human fixation studies versus random point selection on datasets of images, regions selected from human fixations consistently show higher signal variance. The mean-variance ratio of random versus real fixations $\sigma^2_{eye}/\sigma^2_{rand}$ is typically found to be approximately 1.35. A reasonable expectation is that in the active-vision context, fixations should be focused around areas of structure, typically synonymous with interesting stimuli such as other humans, signs, vehicles etc.

The problem generally with most approaches to fixation prediction is the focus on free viewing of static images. As noted by Tstotsos, J., the limitation with this is that humans do not process the world in static 2D images. Changes in gaze do not affect the viewing image in the same way that the view seen by a person in real life is constantly changing as they look and move around their environment. Human visual systems contend with stimuli changes from dynamic environments in addition to egocentric effects when gaze and locomotive movements occur, i.e. turning your head entirely changes what you see, as does moving your position. Fixation prediction in active-vision applications such as simulation or robotics must contend with temporally changing environments, changes in observer location, gaze orientation and spatial-temporal memory.

2.4 Gaze Control

One of the closest implementations to the saliency-driven gaze control described in this work is Peters et al. [46]. They used the IKN model [20] to generate saliency maps from the perspective of a virtual agent. The map was used to determine which objects within view would be "salient" and queued them as targets in the scene database. Additionally, in their work, there was an implemented form of memory where agents would keep track of scene objects that they have observed. Noted by the authors, their work was done according to a spirit of 'sensory honesty', which meant attempting to use as little simulation knowledge as possible. The saliency-driven gaze control methods described in this work build off of their philosophy but attempt to take it further by including no information about the transforms of objects in the scene database in gaze, driving gaze entirely by visual stimulus alone. By their own comments, the most significant limitation of the authors was the lack of a top-down attention component. The pseudo-saliency map generation method, parametric saliency maps (PSM), described by this work is meant to address this limitation in the prior art. The benefit of using parametric saliency maps is that the processing time is limited only by the cost of the attention model and the rendering pipeline. Another limitation of prior virtual gaze control models is simply that humans don't have a scene database to draw information from. Approaching the problem of gaze and attention from a visual stimulus-driven standpoint opens the door for more grounded modelling of virtual humans. For over two decades there have been numerous complex totalistic models developed for automating gaze behaviour, in the form of cognitive models of attention and intent which form a high-level controller [23, 3, 37]. Rather interestingly, these models attempt to join ideas of task relevance and action to inform gaze movements. This is often overlooked in more simple gaze control models, despite things such as environmental conditions impacting visual understanding of the environment as well as general locomotion and movements. For example, the increase in foot clearance on steps in different lighting levels [15]. Similarly, binocular vision is an important facet in how footholds are located during locomotion, and degradation of binocular vision impacts visuomotor control [13]. The active gaze control methods proposed in this work do not seek to supersede these prior models, instead they pose a simpler framework for authoring and generating gaze behaviours in a way which compartmentalizes attention and intent away from control. This work fits in as a link between the vision-based approach by [46] and the high-level control structures of [23, 3, 37]. There are pseudo-saliency-driven gaze approaches such as [43], [2] which do not use saliency maps or visual stimulus as input control. Rather, gaze is controlled by explicitly targeting the transforms of objects within the scene. These approaches create reasonably plausible procedural gaze animations using simple selection rules. However, there are some notable limitations. One such limitation is scaling; as a scene becomes increasingly complex the computational costs of such gaze models too will increase. For the purposes of modelling complex visual-cognitive processes, they are also limited because of the non-visual stimulus-driven approaches.

CHAPTER 3

PARAMETRIC SALIENCY MAPS

In order to use saliency maps in a simulation, a fast and effective method is needed to generate saliency maps in real-time for large numbers of agents. Using popular/state-of-the-art models to this end is unfeasible for multiple reasons. The first and largest problem is speed; models like SALICON or DeepGazeII are simply too computationally expensive to be used in a real-time simulation. Second, many models are inherently black box solutions making it difficult to understand and interpret some the results produced. This coupled with the fact that virtually all models were not designed for use with synthetic data, which could lead to even more instability when used in simulations.

3.1 Parametric Saliency Maps (PSM)

Saliency is an interesting and effective way to represent pre-attentive processing of the visual field. It encapsulates how one may perceive stimuli in the world and their relative importance. The most common way this tends to be represented is in the form of a saliency map, which is a 2D, grayscale image highlighting the saliency of stimuli within an original view (image). Models which take as input a real image and output a saliency map are known as saliency models. Many such models have arisen, primarily with the intent to replicate a probability distribution for all possible fixations in an image.

There has been much success in this area, but one might be justified in asking how these models might be utilized. Seeing as the goal is to replicate *human* saliency, the obvious and perhaps underlooked approach is to use saliency models when designing virtual human agents. This work intends to bring the concepts and models from the field of computer vision and integrate them for the purposes of simulation.

There are many excellent saliency models in existence today such as Huang et al.'s SALICON [18], Kümmerer et al.'s DeepGaze II [32], or Pan et al.'s Salgan [45], to name a few. While these are effective for what they do, bringing them to a virtual setting presents many challenges. To begin with, it is difficult to expect identical performance when transitioning from real images (which these models were designed for) over to synthetic images generated in a rendering engine. Having better visual fidelity in the rendering may assist with this however is not always desired or practical. Additionally, these models are designed to work with particular kinds of data and scenes which were part of training. As such there simply may not be enough flexibility to accommodate the infinite possibilities for designing a virtual scene. Finally; and this is the largest issue, they are slow and computationally demanding to run. In some cases, processing a single image may take between 1 to 10 seconds running on modest hardware. For any practical use; especially simulation and games, this is not feasible. It is too slow and resource-intensive to be used in real-time for multiple virtual agents simultaneously. A suggestion could be to use simpler models such as the famous IKN [20], and indeed this has been done [46]. However, these simpler rule-based models come with questions of biological plausibility and realism when compared to the previously mentioned models. Indeed, comparing results it is clear to see the difference in the kinds of saliency maps which are output. Also, while faster, this may still be too expensive when scaled to large simulations. Perhaps the largest issue, noted by Peters [46], is that with either approach there is no ability to model and apply changes in attention, such as target suppression or target focus.

The solution to all of the above issues and challenges is to devise a pseudo-saliency approach which can output saliency maps reasonably congruent with what would be expected from state-of-the-art models. The model should also be parameterized to allow for simple adjustments to be made for the purposes of attention, memory or behavioural changes. The method described in this chapter; dubbed *Parametric Saliency Maps* (PSM), outlines the saliency parameters, how they are combined, and how this can be simply implemented in simulation using GPU programming in a 3D engine.

3.1.1 Model

There are 7 parameters which make up the parametric model proposed in this work. While there are many more factors which affect pre-attention and attention, this subset was chosen from psychology literature as well as from other pseudo-saliency models [43, 2], as an acceptable representation while not being overly complex. The following describes the saliency factors chosen.

- *Interestingness* - Interestingness is saliency which is intrinsic to a given entity, similar to the attractiveness factor from [2]. For example, a uniformed officer would be more interesting (salient) than the average pedestrian and thus have higher interestingness. Let $\vec{S_I} = \{S_{Obj}, S_{human}\}$, where each element of of $\vec{S_I}$ are separated interestingness terms for objects and humans respectively. The set of weights $\vec{w_I} = \{w_{obj}, w_{human}\}$ gives individual weights for each class of entity. The saliency contribution is then given by $\vec{w_I} \cdot \vec{S_I}$. This concept can be expanded to include more classes of entities.
- *Speed* - This factor is simply calculated as the speed of an object within the simulated world. Faster entities are assigned higher saliency values. Ideally, this would be calculated relative to the observer, however, in practice is unnecessary given the slower speeds at which humans typically move. The saliency term and weight are given by $w_v S_v$.
- *Rotation* - Similar to speed, this factor is based on the angular speed of an entity. One can imagine that a person swinging their arms might draw attention more than someone standing still. The saliency term and weight respectively are given by $w_R S_R$.
- *Orientation* - This factor is observer-dependent, and measures the orientation of an entity with respect to the viewing camera's forward vector. Entities which are facing towards the observer have higher saliency. This primarily applies to humans, but can also apply to things such as signs, vehicles etc. The saliency term and weight are $w_F S_F$.
- *Depth* - Refers to the distance of an entity from the observer. Values are normalized over some maximum range, where closer objects are more salient than further away. The saliency term is given by $w_D S_D$.
- *Semantic Masking* - Humans are much more likely to give attention to certain aspects of an entity, such as the head or face of other humans. This factor takes into account the modulatory effects of such attention and assigns saliency on a sub-mesh level. Terms are given by $w_M S_M$.
- *Attention* - Attention is a top-down effect which modulates the saliency of areas within the human visual field. This factor; given by $w_A S_A$ is implemented as a multiplicative term.

The final saliency value is determined by a weighted combination shown in equation 3.1.1. Bottom-up features are implemented as a weighted sum, including the set of interestingness parameters which are given as a dot product of values and weights vectors. Top-down attentional factors of attention and semantic masking are modulatory in nature, and as such they are multiplicative. All weights for additive features range from $[0, \infty) \in \mathbb{R}$, whereas multiplicative weights are restricted to $[\frac{1}{S}, 1] \in \mathbb{R}$ where S is the saliency value being multiplied. The reason for this is that multiplicative weights affect overall saliency. The lower bound of $\frac{1}{S}$ cancels all effects of the factor. An overall weight W is given to control the total saliency, allowed to be any non-negative value.

$$S = W(\vec{w_I} \cdot \vec{S_I} + w_v S_v + w_R S_R + w_F S_F)(w_M S_M)(w_A S_A) \tag{3.1.1}$$

Weight	$\vec{w_I}$	w_v	w_R	w_F	w_D	w_M	w_A
Bounds	$[0, \infty) \in \mathbb{R}^2$	$[0, \infty) \in \mathbb{R}$	$[0, \infty) \in \mathbb{R}$	$[0, \infty) \in \mathbb{R}$	$[0, \infty) \in \mathbb{R}$	$[\frac{1}{S_M}, 1) \in \mathbb{R}$	$[\frac{1}{S_A}, 1) \in \mathbb{R}$

Table 3.1.1: Bounds for all parametric weights

3.1.2 Implementation

When implemented as a GPU program within a game or simulation engine, this method/model is able to generate saliency maps at high speeds for many agents at once and is very computationally inexpensive. Consider the structure of the graphics pipeline shown in Figure.3.1.1. Materials are often instanced along with individual objects within a rendering loop. For each object, the material passes information to a GPU program. This information typically consists of mesh data, UV data material attributes, as well as other data needed for rendering such as lighting information. In this case, all that is needed is the mesh, attributes and UV data. A GPU program or *shader* consists typically of two components: A vertex shader, and a fragment shader. The mesh data is passed along to the vertex shader, which tells the program where on screen to render, which then processes the information and passes it along to the fragment shader.

The fragment shader can access UV data and attributes to determine what colour to draw pixels to the screen. It is in the fragment shader that the bulk of the implementation resides. The material attributes should contain values for all object-dependent saliency factors described in the *Model* section. For every frame of the simulation, the object's velocity, rotational velocity, forward vector, and semantic UV mask are set in its material. All of these values should be easily readable from a game object's transform and rigid body data. The remaining values; orientation, depth, semantic masking and attention are calculated within the GPU program itself. Orientation is calculated by

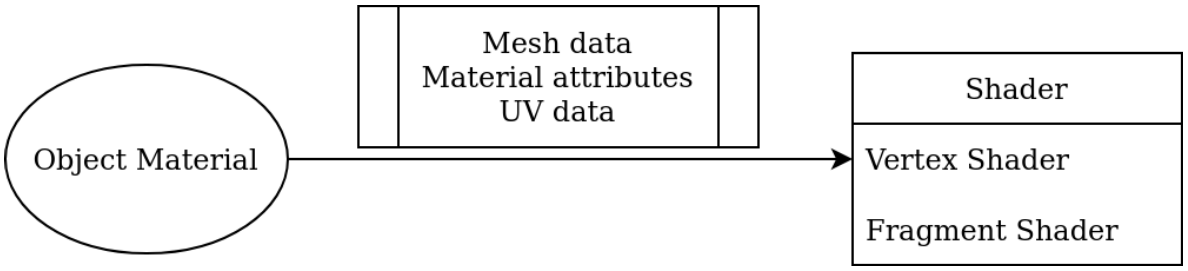


Figure 3.1.1: Diagram showing the flow of data from object materials to a GPU program, known as a *shader*

determining the amount that the object's forward vector $\vec{F}$ aligns with the camera (observer) forward vector $\vec{z}$. The dot-product $\vec{F} \cdot \vec{z}$ (for both normalized vectors) returns a value between -1 and 1, indicating the degree of alignment. There are a few options when deciding how to use this result. One such choice is a binary calculation,

$$S_F = \begin{cases} 1 & \text{if}\, \vec{F} \cdot \vec{z} < 0 \\ 0 & \text{if}\, \vec{F} \cdot \vec{z} \geq 0 \end{cases} \tag{3.1.2}$$

This is functional, however, is equally strong as long as the object is even slightly facing the camera. Of course, $\vec{F} \cdot \vec{z}$ is equivalent to $\cos\theta$, where θ is the relative angle between both vectors, thus an approach which preserves the relative "strength" of the object's orientation is,

$$S_F = \frac{1}{2}\left(1 - \vec{F} \cdot \vec{z}\right) \Rightarrow \frac{1}{2}\left(1 - \cos\theta\right) \tag{3.1.3}$$

This gives a similarly bounded $S_F \in [0, 1]$, where $S_F = 1$ when $\vec{F} = -\vec{z}$, and $S_F = 0$ when $\vec{F} = \vec{z}$. One can go a step further and narrow the profile of the orientation saliency function by squaring Equation 3.1.3. This has the same properties but makes it such that there is a sharper drop-off of saliency as the object faces away from the camera. Figure 3.1.3 illustrates this.

$$S_F = \left(\frac{1}{2}\left(1 - \vec{F} \cdot \vec{z}\right)\right)^2 \tag{3.1.4}$$

To calculate depth, the Z-distance can be extracted from each vertex of the object mesh. The key idea of depth saliency is that objects further away should be less salient than closer objects. So, using the depth value d, the depth saliency is calculated by,

$$S_D = \left(1 - \frac{d}{d_{max}}\right)^2 \tag{3.1.5}$$

where d_{max} is the distance at which saliency drops to zero. Again, the function is squared to emphasize objects that are particularly close. An appropriate value for d_{max} is 10m. Semantic masking requires providing a UV mask, indicating which areas of a mesh are to be highlighted. This kind of masking is a common technique in games and film to apply effects on a particular area of an object. As such, most game and rendering engines allow for supplying a UV mask to a GPU program. Multiplying the saliency value by the mask will modify the values of the texture specified, and leave the rest untouched. In the implementation of PSM used for this work, semantic masking was used to highlight the heads of humans. Similarly, attention is also multiplicative but affects the saliency of the entire frame. Visual attention modulation is represented using 2D attention textures from [30]. The depth of a pixel is read from the camera as well as the horizontal and vertical angle between the pixel (in viewport space) from the viewer. These are then projected onto the XZ and YZ planes. These are finally converted to UV coordinates and sampled from the appropriate visual attention texture. at that location. To approximate horizontal and vertical depth, the final modulatory value is given as the average of the texture in the XZ and YZ planes.

$$S_A = \frac{1}{2}\left(A(x,z) + A(y,z)\right) \tag{3.1.6}$$

These visual attention textures modulate based on the horizontal (or vertical) position of stimuli, but also their depth. By changing textures, one can easily model different types of visual attention such as distraction, thus enhancing or suppressing stimuli based on an attentive state. Figure 3.1.2 shows how textures for Normative and three different types of distraction are represented.

3.2 Parameter Optimization and Validation

The purpose of creating a parametric model for generating saliency maps is to allow for the authoring of a wide range of possible visual attention facets. The parameters described in PSM were chosen to allow for such authoring while simultaneously bounding the range of possible solutions to be psychologically plausible. While these factors (Speed, Rotational speed, Depth, Interestingness, Orientation, Semantic Masking and Attention) were chosen from saliency literature, it is valuable to justify these choices as well as outline methods to determine optimal parameters. Through numerical optimization of

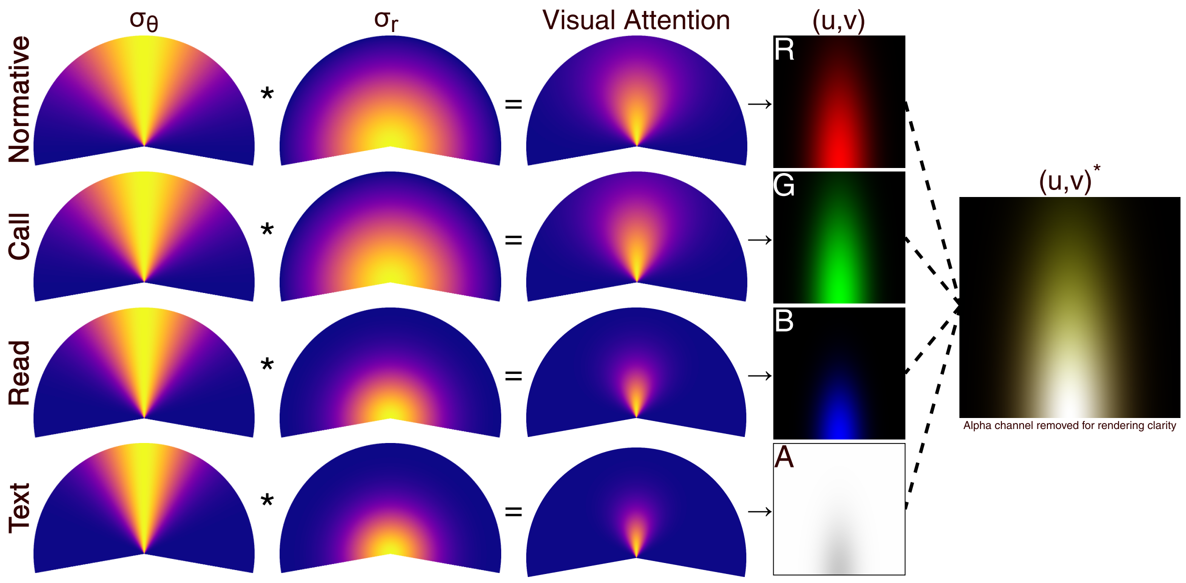


Figure 3.1.2: Visual attention textures for various attentive/distracted states. Each state is encoded in a colour channel (RGBA) representing the strength of attention within an agents visual field. Figure used with permission from [30].

