Dissertation Type: Research

DEPARTMENT OF COMPUTER SCIENCE

ARt-CG:
Assisted Real-time Content Generation of 3D Hair by Learning Manifolds

Dillon Keith Diep [INCOMPLETE DRAFT, NOT FOR SUBMISSION]

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Engineering in the Faculty of Engineering.

Saturday 15th April, 2017

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MEng in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Dillon Keith Diep [INCOMPLETE DRAFT, NOT FOR SUBMISSION], Saturday 15th April, 2017

Contents

List of Figures

List of Tables

List of Algorithms

List of Listings

Executive Summary

A compulsory section, of at most 1 page

The topic of this thesis explores assisted content generation by training generative models using unsupervised machine learning for the production of 3D hair geometry. The research hypothesis of this study is that non-linear probabilistic principal component analysis with the Gaussian Process Latent Variable Model is applicable for improving the creative production workflow of complex 3D hair geometry for humanoids.

Production of 3D virtual worlds is a time-consuming and costly process that also demand expert knowledge. 3D assets encompass a vast range of applications, ranging from simulations and research, to contributing towards the functioning of many businesses. The production of 3D assets impacts many industries including engineering, medicine, and the provisioning of entertainment. One particular task is the creation of 3D hair geometry for humanoid characters. The production of 3D hair is arduous as hair structure is a complex system containing much interdependence between components.

Machine learning applications typically use large data sets for training on problems that often have a concise answer for a given prediction. The application of machine learning to enhance production for creative work is an exciting field that tackles novel challenges: artistic products tend to have small sets of data available, and evaluation of quality is subjective. Given the same input, acceptable solutions can vary significantly. The mentioned peculiarities of applying machine learning for 3D mesh data establish a unique field of problems to investigate.

Existing tools for 3D modelling have remained mostly static in the paradigm of approach over the past several decades. Automation through methods such as procedural generation can produce output much faster, but the lack of control over the final result makes it less desirable than traditional methods of 3D modelling. The focus of this project is to formulate a revolutionary approach that improves the workflow of producing 3D hair geometry through unsupervised training of generative models.

- Formulation of a generative model for 3D humanoid hair structure
- Resolving the alignment problem by approximating raw input data using a generative model
- Learning a low dimension latent-space of high dimensional hair structure data
- Evaluated the performance of various kernels for high dimensional training data
- Formulation of reparation techniques on output generation of 3D geometry to conform with established constraints
- Implemented an add-on package for a 3D production program, Blender
  - The implementation creates guiding splines that are useful for generating hair geometry
  - Appropriate for small training set that is practical for content creators
  - Real-time performance that matches state of the art non-learning tools

Supporting Technologies

A compulsory section, of at most 1 page

This section should present a detailed summary, in bullet point form, of any third-party resources (e.g., hardware and software components) used during the project. Use of such resources is always perfectly acceptable: the goal of this section is simply to be clear about how and where they are used, so that a clear assessment of your work can result. The content can focus on the project topic itself (rather, for example, than including "I used LaTeX to prepare my dissertation"); an example is as follows:

- I used the Java `BigInteger` class to support my implementation of RSA.
- I used a parts of the OpenCV computer vision library to capture images from a camera, and for various standard operations (e.g., threshold, edge detection).
- I used an FPGA device supplied by the Department, and altered it to support an open-source UART core obtained from `http://opencores.org/`.
- The web-interface component of my system was implemented by extending the open-source WordPress software available from `http://wordpress.org/`.

Notation and Acronyms

An optional section, of roughly 1 or 2 pages

Any well written document will introduce notation and acronyms before their use, *even if* they are standard in some way: this ensures any reader can understand the resulting self-contained content.

Said introduction can exist within the dissertation itself, wherever that is appropriate. For an acronym, this is typically achieved at the first point of use via "Advanced Encryption Standard (AES)" or similar, noting the capitalisation of relevant letters. However, it can be useful to include an additional, dedicated list at the start of the dissertation; the advantage of doing so is that you cannot mistakenly use an acronym before defining it. A limited example is as follows:

AES	:	Advanced Encryption Standard
DES	:	Data Encryption Standard
	$\vdots$	
$\mathcal{H}(x)$	:	the Hamming weight of x
$\mathbb{F}_q$	:	a finite field with q elements
x_i	:	the i-th bit of some binary sequence x, st. $x_i \in \{0, 1\}$

Acknowledgements

An optional section, of at most 1 page

It is common practice (although totally optional) to acknowledge any third-party advice, contribution or influence you have found useful during your work. Examples include support from friends or family, the input of your Supervisor and/or Advisor, external organisations or persons who have supplied resources of some kind (e.g., funding, advice or time), and so on.

Chapter 1

Contextual Background

1.1 Production of 3D Content

In computer graphics, 3D objects are represented in many forms. 3D scanners capture raw data in various forms such as point clouds, range images, and voxels. A point cloud is simply a collection of points, often used in computer vision to sample surface geometry. Range images map pixels of a depth image to a set of points in the scene. A voxel is a unit cube, corresponding to a pixel, a collection of voxels define the volume of an object. Voxels have applications in many fields, including visualisation of medical data. [10]

In a production environment it is straightforward to define geometry as opposed to capturing examples. The most common representation used for CG production are polygonal meshes, data that contains information of vertices, edges, and faces. The topology of a mesh is the organisation of components that define the geometry. Two surfaces with the same appearance could have different topology, affecting how well the meshes could deform and react to operations. Where precision is concerned, parametric definitions such as NURBS (Non-uniform rational basis spline) are preferred as mathematical models are exact. Each representation has its advantages depending on the use case. It is possible to convert between representations, but data loss may be incurred. Properties that make polygon meshes desirable include efficient rendering, simple to define, expressive enough to capture geometry required, and works well with established techniques such as UV texture mapping and a plethora of mesh-based algorithms. The rendering pipeline often converts meshes to tri-faces (faces constructed by three edges) as an optimisation process, but best practice for 3D artists is to maintain a topology of quad-faces which are easier to organise and conforms better with editing tools and algorithms.