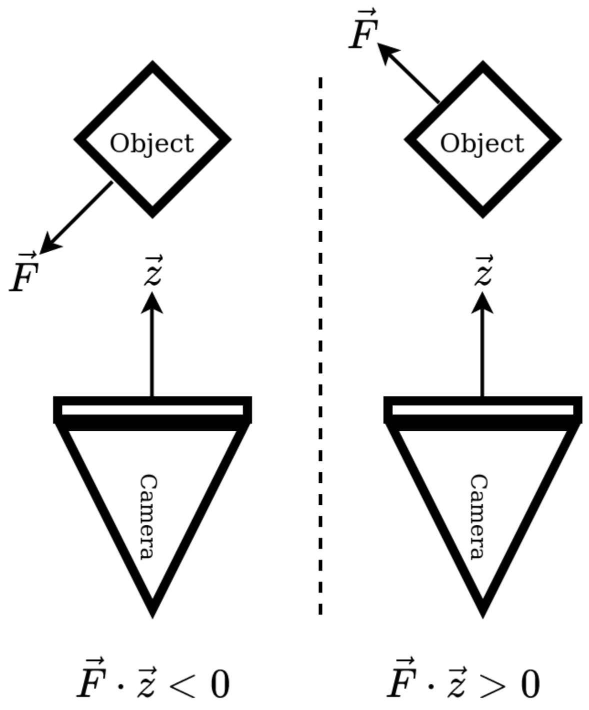


Figure 3.1.3: Top-down illustration of how the orientation of forward vector $\vec{F}$ in the x-z plane affects the dot-product with the camera's $\vec{z}$ direction

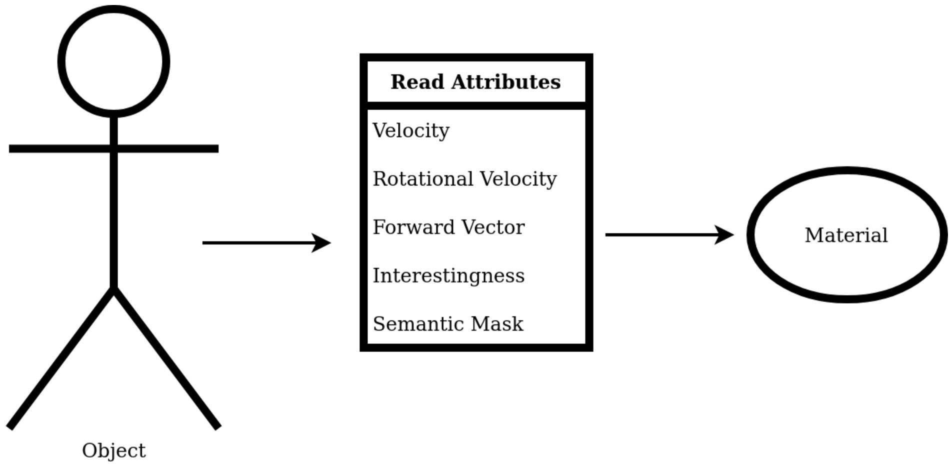


Figure 3.1.4: Per frame, material attributes are read and set in the objects material

parameters, it can be shown that the PSM model is able to generate output highly congruent with state-of-the-art models.

3.2.1 Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

Covariance matrix adaptation evolution strategy [16] is a numerical optimization strategy widely used in science and engineering applications. The strategies it employs are useful for non-convex optimization problems. It works by generating solutions and evaluating them against an error function. From the set of generated solutions, a subset of solutions is then selected to be starting points for the next round of optimization based on their relative error. It can be thought of as an evolutionary algorithm which incrementally improves upon previous generations of solutions. It is a relatively fast and competitive technique for converging to optimized values, as well as able to handle poor objective conditions and noisy data. The algorithm runs until it can no longer improve or after a user-defined fixed number of iterations.

Given some set of parameters W, and an objective function $f_D(W)$ for dataset D the output of the CMA-ES algorithm is given by:

$$W^* = \arg\min_W f_D(W) \tag{3.2.1}$$

Where W^* is the most optimal set of weights which minimizes the objective $f_D(W)$ on set D.

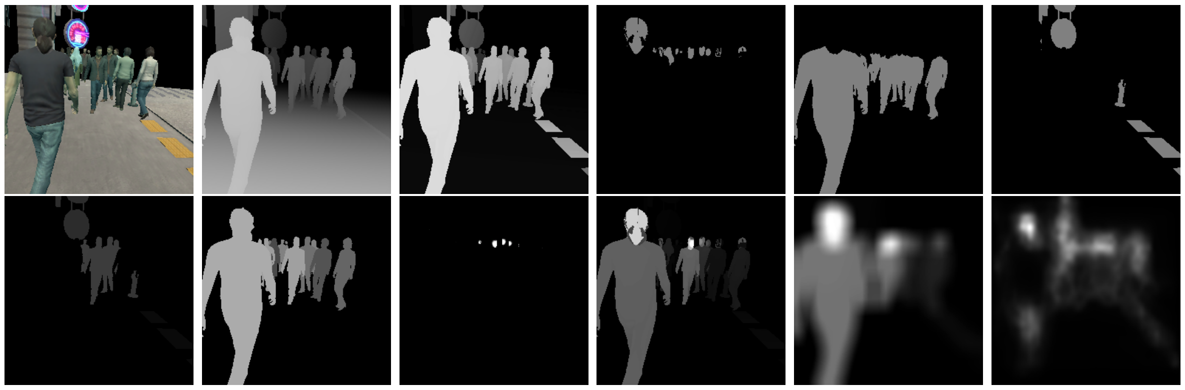


Figure 3.2.1: Decomposition of synthetic dataset example. Top (left to right): RGB input, depth map, attention map, head interestingness map, body interestingness map, object interestingness map. Bottom (left to right): orientation map, velocity map, face mask map, combined PSM, combined PSM output with Gaussian blur, SALICON output

3.2.2 Dataset Generation

In this task, the objective is to generate saliency maps which are similar in structure to those of an established saliency model. For this work, the chosen model to optimize to is SALICON [18]. SALICON is a 'black box' solution in that it is a deep learning model, so to approximate it requires comparing outputs from the same input and updating iteratively. To this end, a synthetic dataset was generated to compare results from PSM to SALICON. Define dataset $D = (x_1, y_1), ...(x_n, y_n)$ such that each pair (x_i, y_i) are the outputs from PSM and SALICON respectively, where each x is a decomposition of all saliency factors into their own 2D maps. Each data pair is generated by considering a single frame in an urban simulation. The normal RGB camera output from the perspective of an agent is used as input for SALICON. From this same frame, a PSM map is extracted as its decomposed elements where each $x_i = S_I * w_I + S_v * w_v + ...$ The result is two co-registered saliency map outputs, but with the ability to adjust the relative weights for each saliency factor in the x_i values.

3.2.3 Optimization Process and Details

For some loss function, given by $L(x, y)$ where x is the model output, and y is the corresponding ground truth, the goal is to minimize the global error over dataset D with regards to this loss. This is the standard formulation of optimization found in machine learning tasks. Often the choice of metric directly impacts the performance of the resulting model once optimized. The effects may be seen in average metric values, as well as in properties of the distribution such as variance. For this task, three choices of loss function were experimented with:

1. *Kullback-Liebler Divergence (KL) Loss*

$$L_{KL}(x, y) = KL(x, y) \tag{3.2.2}$$

2. *Similarity Score (SIM) Loss*

$$L_{SIM}(x, y) = 1 - SIM(x, y) \tag{3.2.3}$$

3. *Weighted KL-SIM Loss*

$$L_{KL-SIM}(x, y) = \alpha \cdot L_{KL}(x, y) + \beta \cdot L_{SIM}(x, y) \tag{3.2.4}$$

Kullbach-Liebler divergence is formulated such that smaller values correspond to more agreement, with 0 meaning both distributions are identical. Similarity score on the other hand is a similarity measure, with 1 corresponding to x and y being identical, and 0 being no overlap. As a result, to be a usable loss function it needs to be inverted, shown in Equation 3.2.3. This is doable because similarity score is a strictly bounded metric, between $[0, 1]$. This is in contrast to Kullbach-Liebler divergence, which is only lower bounded as $[0, \infty)$. As a result, a naive combined loss function would likely bias heavily towards the minimization of Kullbach-Liebler divergence over similarity score. The solution (shown in Equation 3.2.4) is a weighted sum of both loss functions for pairs (x, y), with affine coefficients α and β. For the experiments done in this work, α and β were chosen to be 0.6 and 0.4 respectively.

1. *Average* (AVG)

$$AVG_D(W) = \frac{1}{|D|} \sum_{i=1}^{|D|} L(y_i, x_i(W)) \tag{3.2.5}$$

2. *Root Mean Squared* (RMS)

$$RMS_D(W) = \sqrt{\frac{1}{|D|} \sum_{i=1}^{|D|} L(y_i, x_i(W))^2} \tag{3.2.6}$$

3. *Stochastic Root Mean Squared* (SRMS)

$$SRMS_{D' \sim D^n}(W) = \sqrt{\frac{1}{|D'|} \sum_{i=1}^{|D'|} L(y_i, x_i(W))^2} \tag{3.2.7}$$

The choice of objective function also plays a vital role in achieving an acceptable result when optimizing over a dataset. Let's let $x_i \leftarrow x_i(W)$, where $x_i(W)$ is the output of PSM for the i^{th} set of saliency factors with weights W. Three types of objective function were formulated and experimented with (where objective $f_D(W)$ is the objective f, on dataset D according to weights W): AVG (Equation 3.2.5), RMS (Equation 3.2.6), SRMS (Equation 3.2.7).

Arguably the simplest objective function is to calculate the average loss over an entire dataset. This however has many disadvantages and is not likely to give the best result. The reason for this is that all error values are considered equally. In contrast, a root mean squared objective is more greatly swayed by larger values in the loss function. The effect this should have is that minimization will focus on improving the worst or highest values across the dataset, however, will also be more skewed by outliers. In this case, that is actually a desirable effect seeing as the goal is to achieve acceptable congruence between all ground truth and generated data. To put it another way, it is preferable to achieve a decent lower bound on performance rather than very high performance for some examples, especially considering the limited degrees of freedom in the PSM model, it is likely impossible for most samples to achieve perfect or near-perfect congruency. This approach too has some trade-offs, however. For one thing, calculating over the entire dataset after every update is expensive for any reasonably sized set. Second, it may be difficult for the optimization to first minimize over the entire set at once, and thus very little 'movement' will occur in the sense that it may be unclear which direction (in the space of weights W) to go. Once again, borrowing from machine learning techniques, stochastic descent is often used to improve optimization speeds without sacrificing much if any in performance quality. Instead of calculating error over the entire dataset, a random subset $D' \sim D^n$; with batch size n, is sampled. The batch size is gradually increased throughout optimization until eventually the entire set is being sampled again. The overall effect this approach has is to give the optimizer a 'kick' in the early stages to get it going, and then gradually include larger samples so that results stabilize. This approach is significantly faster at converging, at the cost of some accuracy.

Looking at some example outputs from PSM and SALICON, one issue that may arise is that PSM values are all hard-edged and discrete. A natural remedy to make the comparison more valid is to first normalize both results, and then apply a Gaussian blur to the PSM model output. Again, this technique is common in deep learning saliency literature where ground truth data generated from real human fixations is often blurred when comparing to the output of saliency models. As a part of this, the Gaussian kernel size k is included in the optimization parameters as it is non-obvious what an appropriate kernel size would be and this parameter can greatly affect results. The final set of parameters being optimized is then:

$$W = \{w_d, w_F, w_M, w_v, w_R, w_{obj}, w_{head}, w_{body}, w_A, k\} \tag{3.2.8}$$

3.2.4 Experiments and Results

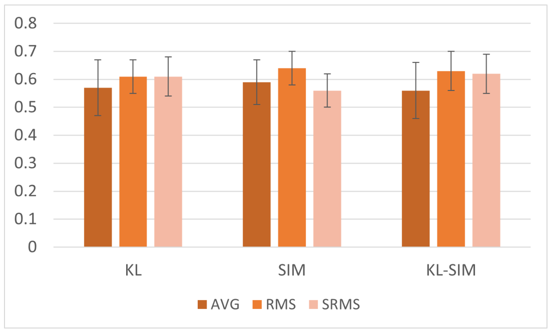
CMA-ES was run for all combinations of objective and loss functions over the synthetic dataset. Multiple attempts were run and the best performing results were recorded, according to similarity score and Kullbach-Liebler divergence metrics. In total, 9 resulting optimized models were compared; labelled by their objective function and loss function: Average w/ Similarity Score Loss, Average w/ Kullback-Liebler Divergence loss, Average w/ KL-SIM loss, Root Mean Squared w/ Similarity score loss, Root Mean Squared w/ Kullback-Liebler Divergence loss, Root Mean Squared w/ KL-SIM loss, Stochastic Root Mean Squared w/ Similarity score loss, Stochastic Root Mean Squared w/ Kullback-Liebler Divergence loss, Stochastic Root Mean Squared w/ KL-SIM loss. Each training attempt ran until CMA-ES termination conditions were met (Δ error less than $\sigma = 1 \times 10^{-9}$, or the max iterations exceeded 1000 (in practice, training attempts rarely exceeded 500 iterations). Upon termination, the result of the optimization is a set of parameters W which can be used to generate an evaluation set Y'.

Loss	Objective	Mean KL	Mean SIM	Best KL	Best SIM
KL	AVG	1.04 ± 0.67	0.57 ± 0.10	0.21	0.76
SIM	AVG	1.48 ± 0.78	0.59 ± 0.08	0.37	0.77
KL-SIM	AVG	1.05 ± 0.64	0.56 ± 0.10	0.22	0.78
KL	RMS	0.84 ± 0.47	0.61 ± 0.06	0.28	0.74
SIM	RMS	1.03 ± 0.60	0.64 ± 0.06	0.24	0.77
KL-SIM	**RMS**	$\mathbf{0.91 \pm 0.65}$	$\mathbf{0.63 \pm 0.07}$	**0.22**	**0.79**
KL	SRMS	1.35 ± 0.75	0.61 ± 0.07	0.26	0.76
SIM	SRMS	1.07 ± 0.65	0.56 ± 0.06	0.42	0.67
KL-SIM	SRMS	0.96 ± 0.75	0.62 ± 0.07	0.23	0.76

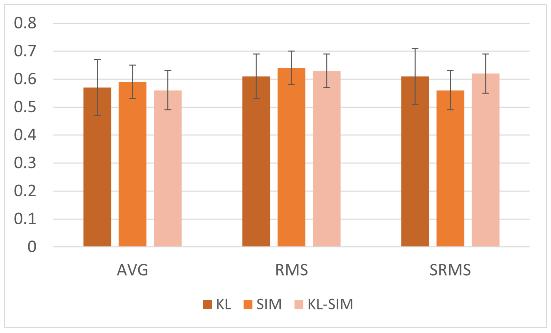
Table 3.2.1: Resulting scores from optimized models obtained using each training objective + loss combination. For interpretation, KL < 1 and SIM > 0.6 are considered good.

Loss	Objective	w_I	w_{body}	w_{head}	w_D	w_M	w_S	w_O	w_A	k
KL	AVG	0.75	0.95	30.03	0.00	3.61	1.10	1.23	0.53	27
SIM	AVG	449.10	176.42	259.92	0.77	34.67	282.38	1.08	12.71	37
KL-SIM	AVG	7.29	0.02	26.06	0.03	24.31	30.88	42.31	0.02	31
KL	RMS	3.33	0.41	9.55	0.01	4.29	0.02	21.70	0.09	25
SIM	RMS	4.14	2.95	0.26	1.35	1.13	2.06	0.51	0.25	37
KL-SIM	RMS	9.40	4.11	9.15	1.17	5.05	14.42	13.81	0.05	25
KL	SRMS	0.47	1.94	1.09	0.01	0.16	0.10	0.82	0.49	81
SIM	SRMS	1.39	0.70	1.07	0.53	0.02	0.87	2.58	0.27	5
KL-SIM	SRMS	0.30	0.17	0.65	0.07	0.12	0.55	0.73	0.95	23

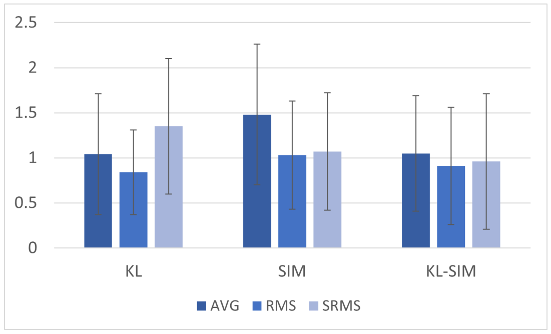
Table 3.2.2: Resulting PSM weights from optimization



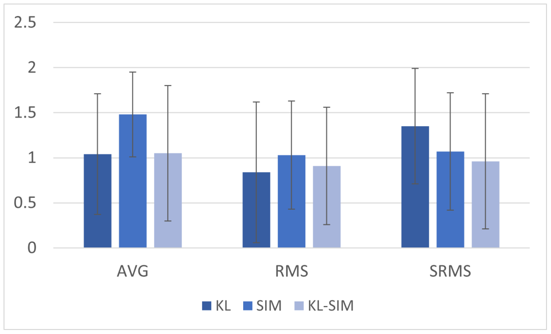
(a)



(b)



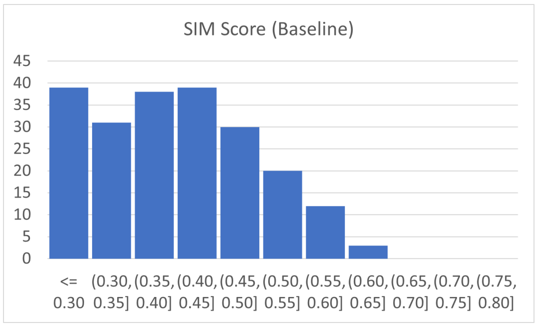
(c)



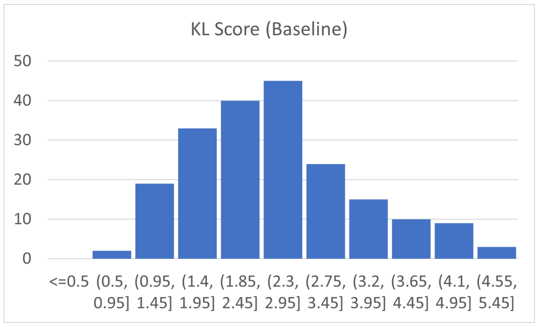
(d)

Figure 3.2.2: Mean scores for SIM (a), (b) and KL (c), (d), grouped by loss (left) and by objective function (right)

Table 3.2.1 displays the average metric scores for Kullback-Liebler Divergence (KL) and similarity score (SIM), as well as the best scores for each metric for each optimized model. This data is plotted in Figure 3.2.2. Comparing the results yields interesting findings in both the optimization parameters and the datasets themselves. Purely based on average performance, the KL-RMS model boasts the smallest mean KL score. However, the mean values for SIM score appear to be more uniform across all models. This, obviously is not enough information to gleam useful insights into the success of optimization, and decide on the best performing set of parameters. Looking at the score distributions for both KL and SIM shown in Figures 3.2.3, 3.2.4, 3.2.5, 3.2.6 (plots for all models can be found in Appendix I), there are clear differences in the shape of the distributions. Ideally, the SIM score distributions should skew towards higher values, and KL score distributions should skew towards lower values. With this in mind, the KL-SIM-RMS and KL-SIM-SRMS models have much more points which have more mass in the desired regions.

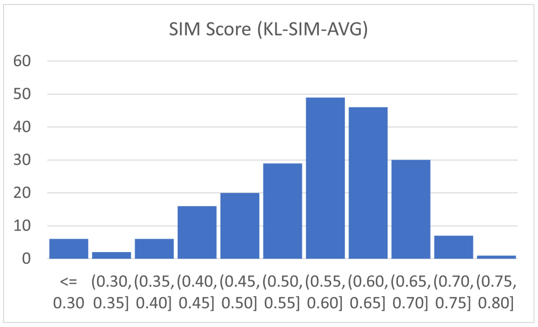


(a)

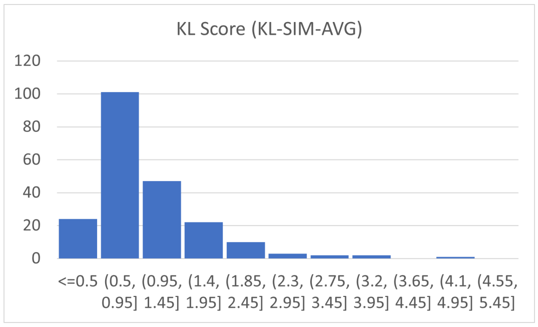


(b)

Figure 3.2.3: Distributions of metric scores for model for unoptimized default parameters, (a) SIM Score (b) Kullbach-Liebler Score

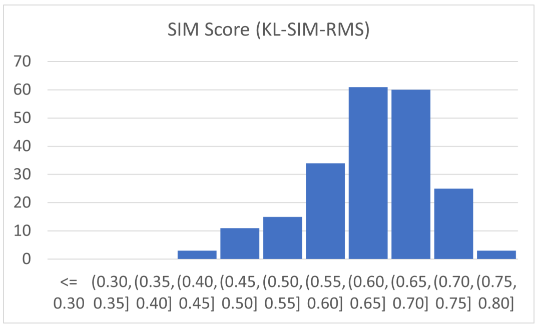


(a)

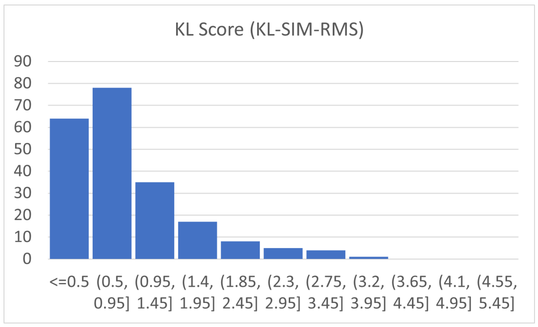


(b)

Figure 3.2.4: Distributions of metric scores for model optimized with KL-SIM loss and AVG objective, (a) SIM Score (b) Kullbach-Liebler Score

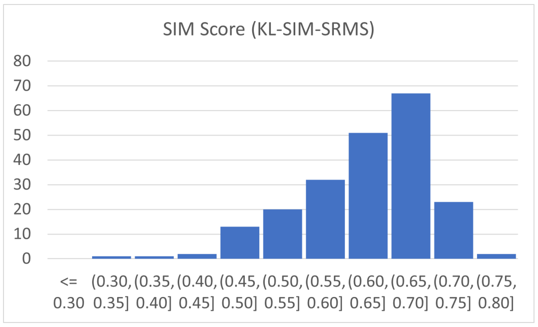


(a)

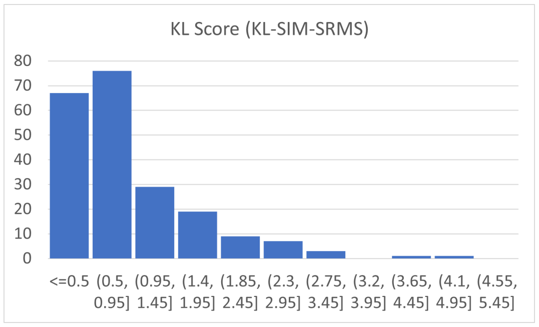


(b)

Figure 3.2.5: Distributions of metric scores for model optimized with KL-SIM loss and RMS objective, (a) SIM Score (b) Kullbach-Liebler Score



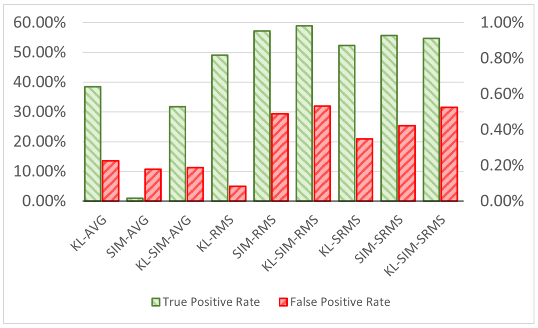
(a)



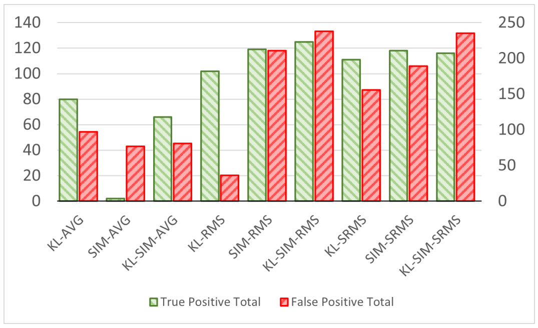
(b)

Figure 3.2.6: Distributions of metric scores for model optimized with KL-SIM loss and SRMS objective, (a) SIM Score (b) Kullbach-Liebler Score

With this information, it can be concluded that optimization attempts succeeded. However, more information is required to decide if the resulting models actually do an acceptable job of approximating SALICON. Over the entire set of PSM/SALICON image pairs, comparisons can be done between all like-pairs and unlike-pairs with the KL and SIM metrics. Using a simple cut-off predictor, the comparisons can be plotted. Figure 3.2.8 shows the result of plotting all pairs which achieve a KL score under 1.00 and SIM score over 0.60. There are two expectations to be had with these plots. First, if the metrics do adequately measure similarity then very few pairs should meet the prior thresh-hold. Second, if the models are optimized then there should be a clear structure to the plots, with distinct diagonals representing true positives, i.e. pairs which are predicted to be the same and actually are. Observing the plots, it can be seen that in the baseline optimized model virtually no pairs are measured to be the same, demonstrating the first expectation to be true. In the case of optimized models, there is clear structure in the predictions with very clear diagonals. Interestingly, the occurrence of false positives has also increased with optimization suggesting that the overall structure of optimized PSM saliency maps more closely match SALICON. Additionally, many false positives seem to be generated from particular examples from either the PSM or SALICON sets, as indicated by the appearance of horizontal or vertical lines in the plot. Comparing prediction rates across all optimized models shown in Figure 3.2.7, it can be seen that false positives in general are relatively low but do tend to scale with increased true positive rates. The best true positive rate comes from the KL-SIM-RMS model at approximately 60% accuracy. The KL-RMS model performs about 10% worse but boasts significantly lower false positives. That being said, as the overall false positive rates are all under 1%, it can be said that these differences are negligible. Considering the limited degrees of freedom in the formulation of PSM and the (at times) unpredictability of deep saliency models, improving past random is impressive. Ultimately the end result of optimization should achieve reasonable separability of like-pairs versus unlike-pairs. Looking at the plots in Figure 3.2.9 this is clearly true. In the case of the KL-SIM-RMS



(a)

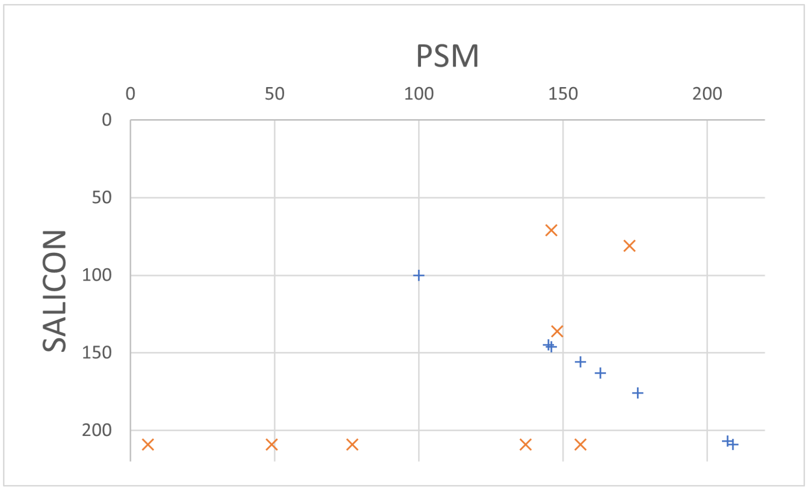


(b)

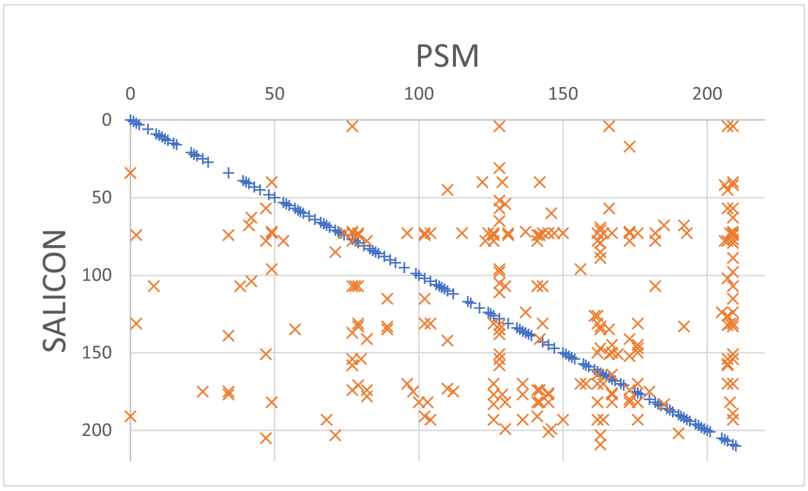
Figure 3.2.7: Plots of true positive (green) and false positive (red) predictions using a KL ≤ 1.00 and SIM ≥ 0.60 predictor. Values are presented as (a) rate, (b) total.

model, there is considerable separation, especially compared to the un-optimized baseline. Intriguingly, even the un-optimized PSM output does achieve some degree of separation as the overall mass is skewed more towards the lower-right region of the KL-SIM space. This speaks to the effectiveness of the chosen saliency parameters that even un-optimized there is some inherent similarity in structure as opposed to random.

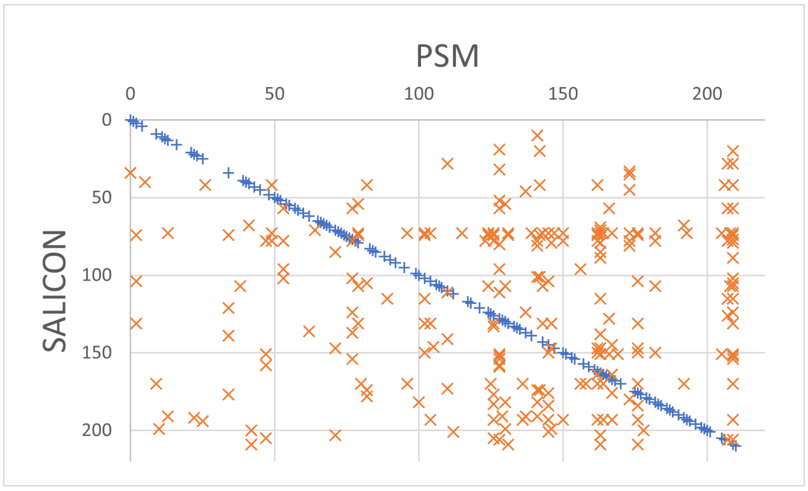
Runtime was also compared with SALICON and DeepGazeII, by measuring the time needed to generate 256×256 pixel images. On average, PSM saliency maps were output in 16.67×10^{-3} seconds, while the average processing time for SALICON was 24.09 seconds and 15.90 seconds for DeepGazeII. All experiments were run in Unity 3D on an i7-7700HQ CPU, GTX 1060 GPU machine with 16GBs of memory. Predictably, PSM is orders of magnitude faster due to being a simple GPU program



(a)

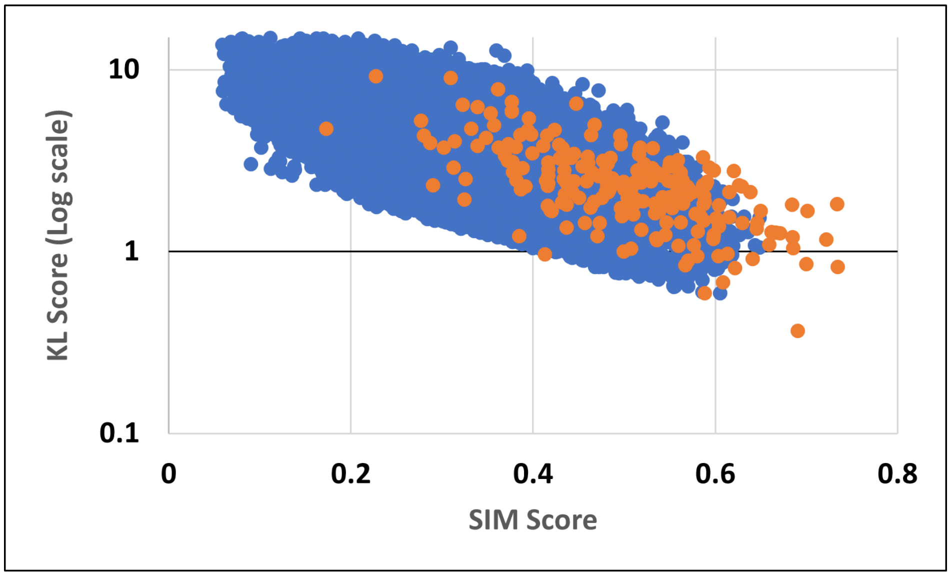


(b)

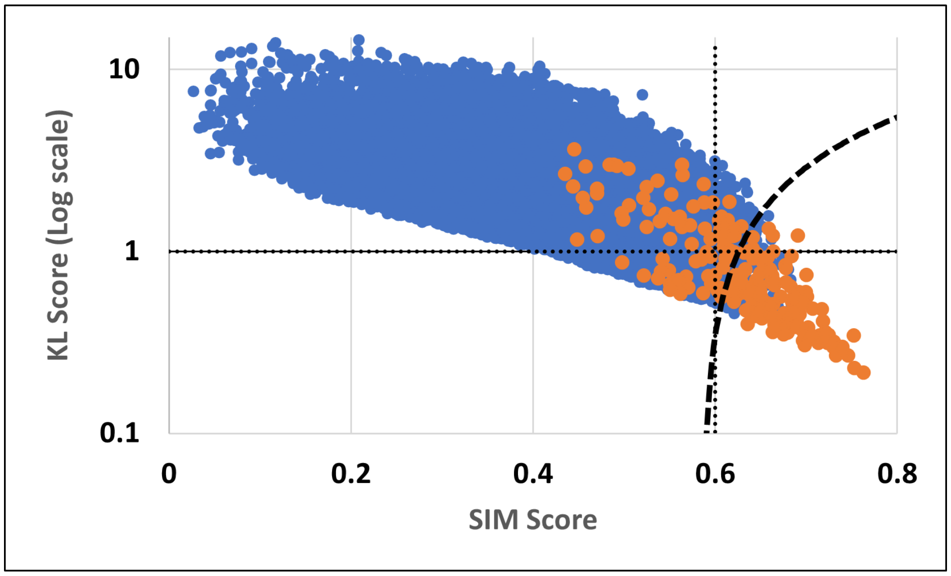


(c)

Figure 3.2.8: Prediction charts for (a) baseline, (b) KL-SIM-RMS, and (c) KL-SIM-SRMS models. All 212 SALICON and PSM images are compared (212^2 comparisons) with both SIM and KL scores. A simple predictor evaluates a pair to be the same if it has a KL score ≤ 1.00 and SIM score ≥ 0.60. Points which are succesful are plotted, with true-positives lying along the diagonal.



(a)



(b)

Figure 3.2.9: Scatter plots of all 212 PSM/SALICON pairs compared, with like-pairs represented in orange, unlike pairs are blue, in the (a) unoptimized case, (b) optimized PSM model from KL-SIM-RMS. In the optimized case, the thick dashed line represents an optimal separating hyperplane found with a support vector machine (SVM) algorithm. The thin dotted lines are placed at KL = 1 and SIM = 0.6.

3.2.5 Conclusions

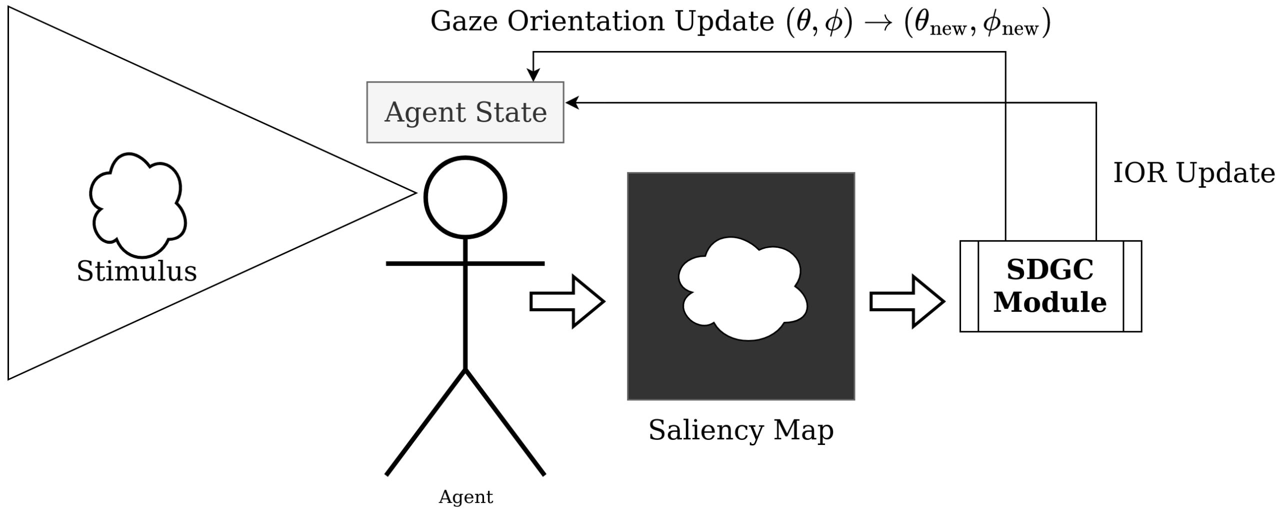
Parametric Saliency Maps are an effective way to approximate the outputs of machine learning trained saliency models. It was shown that the weights of the saliency parameters can be optimized to approximate state-of-the-art models in saliency to a high degree of similarity despite consisting of only 9 degrees of freedom. This may also be a lens with which to analyze existing saliency models based on deep networks. Perhaps this offers some explainability to the features and factors the networks are selecting for when generating saliency maps. Since all saliency factors in PSM are derived from the psychological literature on attention, and deep learning models are trained on real human data, this yields credit that these deep learning models are in fact learning to reflect real ground truth human cognitive processes.

CHAPTER 4

SALIENCY DRIVEN GAZE CONTROL

Ultimately, the purpose of generating real-time pseudo-saliency maps is to use them to drive systems in a virtual agent. Modelling and simulating human gaze is a complicated endeavour. There are numerous approaches and methods for estimating how a human agent may observe the world. The majority of approaches aim to recreate plausible animated humans. These methods can be very effective, often taking advantage of scene information from the simulation to calculate believable gaze patterns. However, if the goal is to replicate gaze from first principles, i.e. replicating the biological and psychological mechanisms behind human gaze, then it is paramount to adhere as much as possible to a principle of 'sensory honesty' as noted by Peters [46]. To put it a different way, devise methods to drive gaze without using any information or data that a real human agent would not have. The work presented here attempts to work within this constraint by relying primarily on visual input and internal attention modelling to drive gaze. Specifically, the focus is on utilizing saliency maps as input representing pre-attentive processing of visual stimulus by the human psycho-visual system. There are many factors and complications to consider when presenting a model of human gaze. As such this work constructs a framework for authoring a variety of gaze behaviours driven entirely by saliency map information.

Figure 4.0.1 outlines the Saliency Driven Gaze Control (SDGC) framework. The following



Gaze Orientation Update $(\theta, \phi) \rightarrow (\theta_{\text{new}}, \phi_{\text{new}})$
Agent State
IOR Update
Stimulus
SDGC Module
Saliency Map
Agent

Figure 4.0.1: High-level conceptual diagram of a full SDGC approach. For every time t, a saliency map corresponding to the current agent view is generated. This is passed to an SDGC module which processes the saliency map and generates an updated gaze orientation represented by (θ, ϕ). Updates to an agent's internal attention model are output, as well as a function of an inhibition of return (IOR) mechanism.

sections propose implementations for SDGC modules which process saliency data and then produce an update for the gaze orientation of the viewing agent. Additionally, updates to the agent's internal attention are applied in the form of inhibition of return (IOR) updates.

4.1 Spline Surfaces as Height Fields

Implicit curves have a long history of use within the field of computer graphics and animation. These curves often referred to as *splines* allow for infinite resolution and simple approximation of much more complex shapes by stitching together polynomial functions bound by a set of control points. Many such spline paradigms exist, such as Hermite splines, Catmull-Rom splines (named for famous Pixar co-founder Edwin Catmull and Raphael Rom), Bezier splines, B-splines and many more. Splines are often used in interpolating problems because they are predictable and smooth. They avoid the issue of using higher-degree polynomials to express a series of control points (otherwise known as Runge's Phenomenon) which tend to cause unwanted oscillations. This type of method works in any number of dimensions, able to create curves, surfaces and even volumes. This section will focus on the use of B-splines for generating 3D surfaces representing height maps, and the challenges associated with them. Specifically, the following questions are addressed: how are B-splines defined? How is a surface generated? How do control points affect the properties of the spline? The final result will be a spline surface which approximates a function $z(x, y)$.

A B-Spline of order p, with control points $\{P_0, ...P_n\}$ can be defined by the function,

$$\vec{S}(u) = \sum_{i=0}^{n} N_{i,p}(u)\vec{P_i} = \langle \mathbf{x}(u), \mathbf{y}(u) \rangle \tag{4.1.1}$$

Where $N_{i,p}(u)$ are basis functions used to control blending between points, based on parametric value u. These functions are defined recursively by the following set of expressions,

$$N_{i,0} = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{else} \end{cases} \tag{4.1.2}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{4.1.3}$$

For a set of $n+1$ control points, there are an equivalent number of basis functions with degree p. The values u_i are drawn from the *knot vector* $u_i \in \vec{U}$. The knot vector controls where and how control points affect the resulting curve. For a knot vector to be valid, $u_i <= u_{i+1}$ for all i values in $\vec{U}$. A knot vector is said to be uniform if there is equal spacing between all adjacent values in $\vec{U}$, for example, $\{0, 1, 2, 3, 4, 5\}$. A knot vector is said to be standard if there are $p + 1$ repeating knots on either end and equal spacing otherwise. For example, $\{0, 0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1, 1\}$ is the standard knot vector for a degree 3 spline, with 7 control points. Standard knot vectors have a useful property that the resulting spline starts and ends exactly on the first and last control points respectively.

An implicit surface can then be created via a tensor product of two B-Spline curves for $n \times m$ control points which takes two parametric values u, v. With control points in 3D space, the resulting function $\vec{M}(u, v)$ returns a 3D point value for each (u, v).

$$\vec{M}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,p}(v) \vec{P}_{i,j} = \langle \mathbf{x}(u, v), \mathbf{y}(u, v), \mathbf{z}(u, v) \rangle \tag{4.1.4}$$

Since the dimensions are independent, another representation is as a vector of 3 one dimensional spline functions, $\mathbf{x}(u, v), \mathbf{y}(u, v), \mathbf{z}(u, v)$. If one wishes to use the surface as a height field, this must also be able to be formulated in such a way that $\mathbf{z}(u, v) = \mathbf{z(x,y)}$. Consider the more simple case of a B-spline curve constructed from 2D points shown in Figure 4.1.1. Suppose one wished to find the gradient (or derivative in this case) of y with respect to x. This can be expressed as:

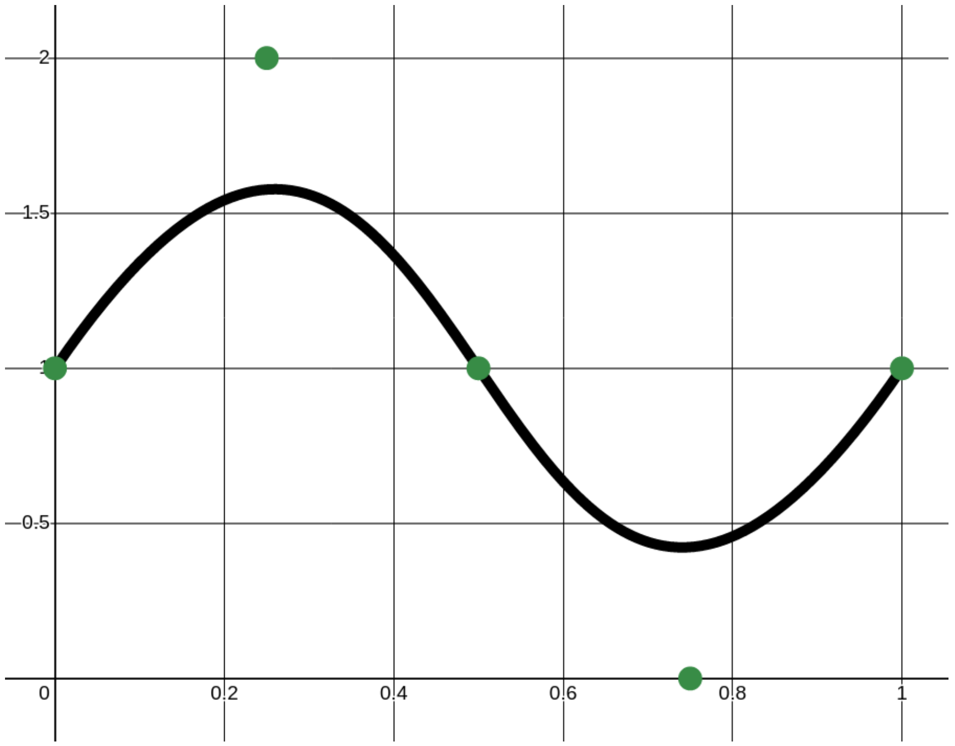


Figure 4.1.1: Cubic B-spline with 5 control points using a standard knot vector 0,0,0,0,0.5,1,1,1,1

$$\frac{dy(u)}{dx} = \frac{dy}{du}\,\frac{du}{dx} \tag{4.1.5}$$

Naturally, if looking for an analytical solution then expressions for $\frac{dy}{du}$ and $\frac{du}{dx}$ need to be found. However, the problem is that B-spline basis functions are not easily invertible. (From [50]) The first order derivative of a B-Spline can be calculated as:

$$\begin{gathered} \frac{d}{du}S(u) = \sum_{i=0}^{n} \frac{d}{du}N_{i,p}(u)P_i \\ \frac{d}{du}N_{i,p}(u) = N_{i,p}^{'}(u) = \frac{p}{u_{i+p}-u_i}N_{i,p-1}(u) - \frac{p}{u_{i+p+1}-u_{i+1}}N_{i+1,p-1}(u) \end{gathered} \tag{4.1.6}$$

The wonderful thing about this expression is that the derivative of the basis function is itself another basis function. What is still missing however is an expression for $\frac{du}{dx}$. While clearly $\frac{dx}{du}$ is virtually identical to $\frac{dy}{du}$, this expression is **not** invertible. In fact, there is no analytical solution to invert a B-spline basis function. The only way this function is solvable if and only if $x(u) = u$, thus $\frac{du}{dx} = \frac{dx}{du} = 1$. It would appear now that the solution to this conundrum is to formulate the spline such that $x(u) = u$ (and $y(v) = v$ in the case of a surface).

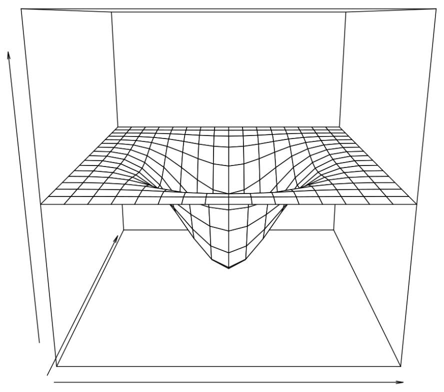


Figure 4.1.2: Generated spline height field

A height field is typically represented in the form of a scalar-valued function $z(x, y)$, where x and y are independent parameters for example, $z(x, y) = x^2 + y^2$. Similarly, B-splines can be used to construct a surface using two independent parametric values u and v; shown in Equation 4.1.4. The x and y values of the control points must be strictly increasing and linear to be a valid singly-valued function. The gradient of a B-spline surface can be calculated as the tensor product of the surface $M(u, v)$ with the gradient operator with respect to u and v.

$$\begin{aligned}\nabla \otimes M(u,v) &= [\frac{\partial}{\partial u}M(u,v), \frac{\partial}{\partial v}M(u,v)]\\ \frac{\partial}{\partial u}M(u,v) &= \sum_{i=0}^{n}\sum_{j=0}^{m} N'_{i,p}(u)N_{j,p}(v)P_{i,j}\\ \frac{\partial}{\partial v}M(u,v) &= \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N'_{j,p}(v)P_{i,j}\end{aligned}$$

(4.1.7)

$\nabla\mathbf{z}(u, v) = (\frac{\partial M}{\partial u}, \frac{\partial M}{\partial v})$ is a vector which represents the direction of steepest ascent of $\mathbf{z}$ in the u-v plane. Let the matrix of control points with dimensions $n \times m$ be given by $(z_{i,j}) \sim \mathbf{P}$.

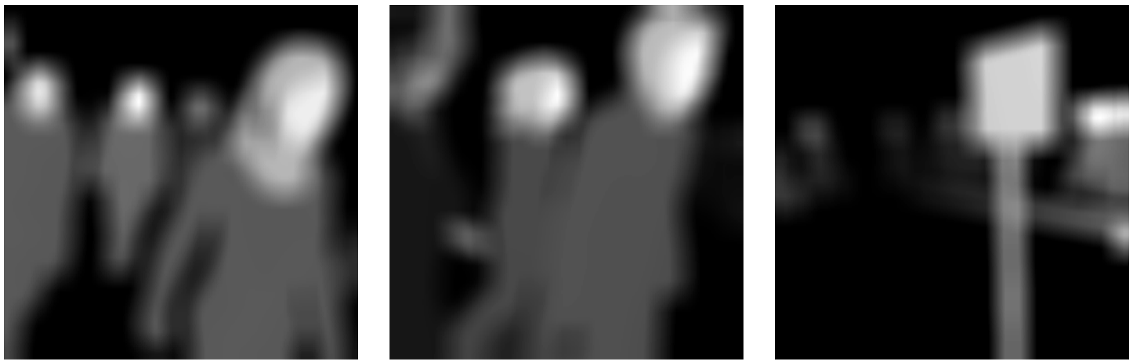


Figure 4.2.1: Examples of generated saliency maps from the perspective of an agent walking through a simulated urban crowd, using PSM weights specified in Section 3.2.

$$\mathbf{P} = \begin{pmatrix} (z_{0,0}) & (z_{1,0}) & \cdots \\ \vdots & \ddots & \\ (z_{0,m}) & & (z_{n,m}) \end{pmatrix} \tag{4.1.8}$$

This works because the 3D spline is a combination of two independent splines in the u and v dimensions. The final formulation is a spline surface where $z(u, v) = z(x, y)$. The result allows for a simple, fast and analytic gradient to be calculated at any point in the spline surface; given by $\vec{\nabla} z = (\frac{\partial x}{\partial u}, \frac{\partial y}{\partial v})$, since it is now treated as a height field $z(x, y)$.

4.2 Particle Gaze Model

When conceptualizing how a saliency map should influence gaze behaviour, the natural assumption is that gaze is attracted towards objects or areas of high saliency, seeing as a saliency map describes the probability of fixations for any given point in the visual field. In this way, the problem could be modelled as a physical system, where a “particle” is attracted towards areas of high saliency, much like how a charged particle moves within an electric field from areas of high electric potential to low electric potential. A saliency map can represent a potential field, where a gaze “particle” moves according to Newtonian equations of motion. Such a simulation is quite simple and easily tuned to result in various behaviours. Consider the center of an agent’s visual field, corresponding to a point in the 3D viewing sphere around an agent (or agent’s head). In this case, moving the point around the sphere results in changing where the agent is currently looking. By treating this point as a particle and applying ’forces’ to it, a mode of gaze can be approximated.

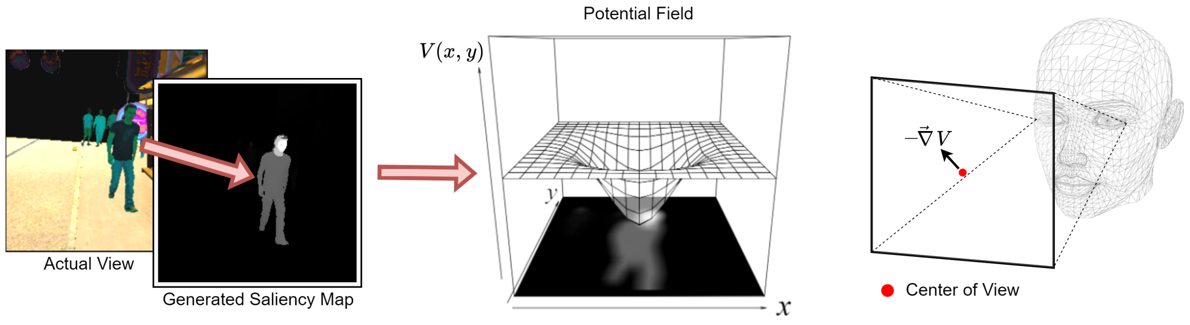


Figure 4.2.2: High-level diagram of particle model. A saliency map is generated from the current view. It is then interpreted as a continuous potential field. The gradient at the center of this field is measured as used to update the gaze of the agent.

Let $\vec{x}$ represent the position of a particle, moving in a potential field $V(\vec{x})$. The differential equation describing the particle's movement is given as,

$$\ddot{\vec{x}} = -\vec{\nabla}V(\vec{x}) \tag{4.2.1}$$

This equation describes a particle which will accelerate towards regions of lower potential energy. It is useful to additionally include a damping term in the equation, for the purposes of imposing a speed limit and controlling the dynamics of the system. The equation of motion is then,

$$\ddot{\vec{x}} = -\vec{\nabla}V(\vec{x}) - k_d \cdot \dot{\vec{x}} \tag{4.2.2}$$

where k_d is the damping coefficient. The result is that the particle will minimize its potential and settle into potential wells, similar to that of a gradient descent algorithm. Consider an agent's gaze, which lies on a unit sphere around an agent's head, described by position $\vec{G} = (\theta, \phi)$. Shown in Figure 4.2.3, converting the 2D point $\vec{x}$ in viewport coordinates to a 3D point $\vec{\mathbf{X}} = (x, y, z)$ in world coordinates, a vector can be drawn from the camera origin to $\vec{\mathbf{X}}$. The gaze orientation $G = (\theta, \phi)$ is updated by values $(\Delta\theta, \Delta\phi)$:

$$\begin{aligned} \Delta\phi &\simeq \Delta x \\ \Delta\theta &\simeq \Delta y \\ \vec{G} &\Rightarrow \vec{G} + (\Delta\theta, \Delta\phi) \end{aligned} \tag{4.2.3}$$

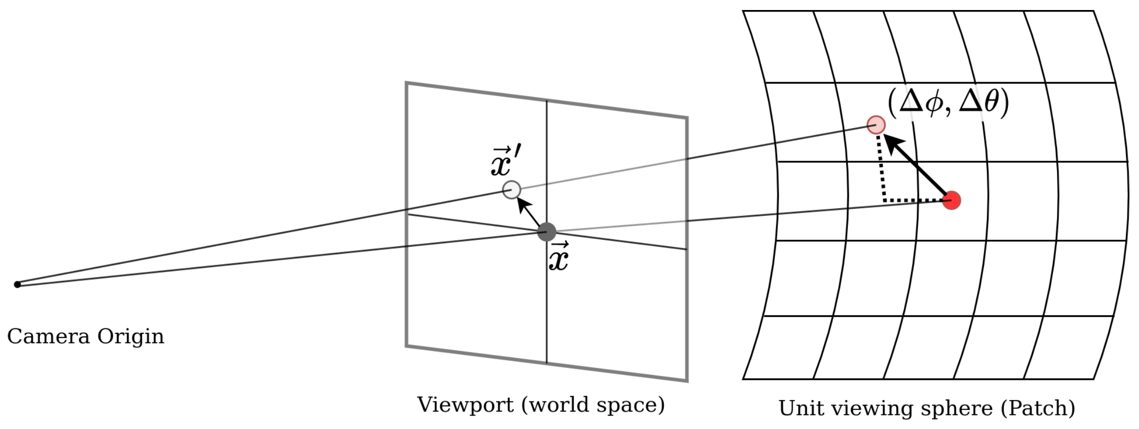


Figure 4.2.3: Projection of point positions $\vec{x}$ to $\vec{x}'$ in viewport space to a change in orientation $(\Delta\phi, \Delta\theta)$

For sufficiently small steps, the simplification can be used that the Cartesian $\Delta x, \Delta y$ coordinate updates are equivalent to their spherical counterparts (small angle approximation). $\vec{\nabla} V$ is the force applied to the gaze particle based on what the agent is currently seeing, S, and λ is the step size. The algorithm can be written out using Euler integration,

$$\begin{aligned} \ddot{G}_{t+1} &= -\vec{\nabla} V - k_d \dot{G}_t \\ \dot{G}_{t+1} &\Rightarrow \dot{G}_t + \lambda \ddot{G}_{t+1} \\ G_{t+1} &\Rightarrow G_t + \lambda \dot{G}_{t+1} \end{aligned} \tag{4.2.4}$$

Additionally, a noise term $A \cdot z_t$ can be included; where $z_t \in [-1, 1]^2$ with amplitude A, in the final position update which gives added flexibility to model more complex gaze movement characteristics. The final update is then,

$$G_{t+1} \Rightarrow G_t + \lambda \dot{G}_{t+1} + A \cdot z_t \tag{4.2.5}$$

An important consideration is then how to construct the potential field. By treating a saliency map as a potential, where high saliency areas are potential wells in the field, then following the gradient will draw the particle into these salient regions. Some possible issues arise concerning the computation of this potential from an agent's saliency map. Looking at examples in Figure 4.2.1, one problem is that in most saliency maps there are large regions of little to no saliency. This presents a problem because there would be no gradient in these regions. Yet one more consideration is that highly salient stimuli should draw gaze towards it regardless of where it is in the visual field. Calculating the potential field

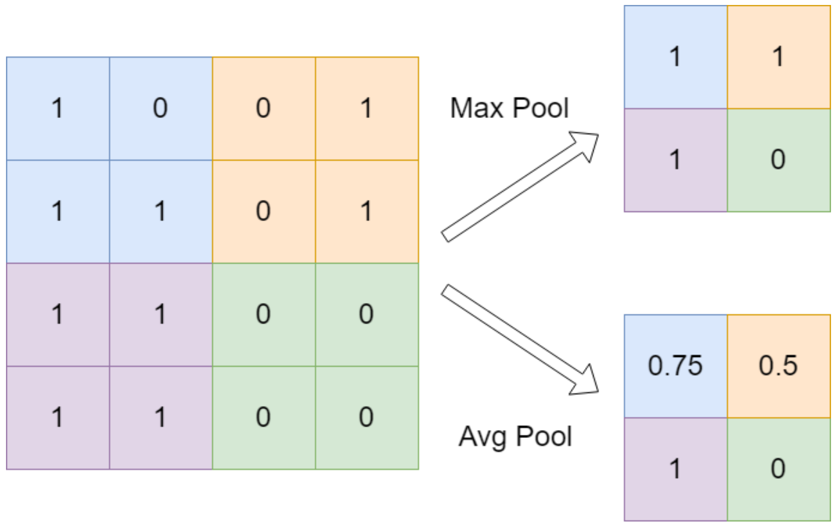


Figure 4.2.4: Max-pooling vs. Avg-pooling, sampling down a 4×4 image to 2×2

from an $n \times n$ image could be computationally very costly, especially scaled to scenarios with many agents. The solution to both of these problems is to use a parametric surface to model the potential by sampling from the image. Cubic B-Splines have useful properties which make them very efficient for this task. Using a cubic B-spline height field, we can then sample from the image to derive the z-coordinates (height) of each point. Assuming an appropriately chosen number of control points, a cubic B-spline height field will have a non-zero gradient in almost all regions of space, on top of being very fast to compute.

Drawing from computer vision, two appropriate methods for pooling values from a saliency map into control points are the max-pool and average-pool algorithms. As shown in figure 4.2.4, for a given window of size $n \times m$, an image can be down-scaled by taking the maximum value within the window (max-pool) or averaging all the values (avg-pool). The result of this operation gives a smaller set of values based on the number of windows. This method can be used to assign heights to control points on our surface by pooling values from the saliency map. In this case, control points are set to the negative value from pooling to create a well in the potential. One consideration is that for a B-spline height field, control points are not equidistant in the u-v plane parametric plane. The equivalent x-y positions can be found at the *Greville Abscissae*, which is discussed in more detail in Appendix II. The resulting positions show the non-linear spacing that affects the control point influence regions. Pooling kernel sizes should not be equally sized as we want the influence of points to correspond with the surface's control net. The scheme shown in figure 4.2.5 is how kernel sizes are chosen in accordance with virtual control point locations. The scheme is such that each edge of the kernel is halfway between adjacent points for the x and y directions.

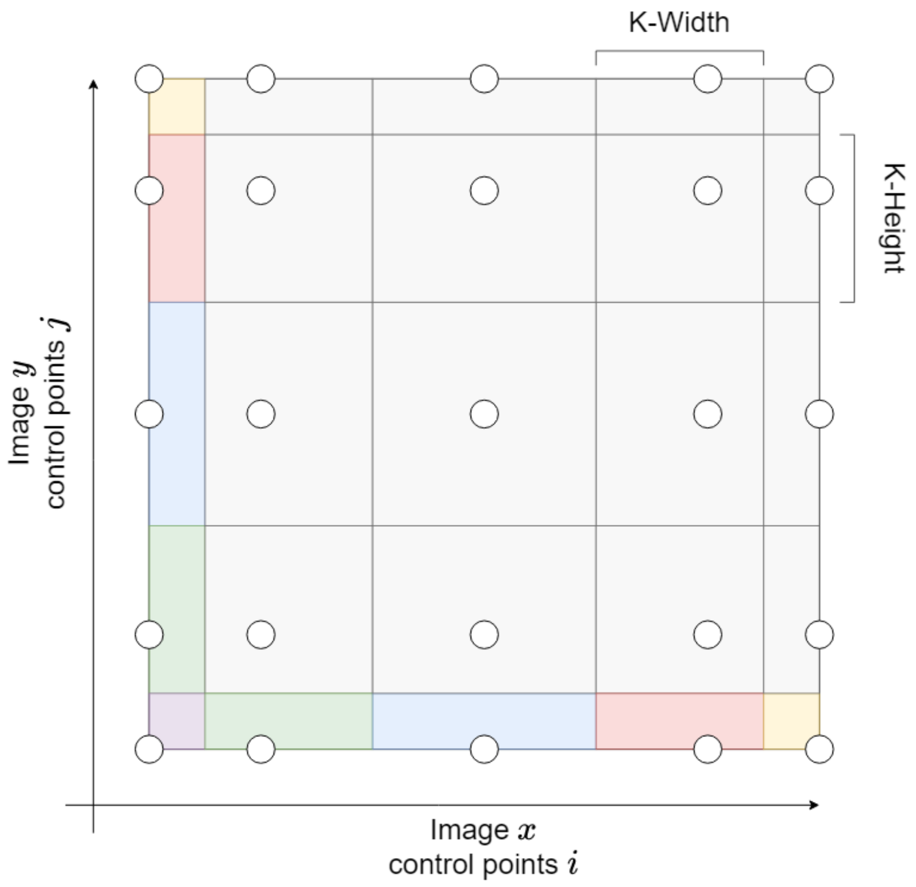


Figure 4.2.5: Illustrating the pooling windows for a set of control points. Each window corresponds to the control point within it

At each time t the control points of a B-Surface are set, giving the potential field. Since the gaze particle is always at the center of the visual field, The potential is sampled at parametric coordinates $(0.5, 0.5)$. $\vec{\nabla} V$ is then the gradient of the surface at that point, which is given by equation. 4.1.7. Of course, one could numerically solve for the gradient, however, this is computationally expensive. A useful property of a parametric surface is that it is continuous everywhere. It allows there to be gradient flow in nearly all locations of the surface, given appropriately chosen dimensions for the surface. Too few points and there is not enough locality to the effects of the saliency map on the field. Too many however and the surface will lose the property of having points influence most of the entire field. One such option for increasing the granularity of the control points is to include some localized area of effect when setting control point depths. A neighbour weighting effect can be achieved by adding a weighted influence of surrounding points on each given point. Again this must be modulated accordingly otherwise loss in locality will occur.

4.2.1 Emulating Gaze Movements

There are properties of the particle model which lend themselves well to controlling head movements, as well as smooth-pursuit eye movements. First, is the naturally smooth motion which arises towards targets of high interest. Second, for a small number of control points; recommended 7 for a degree 3 spline surface, this method has the natural tendency to align with general areas of high interest at low resolution. This often means looking at the "center of mass" of areas with high saliency targets as opposed to specific individual elements if there are many within view. If there are sparse, spaced-out objects of interest the gaze will instead align with the individual elements. Both these behaviours arise without explicit programming. Setting the points in the control surface to a higher resolution will yield more spacial acuity, and thus the gaze will fall on more narrow targets. Changing the step size λ will determine how fast the gaze will move towards targets, as well as how strongly those targets will be tracked. Smooth pursuit eye movements can be elicited by having a high resolution in the control surface; recommended 11 for a degree 3 spline surface, and a larger $\lambda_{fixation}$ value. It is difficult to recommend any particular value for $\lambda_{fixation}$ because this will be scaled with how the spline surface is defined, how steep peaks are, as well as how fast objects move across the field of view which is limited by the frame rate of a given simulation. The length of smooth pursuits is something contextual. For a typical "search" behaviour, the length of fixations $\tau_{fixation}$ should average $150 - 300$ ms. For saccadic movements, a larger λ_{search} value will give faster rapid target acquisition. To emulate micro-saccades we can perturb the final position using a noise term $A \cdot z_t$, where the amplitude corresponds to less than $0.1°$ of visual angle. This will depend on camera projection parameters, but a small angle approximation $A \simeq 0.1^\circ$ is acceptable. Additionally, to improve accuracy and avoid oscillations, multiple steps can be taken per simulation time step. In the scope of this work, we do not describe how to switch between saccades and smooth pursuits. This is largely because smooth pursuits are typically *intentional* actions and need to be specified by the author of the behaviour.

4.2.2 control

Algorithm 1 Particle Gaze Model

```
$STATE \leftarrow search$
$\vec{x} \leftarrow (0.5, 0.5)$                                  ▷ Center of viewport
$\dot{\vec{x}} \leftarrow (0, 0)$
while $true$ do
    $V \leftarrow \text{SetPotential}(S_t)$
    $\ddot{\vec{x}} = -\vec{\nabla} V(\vec{x}) - k_d \cdot \dot{\vec{x}}$
    if $STATE == search$ then
        $\lambda \leftarrow \lambda_{search}$
        if $FixationDetected()$ then
            $STATE \leftarrow fixation$
        end if
    else if $STATE == fixation$ then
        $\lambda \leftarrow \lambda_{fixation}$
        if $fixationtime > \tau_{fixation}$ then
            $STATE \leftarrow search$
        end if
    end if
    $\dot{\vec{x}} \leftarrow \dot{\vec{x}} + \lambda \cdot \ddot{\vec{x}}$
    $\vec{x} \leftarrow \vec{x} + \lambda \cdot \dot{\vec{x}}$
end while
```

4.3 Probabilistic Gaze Model

In this section, another method is introduced for saliency-driven gaze control, based largely on prior works in fixation prediction for static images. A saliency map can be thought of as a probability distribution for likely gaze targets. With this interpretation, fixation targets can be sampled from this distribution. For a probability distribution S_t, a random point $\mathbf{x} \sim S_t$ is drawn. Based on the projection parameters of the virtual camera, this point in the viewing image can be converted to an orientation (see Figure 4.2.3). The agent's view can then be rotated accordingly to match this orientation.

Given a point $\mathbf{x}$ S_t in viewport coordinates, a line can be drawn from the camera center through this point in world space. This vector represents an orientation G'. The current camera orientation G can then be interpolated to this new orientation over the desired time. The speed of the rotation is then determined by the interpolation time.

Control is divided into two primary states: search and fixation. In the search state, a point is sampled from the entire field of view. The view is then oriented to this target over $\Delta\, t_{saccade}$. The angular speed of the saccade is the amplitude (angular) divided by $\Delta t_{saccade}$. Once this target is picked the state transitions to fixation control. Over a total time $\tau_{fixation}$ saliency outside a small foveated region of radius R_{focus} is suppressed. Within this fixation, new points are drawn from the foveated region of interest as targets for micro-fixations. The point is then interpolated to over $\Delta t_{\mu\, saccade}$. This point is looked at for time $\tau_{\mu\, fixation}$, at which point a new target is selected. This repeats over the entire fixation length. Once the fixation has concluded, the state returns to search. Each parameter can be set statically or dynamically depending on desired behaviours.

4.3.1 Emulating Gaze Movements

This method of control is designed to allow modelling of target point selection saccade and micro-saccade eye movements. Depending on the level of detail desired, keeping $\Delta\, t_{saccade}$ and $\Delta\, t_{\mu saccade}$ constant will achieve linear eye velocities expected for angular distances less than $20°$, which typically reach up to $300°/s$. However, for most applications, it suffices to have a very small or zero travel time (i.e. instantaneous). Changing the $\tau_{fixation}$ parameter will affect how much searching is done in the visual field. Veering from typical reported values of around $100 - 200ms$ will result in either rapid eye-darting for smaller values, or more focused eye movements in the case of larger values. Tightening or increasing the size of the focus region R_{focus} will either restrict the space of micro-saccade movements (thus decreasing their amplitude) or allow for more outside stimuli to draw micro-saccades respectively. Depending on the desired behaviour either can be appropriate. For example, a character reading a book would have very infrequent saccades (large or infinite $\tau_{fixation}$), frequent micro-saccades (small $\tau_{\mu fixation}$, and a small radius of focus R_{focus}. Similarly to the particle method, inhibition of return is implemented as a decay in object saliency.

Algorithm 2 Probabilistic Gaze Model

```
**Def:** $LookAt(point, time)$
$STATE \leftarrow search$
$G \leftarrow (0.5, 0.5)$                                        ▷ Center of viewport
**while** $true$ **do**
    **if** $STATE == search$ **then**
        $\mathbf{x} \leftarrow SamplePoint(S_t)$
        $LookAt(\mathbf{x}, \Delta\, t_{saccade})$
        $STATE \leftarrow fixation$                              ▷ Wait until reached target
    **else if** $STATE == fixation$ **then**
        $S_W \leftarrow S_t.window(R_{focus})$
        $\mathbf{x} \leftarrow SamplePoint(S_W)$
        **if** $fixationTime > \tau_{fixation}$ **then**
            $STATE \leftarrow search$
        **else**
            $LookAt(\mathbf{x}, \Delta\, t_{\mu\ saccade})$
            $Wait(\tau_{\mu\ fixation})$                         ▷ Hold for length of $\mu$-fixation
        **end if**
    **end if**
**end while**
```

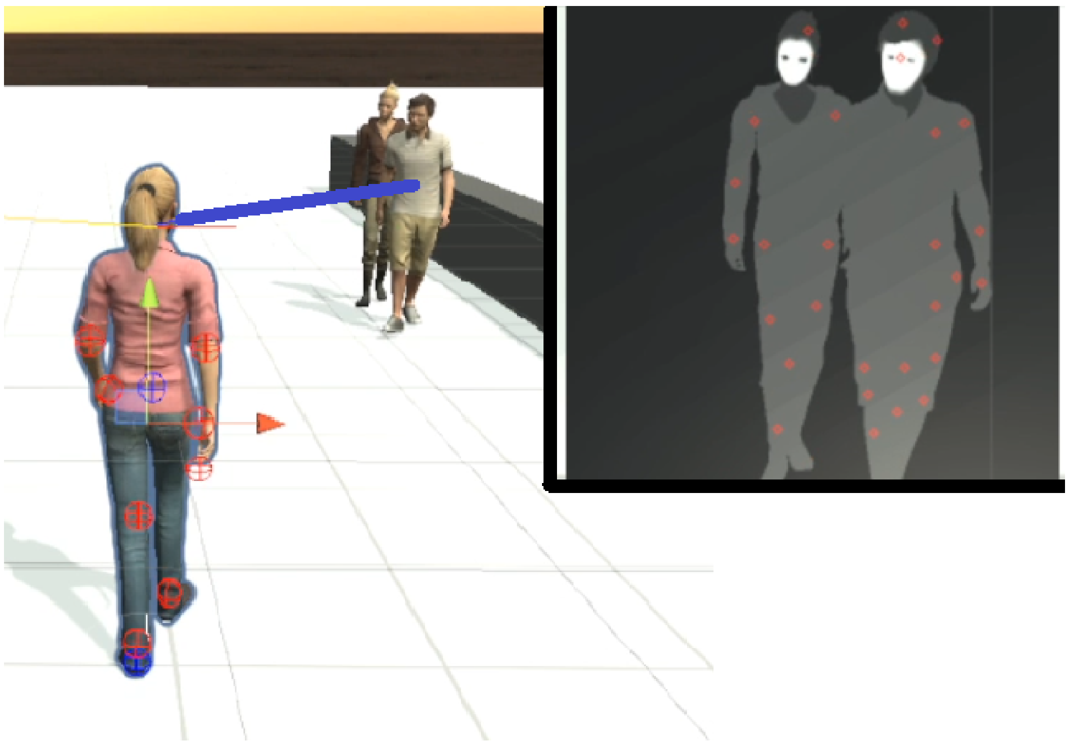


Figure 4.3.1: Screen capture from agent simulation. An agent is viewing two pedestrians walking toward them. On the right is the saliency map of the current view as well as generated possible fixation points overlayed. In this implementation, only the head of agents is animated.

CHAPTER 5

CONCLUSION

Saliency models are seeing more and more use in trying to approximate how humans view and perceive stimuli. Accompanying these models are techniques and tools which use these models to analyze and predict human gaze behaviours. Such models have seen marked improvements in recent years, however, these concepts struggle to integrate themselves into the domain of human agent simulation. The primary reason for this is that most visual models are prohibitively computationally expensive to be used in any real-time application such as video games or crowd simulations.

The objective of this thesis was to explore if a simple model of saliency can output reasonable approximations of state-of-the-art saliency models in the form of saliency maps. It presents a model for parametric saliency maps, a method for generating pseudo-saliency maps from the perspective of a virtual agent as a weighted combination of 7 saliency factors. The saliency parameters were chosen based on extensive research of real-world human visual attention studies, as well as factors used in virtual saliency models. Additionally, reviewing the literature on human visual attention outlined the importance of differentiating top-down attentional factors from bottom-up stimuli. These differences significantly impacted the design of the model, namely how saliency factors are combined between additive features and modulatory features. The model is highly adaptable, weights can be changed to cover a wide range of plausible representations of visual attention, for the scope of minimally interacting casual pedestrians.

The intent generating realistic saliency maps for virtual humans is to use them to inform and drive behaviour. To this end, this thesis also proposes two methods for low-level saliency-driven gaze control. The particle model is a novel approach which interprets the 2D saliency map information as a potential field (in the physics sense), in which gaze can be controlled via physics simulation by having forces applied to a virtual particle to attract it towards salient stimuli in view. The probabilistic model interprets saliency maps as probability densities for fixations and samples potential fixation points according to a simple rule set. This model is inspired by existing fixation prediction models for static 2D images and adapts concepts for dynamic 3D scenes.

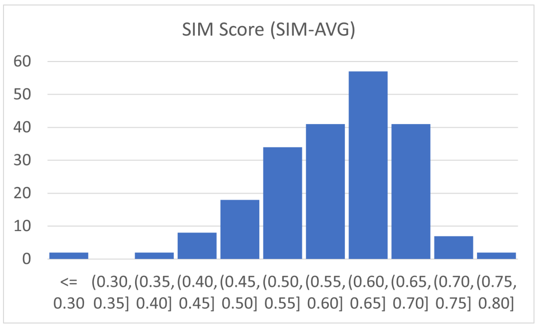
To validate the plausibility of the parametric saliency maps, as well as demonstrate their adaptability an experiment was conducted to optimize the weights of the model in order to approximate a state-of-the-art saliency model. For a fixed set of randomized snapshots from an urban simulation, each element consisted of a generated parametric saliency map and a corresponding 'real' saliency map both representing the same image. The parametric weights for each saliency factor were optimized using covariance matrix adaptation evolutionary strategy. The optimization was implemented according to saliency comparison metrics. The end result showed that to a reasonable degree, the parametric saliency maps model is able to approximate the outputs from a more complicated saliency model, despite having significantly fewer degrees of freedom. Additionally, the analysis found that even unoptimized outputs demonstrated a degree of similarity. This suggests that there are inherent structural similarities in the outputs of parametric saliency maps and 'real' maps generated from a saliency model, and speaks to the validity of the chosen saliency factors.

Future work may include further exploration into optimizing parametric saliency maps to approximate other models of saliency, as well as explore different or expanded sets of parameters. Furthermore, extensions could be made in the form of high-level control which ingrates saliency map generation with saliency-driven gaze control, which autonomously adjusts a virtual agent's attention based on goals, distractions etc and subsequently informs the gaze control module. Experimentation on the proposed saliency-driven gaze control models can be done to determine optimal control parameter values for desired behaviours, as well as quantitative evaluation of results.

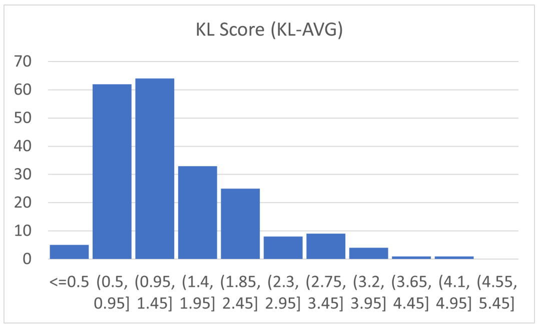
CHAPTER 6

APPENDICES

6.1 Appendix I: Supplemental Figures

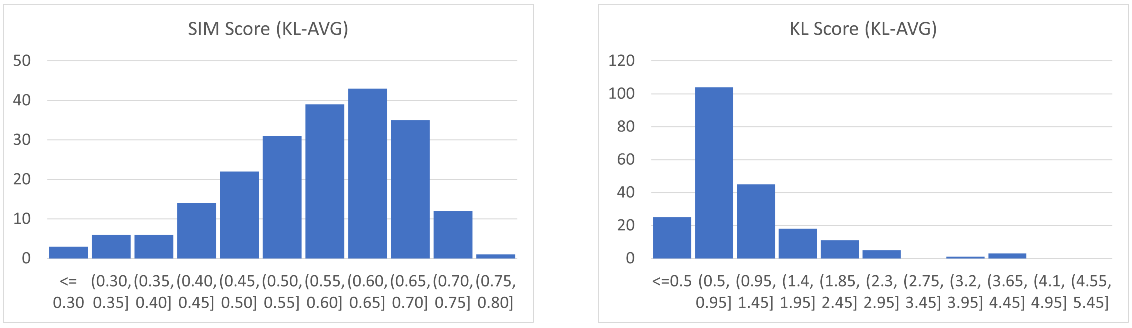


(a)



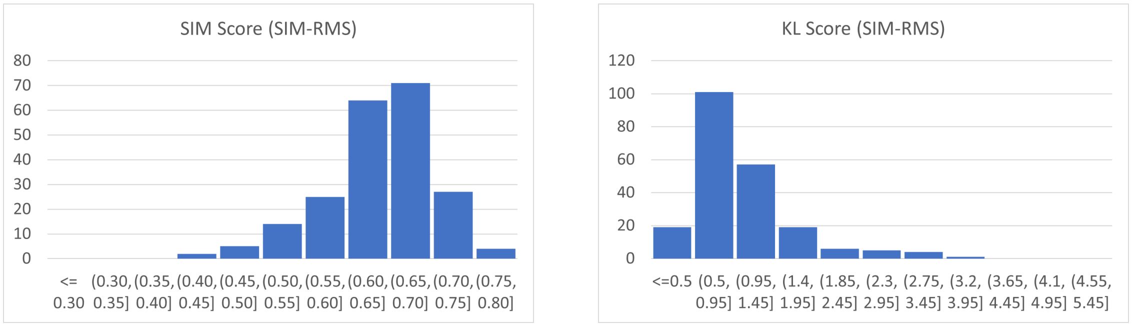
(b)

Figure 6.1.1: Distributions of metric scores for model optimized with SIM loss and AVG objective, (a) SIM Score (b) Kullbach-Liebler Score



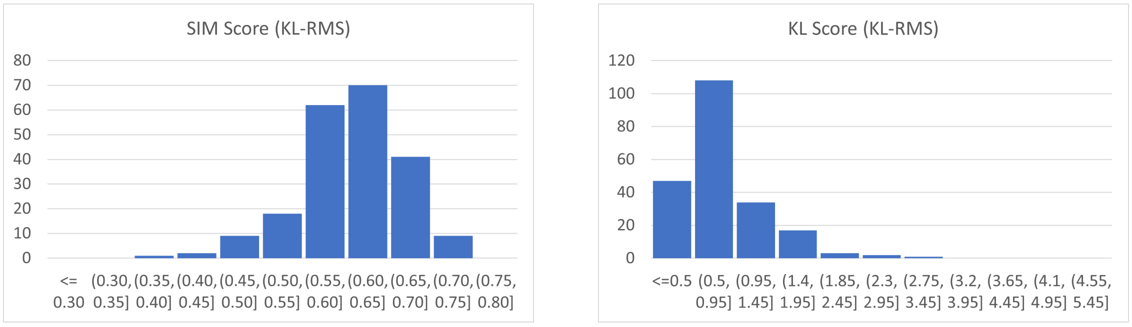
(a) (b)

Figure 6.1.2: Distributions of metric scores for model optimized with KL loss and AVG objective, (a) SIM Score (b) Kullbach-Liebler Score



(a) (b)

Figure 6.1.3: Distributions of metric scores for model optimized with SIM loss and RMS objective, (a) SIM Score (b) Kullbach-Liebler Score



(a) (b)

Figure 6.1.4: Distributions of metric scores for model optimized with KL loss and RMS objective, (a) SIM Score (b) Kullbach-Liebler Score

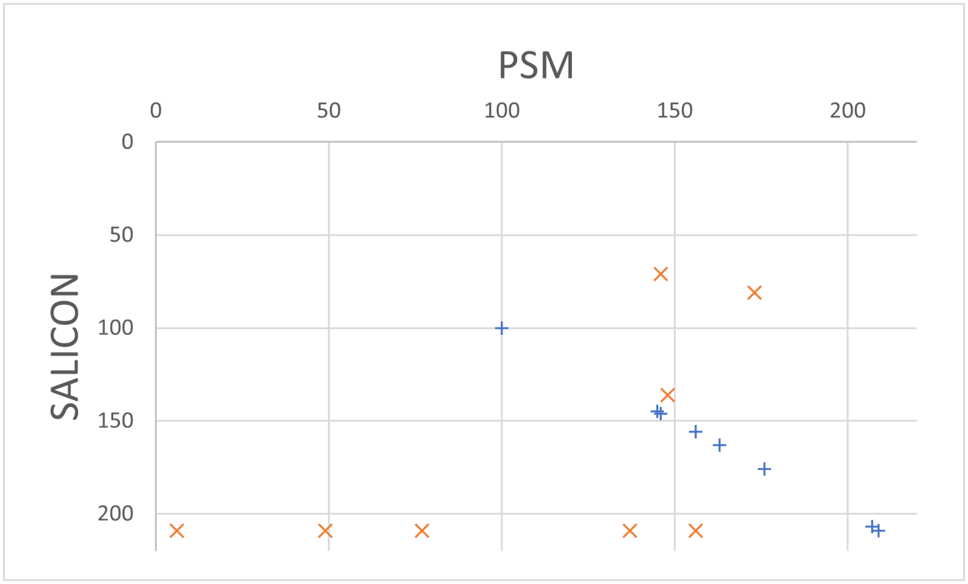


Figure 6.1.7: Prediction plot for dataset generated from default parameters (all weights set to 1). As a baseline, it is expected that very few true positives should emerge, denoted by blue (+) points along the diagonal.

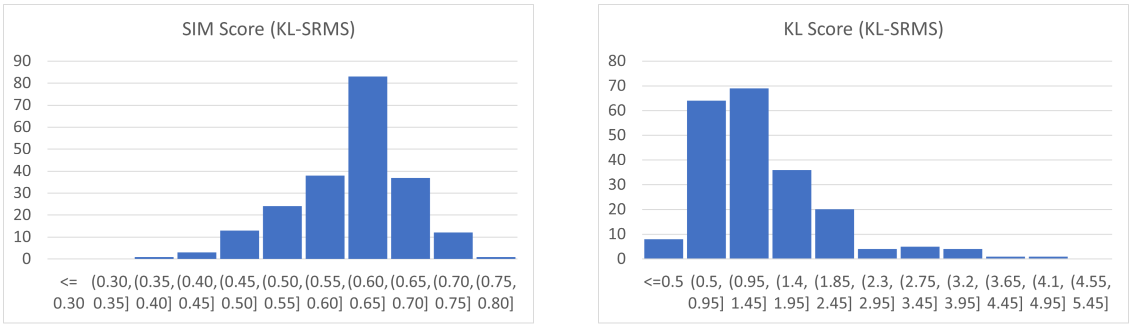


Figure 6.1.5: Distributions of metric scores for model optimized with KL loss and SRMS objective, (a) SIM Score (b) Kullbach-Liebler Score

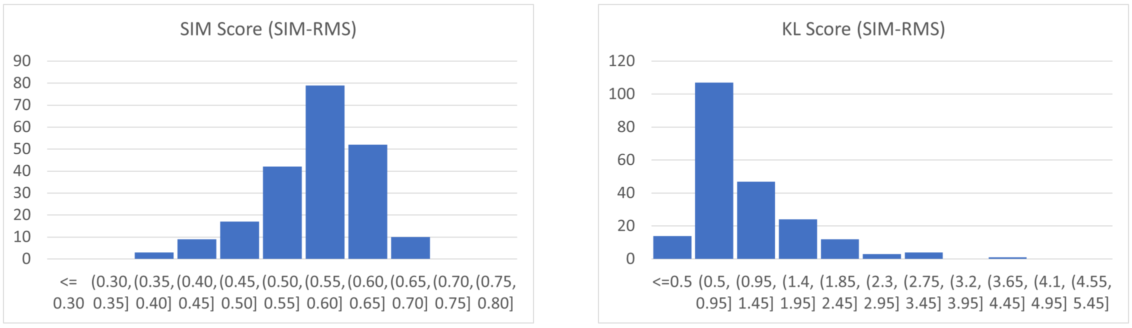
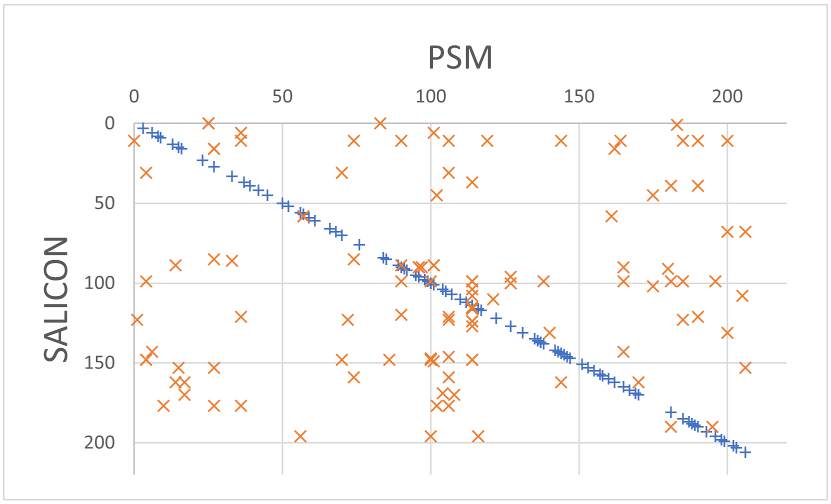
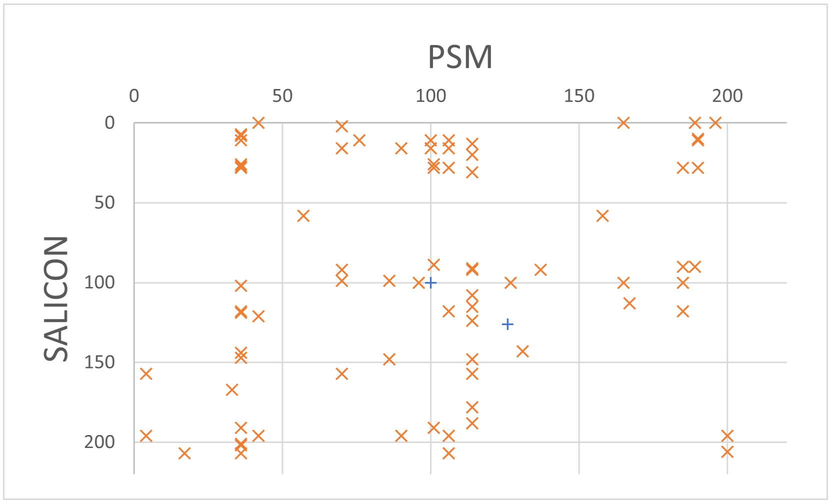


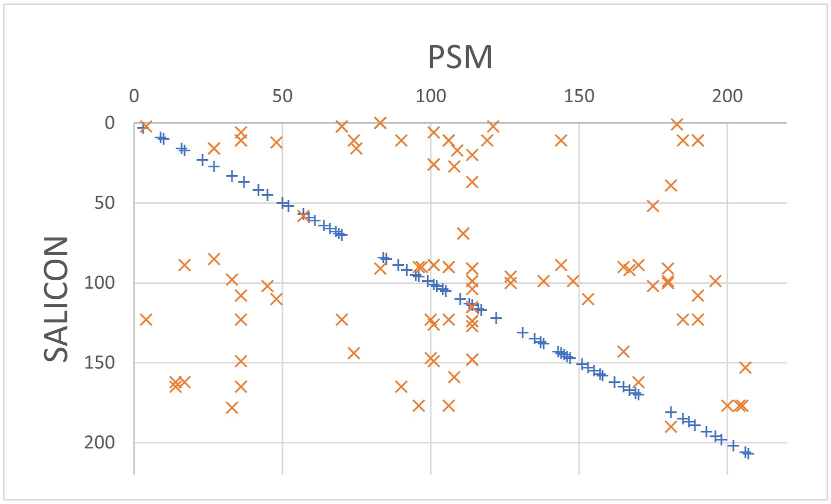
Figure 6.1.6: Distributions of metric scores for model optimized with KL loss and SRMS objective, (a) SIM Score (b) Kullbach-Liebler Score



(a)

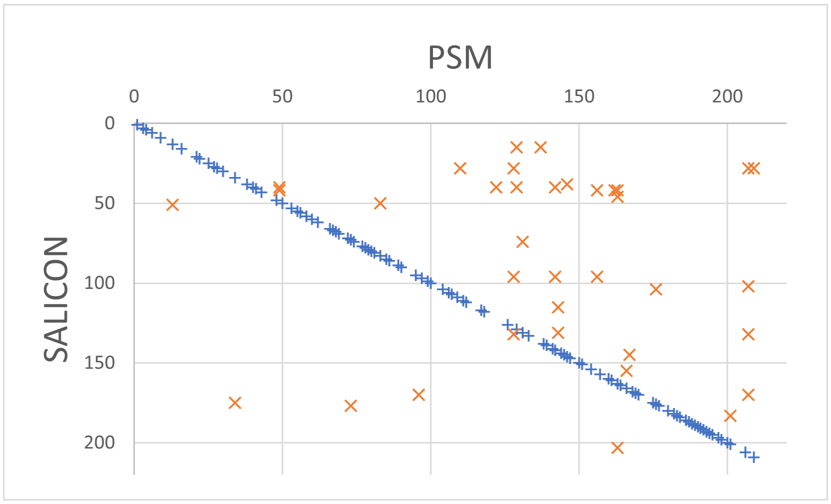


(b)

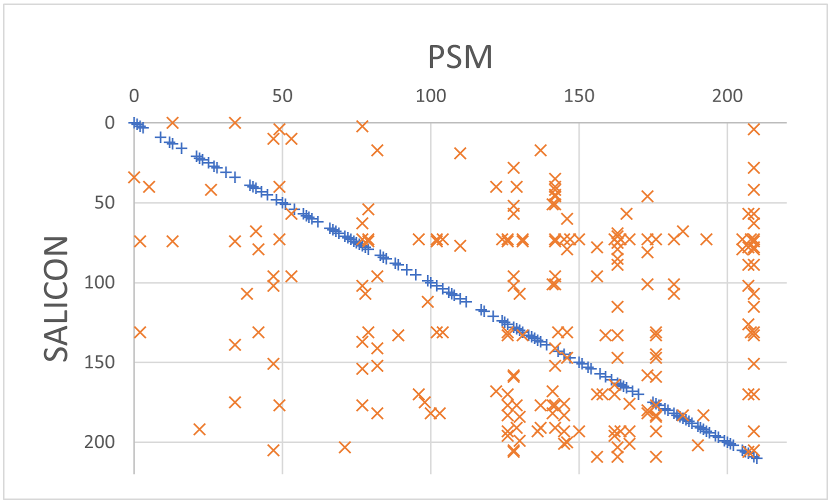


(c)

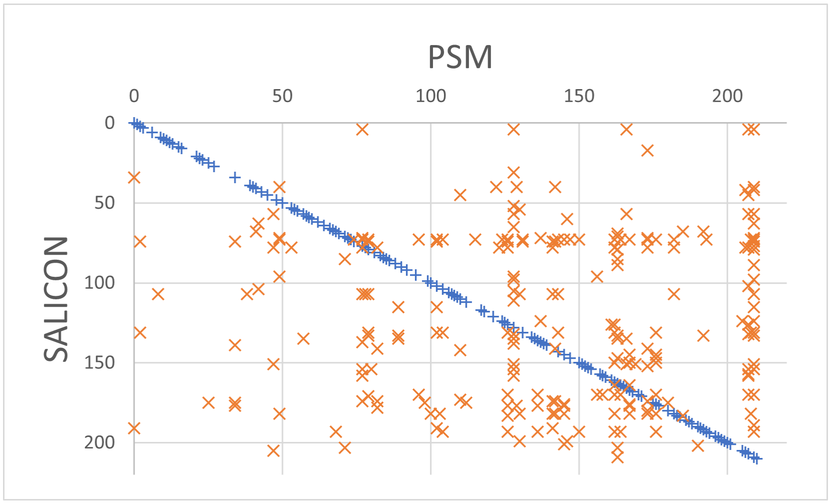
Figure 6.1.8: Prediction plots for varying models (a) KL, (b) SIM, (c) KL-SIM; optimized with a AVG objective. Blue (+) points along the diagonal are true positives. Orange (x) points are false positive. For each image pair, they are said to be the same if the Kullbach-Liebler divergence is ≤ 1.00 and the Similarity score is ≥ 0.6



(a)

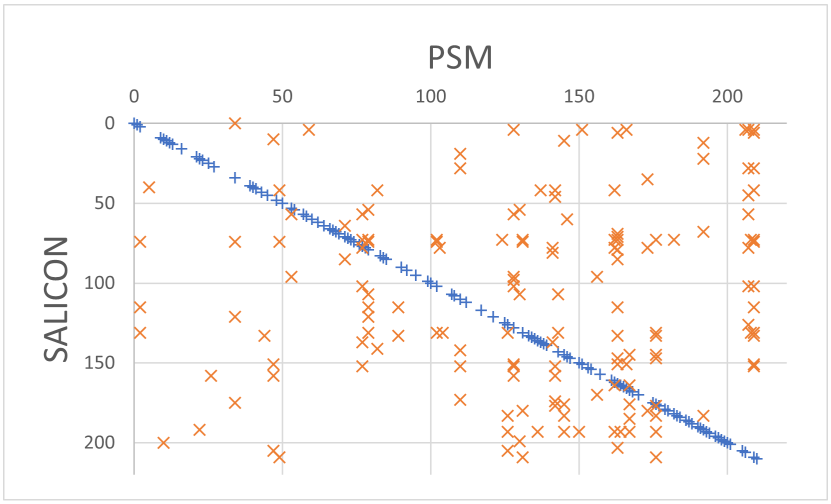


(b)

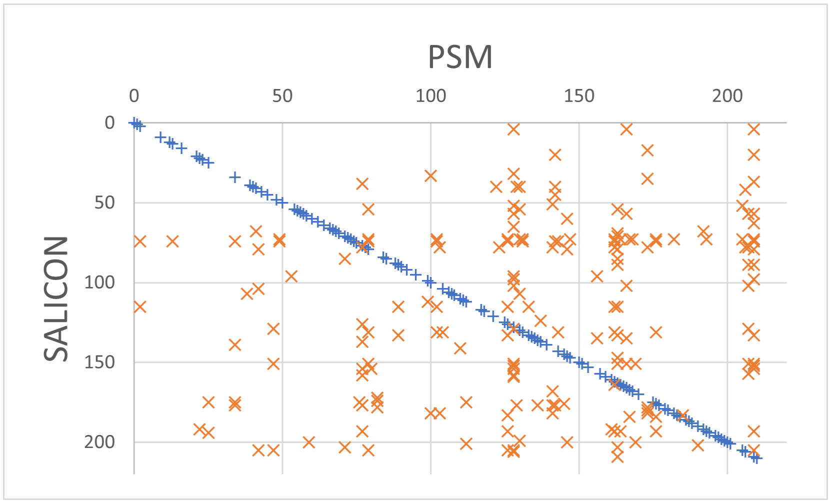


(c)

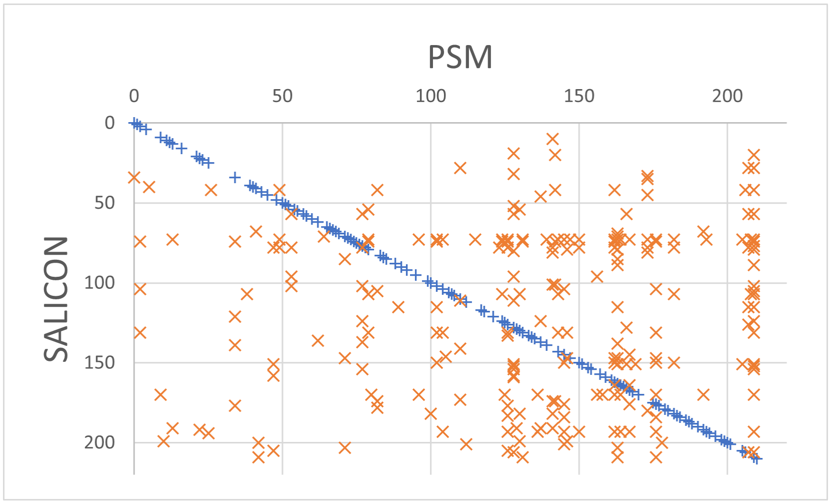
Figure 6.1.9: Prediction plots for varying models (a) KL, (b) SIM, (c) KL-SIM; optimized with a RMS objective. Blue (+) points along the diagonal are true positives. Orange (x) points are false positive. For each image pair, they are said to be the same if the Kullbach-Liebler divergence is ≤ 1.00 and the Similarity score is ≥ 0.6



(a)



(b)



(c)

Figure 6.1.10: Prediction plots for varying models (a) KL, (b) SIM, (c) KL-SIM; optimized with a stochastic RMS objective. Blue (+) points along the diagonal are true positives. Orange (x) points are false positive. For each image pair, they are said to be the same if the Kullbach-Liebler divergence is ≤ 1.00 and the Similarity score is ≥ 0.6

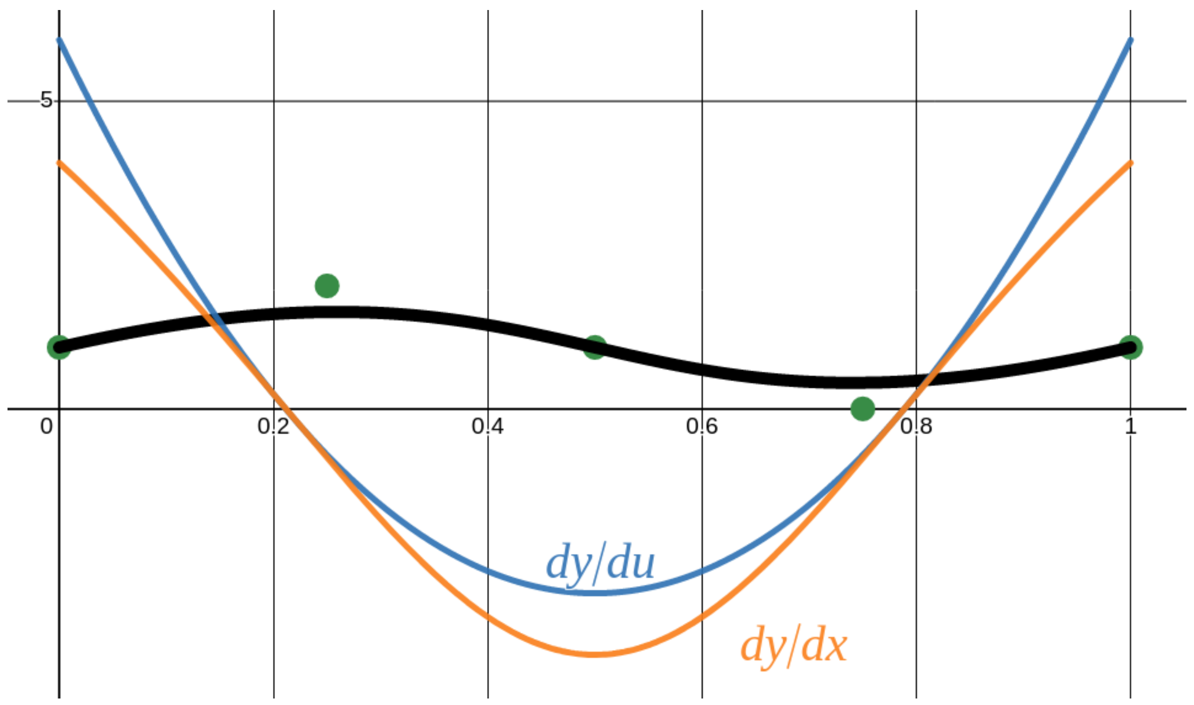


Figure 6.2.1: y derivatives of the spline curve (black) with respect to parametric value u versus x. $\frac{dy}{du}$ is analytical, while $\frac{dy}{dx}$ is computed numerically

6.2 Appendix II: Spline Linearization

Consider the gradient of the curve shown in Figure 6.2.1. Both functions have roots at the same positions at the same values for x as well as u meaning that both functions will always have the same sign. Even the shapes of the functions are very similar but, importantly: they are still different valued functions.

Using only 1 dimensional control points, and substituting parametric values for other dimensions yields a function like y(x) or height-field z(x,y). But they are inherently still just one dimensional splines. Suppose one wished to define a spline with 2D or 3D control points which still represented y(x) or z(x,y)? How can this be achieved for a spline with a standard knot vector, with $n+1$ control points x_i, $(i = 0, 1..., n)$ of degree p? Lets consider a degree 3 B-spline in 2 dimensions from Equation 4.1.1. The standard knot vector will have repeating knots at the beginning and end such that the spline begins and ends on the first and last points. let $k_i \in \vec{K}$ be the standard knot vector for $n+1$ points and degree p,

$$u_i = \begin{cases} 0 & \text{if } 0 \le i \le p \\ 1 & \text{if } n+1 \le i \\ \frac{i-p}{n-c} & \text{else} \end{cases} \tag{6.2.1}$$

Since $k_i \in [0,1]$, the parametric domain is defined from $u \in [0,1]$. The goal is to find some arrangement of control points $x_i \in X$ such that $x(u) = u$. Since the spline begins and ends at x_0 and x_n, then those points should be at 0 and 1 respectively. Also, $x_{i+1} \ge x_i \ \forall i$ must be true clearly. Now we can begin searching for candidate functions to generate positions. Since the goal is to achieve $x(t) = t$, then $x'(t) = 1$. It can be seen that for $x'(0) = 1$ to be true, this depends on only the first and second points in the spline (and the symmetric case is true for $x'(1) = 1$).

$$x'(t) = x_0 N'_{0,3}(0) + x_1 N'_{1,3}(0) + x_2 N'_{2,3}(0) + ... \tag{6.2.2}$$

A property of B-spline basis functions from a standard knot vector is that at the start and end of the parametric space, only the first two and last two (respectively) basis functions are non-zero, can be seen in Fig. Thus equation 6.2.2 can be simplified to

$$\begin{aligned} x'(t) &= x_0 N'_{0,3}(0) + x_1 N'_{1,3}(0) \\ &= 0 \cdot N'_{0,3}(0) + x_1 N'_{1,3}(0) \\ &= x_1 N'_{1,3}(0) = 1 \\ x_1 &= \frac{1}{N'_{1,3}(0)} \\ N'_{1,3} &= \frac{3}{u_4} N_{1,2}(0) - \frac{3}{2u_4} N_{2,2}(0) \end{aligned} \tag{6.2.3}$$

since at $u = 0$*,* $N_{1,2}(0)$ *must be* 1*, then all other basis* $N_{i,2}(0)$ *must be* 0 (see [56])

$$\implies x_1 = \frac{u_4}{3} := \lambda$$

By symmetry, $x_{n-1} = 1 - \lambda$. The intuition here is that since for a standard knot vector there are $p+1$ repeated knots at the front and end, the parametric space is stretched unevenly **redNOTE: include figure on x(t) with equispaced points**. Now by setting the remaining points to be equally spaced this should result in the desired outcome. A good candidate for positions is to take the values at the center of the knot vector,

$$x_i = u_{i+c-1} = u_{i+2} \text{ for } 2 \leq i \leq n-2 \tag{6.2.4}$$

Giving a final function for generating points,

$$x_i = \begin{cases} \lambda & \text{if } i = 1 \\ 1-\lambda & \text{if } i = N-1 \\ u_{i+2} & \text{else} \end{cases} \tag{6.2.5}$$

To prove this, lets first consider the simplest case for a degree 3 2D B-spline with 4 control points. From equation 6.2.5, $X = 0, 1/3, 2/3, 1$

$$\begin{aligned} x'(t) &= \sum_{i=0}^{3} N_{i,3}(t)x_i \\ &= \frac{1}{3}N'_{1,3} + \frac{2}{3}N'_{2,3} + N'_{3,3} \\ &= \frac{1}{3}(3N_{1,2} - 3N_{2,2}) + \frac{2}{3}(3N_{2,2} - 3N_{3,2}) + (3N_{3,2} - 0) \\ &= N_{1,2} + N_{2,2} + N_{3,2} = 1 \end{aligned} \tag{6.2.6}$$

A property of b-spline basis functions is that $\sum_{i=0}^{m} N_{i,c} = 1$, or in other words - the sum of all basis functions of one layer in the recursive pyramid is an affine combination (Partition of Unity) [56]. Now lets consider the more general case of a cubic B-spline with $n+1$ points.

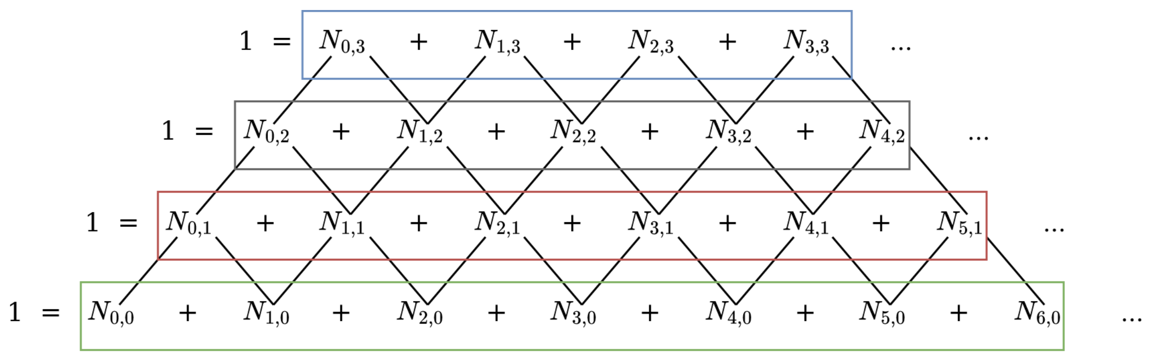

1 = $N_{0,3}$ + $N_{1,3}$ + $N_{2,3}$ + $N_{3,3}$ ...
1 = $N_{0,2}$ + $N_{1,2}$ + $N_{2,2}$ + $N_{3,2}$ + $N_{4,2}$ ...
1 = $N_{0,1}$ + $N_{1,1}$ + $N_{2,1}$ + $N_{3,1}$ + $N_{4,1}$ + $N_{5,1}$ ...
1 = $N_{0,0}$ + $N_{1,0}$ + $N_{2,0}$ + $N_{3,0}$ + $N_{4,0}$ + $N_{5,0}$ + $N_{6,0}$ ...

Figure 6.2.2: Recursive pyramid for B-Spline basis functions. The sum of all non-zero valued basis functions in a layer in the pyramid is an affine combination. A basis function is non-zero for at most $p+1$ intervals. This can be seen by drawing a triangle from a basis function to the bottom. E.g., $N_{1,3}$ is only non-zero on $[u_1, u_5]$

$$
\begin{aligned}
x'(u) &= \sum_{i=0}^{n} N'_{i,3}(u) \cdot x_i \\
&= N'_{0,3} \cdot x_0 + N'_{1,3} \cdot x_1 \; ... \; + \; N'_{n,3} \cdot x_n \\
&\textit{from 6.2.5, substitute values for } x_i. \textit{ (define } x_0 = 0 \textit{ and } x_n = 1\textit{)} \\
&= \lambda N'_{1,3} + u_4 N'_{2,3} + ... u_n N'_{n-2,3} + (1-\lambda) N'_{n-1,3} + N'_{n,3}. \\
&\textit{it can be shown from 6.2.1 that the sequence } u_4, u_5, ... u_{n-2} \textit{ can be written } u_4, 2u_4, 3u_4 ... \\
&= \lambda N'_{1,3} + u_4 N'_{2,3} + 2u_4 N'3,3 + ... + (1-\lambda) N'_{n-1,3} + N'_{n,3}
\end{aligned}
\tag{6.2.7}
$$

Substituting the definition of the derivative from Equation 4.1.6, the terms can be enumerated in conveniently terms of u_4 coefficients,

$$
\begin{aligned}
N'_{1,3} &= \frac{3}{u_4 - u_1} N_{1,2} - \frac{3}{u_5 - u_2} N_{2,2} \rightarrow \frac{3}{u_4 - 0} N_{1,2} - \frac{3}{2u_4 - 0} N_{2,2} \\
N'_{2,3} &= \frac{3}{u_5 - u_2} N_{2,2} - \frac{3}{u_6 - u_3} N_{3,2} \rightarrow \frac{3}{2u_4} N_{2,2} - \frac{3}{3u_4 - 0} N_{3,2} \\
N'_{3,3} &= \frac{3}{u_6 - u_3} N_{3,2} - \frac{3}{u_7 - u_4} N_{4,2} \rightarrow \frac{1}{u_4} N_{3,2} - \frac{1}{u_4} N_{4,2}
\end{aligned}
\tag{6.2.8}
$$

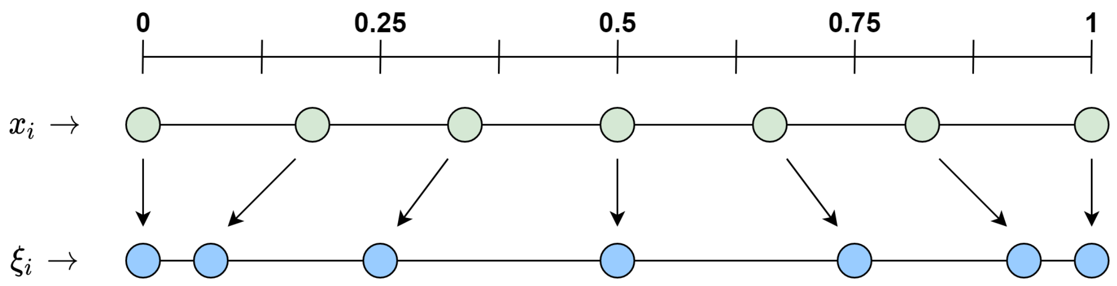


Figure 6.2.3: Transformation from equidistant control points to Greville Abscissae for a degree 3 b-spline with 7 control points

For derivative functions with $3 \leq i \leq n-3$ the form will always be $\frac{1}{u_4}N_{i,2} - \frac{1}{u_4}N_{i+1,2}$. This will be used to cancel out terms when the above is substituted into Equation 6.2.7

$$\begin{aligned}
&= \lambda\frac{3}{u_4}N_{1,2} - \lambda\frac{3}{2u_4}N_{2,2} + u_4[\frac{3}{2u_4}N_{2,2} - \frac{1}{u_4}N_{3,2}] + u_4[\frac{1}{u_4}N_{3,2} - \frac{1}{u_4}N_{4,2}] + ...\\
&= \lambda\frac{3}{u_4}N_{1,2} - \lambda\frac{3}{2u_4}N_{2,2} + \frac{3}{2}N_{2,2} - N_{3,2} + 2N_{3,2} - 2N_{4,2} + 3N_{4,2} - 4N_{5,2} + 5B_{5,2} - ...\\
&= \lambda\frac{3}{u_4}N_{1,2} - \lambda\frac{3}{2u_4}N_{2,2} + \frac{3}{2}N_{2,2} + N_{3,2} + N4,2 + N5,2 + ...
\end{aligned} \tag{6.2.9}$$

substitute definition of λ from 6.2.3

$$= N_{1,2} + N_{2,2} + N_{3,2} + ... + N_{n+1,2} = 1$$

From the properties of B-spline basis functions, the summation of all functions in a layer is always affine. Thus, it is proven for this choice of control point spacing in x, that $x'(u) = 1$, the spline is now linearized such that $\frac{dy}{dx} = \frac{dy}{du}$. What is interesting to note is that this approach does not work for quadratic B-splines. If one would attempt it, they would quickly realize that not only does it require solving for multiple magic numbers, but also the amount of numbers needed changes with the number of control points. The transformed x-coordinates in this form are known as the Greville Abscissae. A closed form solution for all Greville Abscissae for a spline of order p is given as an average of knot values,

$$\xi_i = \frac{1}{p-1}\sum_{k=1}^{p-1} u_{i+k} \tag{6.2.10}$$

BIBLIOGRAPHY

[1] R. A. Abrams and R. S. Dobkin. Inhibition of return: effects of attentional cuing on eye movement latencies. *Journal of Experimental Psychology: Human Perception and Performance*, 20(3):467, 1994.

[2] U. Ağıl and U. Güdükbay. A group-based approach for gaze behavior of virtual crowds incorporating personalities. *Computer Animation and Virtual Worlds*, 29(5):e1806, 2018.

[3] N. I. Badler, D. M. Chi, and S. Chopra. Virtual human animation based on movement observation and cognitive behavior models. In *Proceedings Computer Animation 1999*, pages 128–137. IEEE, 1999.

[4] W. Becker. Saccades. In *Eye Movements*, volume 8, pages 95–117. CRC Pres, Inc., 1991.

[5] J. A. Brefczynski and E. A. DeYoe. A physiological correlate of the'spotlight'of visual attention. *Nature neuroscience*, 2(4):370–374, 1999.

[6] N. D. Bruce and J. K. Tsotsos. Saliency, attention, and visual search: An information theoretic approach. *Journal of vision*, 9(3):5–5, 2009.

[7] N. D. Bruce, C. Wloka, N. Frosst, S. Rahman, and J. K. Tsotsos. On computational modeling of visual saliency: Examining what's right, and what's left. *Vision research*, 116:95–112, 2015.

[8] Z. Bylinskii, T. Judd, A. Oliva, A. Torralba, and F. Durand. What do different evaluation metrics tell us about saliency models? *IEEE transactions on pattern analysis and machine intelligence*, 41(3):740–757, 2018.

[9] R. Datta and E. A. DeYoe. I know where you are secretly attending! the topography of human visual attention revealed with fmri. *Vision research*, 49(10):1037–1044, 2009.

[10] Y. Fang, R. Nakashima, K. Matsumiya, I. Kuriki, and S. Shioiri. Eye-head coordination for visual cognitive processing. *PloS one*, 10(3):e0121035, 2015.

[11] E. G. Freedman. Coordination of the eyes and head during visual orienting. *Experimental brain research*, 190(4):369–387, 2008.

[12] A. Fuchs. Saccadic and smooth pursuit eye movements in the monkey. *The Journal of Physiology*, 191(3):609, 1967.

[13] A. Gibaldi and M. S. Banks. Binocular eye movements are adapted to the natural environment. *Journal of Neuroscience*, 39(15):2877–2888, 2019.

[14] B. S. Gibson and H. Egeth. Inhibition of return to object-based and environment-based locations. *Perception & Psychophysics*, 55(3):323–339, 1994.

[15] K. A. Hamel, N. Okita, J. S. Higginson, and P. R. Cavanagh. Foot clearance during stair descent: effects of age and illumination. *Gait & posture*, 21(2):135–140, 2005.

[16] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of IEEE international conference on evolutionary computation*, pages 312–317. IEEE, 1996.

[17] X. Hou and L. Zhang. Dynamic visual attention: Searching for coding length increments. 2009.

[18] X. Huang, C. Shen, X. Boix, and Q. Zhao. Salicon: Reducing the semantic gap in saliency prediction by adapting deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 262–270, 2015.

[19] L. Itti and C. Koch. Feature combination strategies for saliency-based visual attention systems. *Journal of Electronic imaging*, 10(1):161–169, 2001.

[20] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[21] M. Jiang, S. Huang, J. Duan, and Q. Zhao. Salicon: Saliency in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1072–1080, 2015.

[22] T. Judd, K. Ehinger, F. Durand, and A. Torralba. Learning to predict where humans look. In *2009 IEEE 12th international conference on computer vision*, pages 2106–2113. IEEE, 2009.

[23] S. C. Khullar and N. I. Badler. Where to look? automating attending behaviors of virtual human characters. *Autonomous Agents and Multi-Agent Systems*, 4(1):9–23, 2001.

[24] H. Kim, S. Lee, and A. C. Bovik. Saliency prediction on stereoscopic videos. *IEEE Transactions on Image Processing*, 23(4):1476–1490, 2014.

[25] R. M. Klein. Inhibition of return. *Trends in cognitive sciences*, 4(4):138–147, 2000.

[26] R. M. Klein and J. Ivanoff. Inhibition of return. In *Neurobiology of Attention*, chapter 16, pages 96–100. Elsevier Academic Pres, 2005.

[27] E. Kokkinara, O. Oyekoya, and A. Steed. Modelling selective visual attention for autonomous virtual characters. *Computer Animation and Virtual Worlds*, 22(4):361–369, 2011.

[28] M. Kremer, P. Caruana, B. Haworth, M. Kapadia, and P. Faloutsos. Psm: Parametric saliency maps for autonomous pedestrians. In *Motion, Interaction and Games*, pages 1–7. 2021.

[29] M. Kremer, P. Caruana, B. Haworth, M. Kapadia, and P. Faloutsos. Automatic estimation of parametric saliency maps (psms) for autonomous pedestrians. *Computers & Graphics*, 2022.

[30] M. Kremer, B. Haworth, M. Kapadia, and P. Faloutsos. Modelling distracted agents in crowd simulations. *The Visual Computer*, 37(1):107–118, 2021.

[31] M. Kümmerer, M. Bethge, and T. S. Wallis. Deepgaze iii: Modeling free-viewing human scanpaths with deep learning. *Journal of Vision*, 22(5):7–7, 2022.

[32] M. Kümmerer, T. S. Wallis, and M. Bethge. Deepgaze ii: Reading fixations from deep features trained on object recognition. *arXiv preprint arXiv:1610.01563*, 2016.

[33] M. Kümmerer, T. S. Wallis, and M. Bethge. Saliency benchmarking made easy: Separating models, maps and metrics. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 770–787, 2018.

[34] M. Kümmerer, T. S. Wallis, L. A. Gatys, and M. Bethge. Understanding low-and high-level contributions to fixation prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4789–4798, 2017.

[35] O. Le Meur, P. Le Callet, and D. Barba. Predicting visual fixations on video based on low-level visual features. *Vision research*, 47(19):2483–2498, 2007.

[36] S. Lee, G. J. Kim, and S. Choi. Real-time tracking of visually attended objects in virtual environments and its application to lod. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):6–19, 2008.

[37] S. P. Lee. *Facial animation system with realistic eye movement based on a cognitive model for virtual agents.* University of Pennsylvania, 2002.

[38] H. Leibowitz and S. Appelle. The effect of a central task on luminance thresholds for peripherally presented stimuli. *Human Factors*, 11(4):387–391, 1969.

[39] J. C. Martınez-Trujillo and S. Treue. Attentional modulation strength in cortical area mt depends on stimulus contrast. *Neuron*, 35(2):365–370, 2002.

[40] J. J. McDonald and L. M. Ward. Spatial relevance determines facilitatory and inhibitory effects of auditory covert spatial orienting. *Journal of Experimental Psychology: Human Perception and Performance*, 25(5):1234, 1999.

[41] T. A. Mondor, L. M. Breau, and B. Milliken. Inhibitory processes in auditory selective attention: Evidence of location-based and frequency-based inhibition of return. *Perception & Psychophysics*, 60(2):296–302, 1998.

[42] R. Nakashima and S. Shioiri. Why do we move our head to look at an object in our peripheral region? lateral viewing interferes with attentive search. *PloS one*, 9(3):e92284, 2014.

[43] O. Oyekoya, W. Steptoe, and A. Steed. A saliency-based method of simulating visual attention in virtual scenes. In *Proceedings of the 16th ACM symposium on virtual reality software and technology*, pages 199–206, 2009.

[44] J. Pan, C. C. Ferrer, K. McGuinness, N. E. O'Connor, J. Torres, E. Sayrol, and X. Giro-i Nieto. Salgan: Visual saliency prediction with generative adversarial networks. *arXiv preprint arXiv:1701.01081*, 2017.

[45] J. Pan, C. C. Ferrer, K. McGuinness, N. E. O'Connor, J. Torres, E. Sayrol, and X. Giro-i Nieto. Salgan: Visual saliency prediction with generative adversarial networks. *arXiv preprint arXiv:1701.01081*, 2017.

[46] C. Peters and C. O'Sullivan. Bottom-up visual attention for virtual human animation. In *Proceedings 11th IEEE International Workshop on Program Comprehension*, pages 111–117. IEEE, 2003.

[47] M. F. Peterson and M. P. Eckstein. Looking just below the eyes is optimal across face recognition tasks. *Proceedings of the National Academy of Sciences*, 109(48):E3314–E3323, 2012.

[48] M. I. Posner, Y. Cohen, et al. Components of visual orienting. *Attention and performance X: Control of language processes*, 32:531–556, 1984.

[49] M. I. Posner, R. D. Rafal, L. S. Choate, and J. Vaughan. Inhibition of return: Neural basis and function. *Cognitive neuropsychology*, 2(3):211–228, 1985.

[50] J. Prochazkova. Derivative of b-spline function. In *Proceedings of the 25th Conference on Geometry and Computer Graphics, Prague, Czech Republic*, pages 12–16, 2005.

[51] U. Rajashekar, L. K. Cormack, and A. C. Bovik. Point-of-gaze analysis reveals visual search strategies. In *Human vision and electronic imaging IX*, volume 5292, pages 296–306. International Society for Optics and Photonics, 2004.

[52] J. H. Reynolds and R. Desimone. Interacting roles of attention and visual salience in v4. *Neuron*, 37(5):853–863, 2003.

[53] J. H. Reynolds, T. Pasternak, and R. Desimone. Attention increases sensitivity of v4 neurons. *Neuron*, 26(3):703–714, 2000.

[54] N. Riche, M. Duvinage, M. Mancas, B. Gosselin, and T. Dutoit. Saliency and human fixations: State-of-the-art and study of comparison metrics. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[55] K. Ruhland, C. E. Peters, S. Andrist, J. B. Badler, N. I. Badler, M. Gleicher, B. Mutlu, and R. McDonnell. A review of eye gaze in virtual agents, social robotics and hci: Behaviour generation, user interaction and perception. In *Computer graphics forum*, volume 34, pages 299–326. Wiley Online Library, 2015.

[56] C.-K. Shene. B-spline basis functions: Important properties.

[57] G. Tassinari and D. Campara. Consequences of covert orienting to non-informative stimuli of different modalities: A unitary mechanism? *Neuropsychologia*, 34(3):235–245, 1996.

[58] B. W. Tatler, R. J. Baddeley, and I. D. Gilchrist. Visual correlates of fixation selection: Effects of scale and time. *Vision research*, 45(5):643–659, 2005.

[59] S. P. Tipper, J. Driver, and B. Weaver. Object-centred inhibition of return of visual attention. *The Quarterly Journal of Experimental Psychology*, 43(2):289–298, 1991.

[60] S. P. Tipper, B. Weaver, L. M. Jerreat, and A. L. Burak. Object-based and environment-based inhibition of return of visual attention. *Journal of Experimental Psychology: Human perception and performance*, 20(3):478, 1994.

[61] S. Treue. Visual attention: the where, what, how and why of saliency. *Current opinion in neurobiology*, 13(4):428–432, 2003.

[62] J. Tsotsos, I. Kotseruba, and C. Wloka. A focus on selection for fixation. *Journal of Eye Movement Research*, 9(5), 2016.

[63] J. K. Tsotsos. *A Computational Perspective on Visual Attention.* MIT Press, 2021.

[64] C. Wloka, I. Kotseruba, and J. K. Tsotsos. Active fixation control to predict saccade sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3184–3193, 2018.

[65] C. Zetsche. Natural scene statistics and salient visual features. In *Neurobiology of Attention*, chapter 37, pages 226–231. Elsevier Academic Pres, 2005.

When asked about the applications of his discoveries, Heinrich Hertz replied,

"Nothing, I guess."

His work conclusively proved the existence of electromagnetic waves; radio waves specifically, a discovery which has fundamentally altered and shaped our modern lives. This is not meant to draw a comparison of this work with the world altering discovery of radio, but to touch on the foundations of what it means to be a scientist. As scientists, we study and explore the world not because we always believe our work will change our lives - but rather because we do *not* know. In many cases, the discovery is the point for the sake of itself. To contribute knowledge to the sum total of humanity is the highest honor.