1.1.1 3D Hair Geometry

On average, a human is born with between 90,000 to 150,000 scalp hair follicles. [6] It is computationally very expensive to render and animate physically correct hair, but creative liberties have been taken to approximate, or stylize 3D hair such that it is both acceptable aesthetically and feasible in terms of performance. This study considers modelling of hair geometry, the motion of hair is assumed to be its default resting pose.

In recent years, impressive 3D hair solutions for real-time simulation of realistic hair and fur, such as *NVIDIA HairWorks* and *AMD TressFX* has emerged. These solutions, however, have limited application in comparison to their traditional counterpart of polygonal hair. It is often the case that texture-mapped polygonal hair is used as a fallback when advanced simulation fails. Realism is not necessarily always desirable, polygon hair can flexibly represent different art styles. In some cases, a blend of multiple representations are used to balance between cost and quality. 3D hair in cinematography with large budget can afford to render hair with much higher fidelity for important characters, but would still consider use efficient variants for scenarios such as crowd simulation. Ultimately, it can be observed that the representation of virtual hair generally follows a structure of splines with control points that define the overall organisation of strands or segments.

To add

Figure 1.1: Image of hair geometry.

1.1.2 Procedural Generation and Automated Production

Procedural generation techniques produce output that adhere to rules established by the generative model defined. Such techniques have been successfully applied for generation of terrains and city modelling. [5] Fractals and methods such as the Lindenmayer system has been used to produce patterns that resemble those observed in nature. [?] Automated techniques such as the ones discussed, however, are seldom used for modelling important objects with specific design. It is difficult to control the output of procedurally generated content without heavily restricting its capabilities. Automated methods that do not learn is cannot adapt to changing demands without reimplementation.

1.2 Motivation and Significance

State of the art 3D production software such as AutoDesk Maya, 3DS Max, and Blender are advanced programs with sophisticated list of features. That said, such programs have extremely convoluted user interfaces, even the most experienced professionals do not recognise each and every tool available. The most versatile tools are generally the most basic that perform atomic changes as they are applicable in every scenario. Examples include selection of primitives such as vertices, edges, or faces and performing translation, rotation, and scaling. Sculpting tools move many data points simultaneously, they are often used for defining organic surfaces now that modern machines are sufficiently powerful. Experienced artists might search for an existing base mesh that is similar to start on, but it is not always the case that such a base mesh exists - there are also concerns for quality, such as poor topology. As the geometry becomes more detailed and well-defined, each alteration makes less impact and the space of sensible changes becomes smaller. The design and production of 3D geometry remains a slow and delicate process.

Virtual hair creation is a necessity for characters of CG movies and video games that are embedded within culture both economically and as entertainment. Specialised artists learn to be proficient with the design of hair, variety of styles, and techniques for creating them. Hair geometry is much more concentrated than other types, containing many data points that is exhausting to edit. Soft selection and sculpting tools are good enough for defining the structure but maintaining topology and issues such as overlapping surfaces are still problematic. Learning the relation of hair structure allows the potential of discovering new hairstyles. It can also be used as a mean of rapidly generating initial base geometry that fits the target output better than existing geometry available. Generative methods could ensure a level of quality, clean topology that fits established specifications. Assisted content generation using machine learning provides a convenient, non-intrusive and intuitive method for rapidly generating new hair geometry from existing data.

The application of machine-learning based tools could enhance the workflow of professional users and improve the experience for non-expert consumers. Such tools integrate into the production environment to improve the efficiency of acquiring initial base geometry and visually compare designs during pre-production. Non-expert users receive the ability to produce 3D geometry without requiring to learn the intrinsics of traditional 3D modelling software. The rise of augmented reality and 3D printing inspires the development of generative tools that are intuitive and simplistic to use. Applications that allow users to create personal content could also integrate machine-learning based tools to prevent inappropriate or undesirable creation from being produced while providing options that surpass existing alternatives. An example would be avatar creation for many applications and video games. A space of reasonable options generated from predefined outputs by the developers will allow users to interpolate between sensible configurations, providing an excellent level of customisation while adhering to defined constraints.

1.3 Related Research of Machine Learning in Creative Fields

The task of machine learning can be divided into three major paradigms:

- *Supervised learning* is provided input training examples with desired outputs to learn the mapping of inputs to output.
- *Unsupervised learning* is given only input data, the procedure learns structure of and relation between data.
- *Reinforcement learning* seeks to iteratively improve a pool of solutions by simulating an environment that apply concepts inspired by the theory of evolution.

The role that learning methods play in both manufacturing and consumer application continue to grow, however, adoption has been slow for creative fields. Generally, robust models improve in performance as more reliable data is obtained. Creative production values uniqueness and versatility, properties that cause difficulty in machine learning methods. Varying artistic styles in design complicate feature analysis and ambiguity of correctness is problematic when predicting an output.

To overcome the challenges of the scenario introduced above, unsupervised learning with probabilistic latent variable models such as the Gaussian Process Latent Variable Model [12] present an opportunity to learn stylistic properties of design and predict multiple acceptable outputs by analysing the likelihood. Previous research conducted has explored the idea of applying machine learning in creative fields.

1.3.1 Style-Based Inverse Kinematics

Style-based inverse kinematics introduced the Scaled Gaussian Process Latent Variable Model, based on GPLVM, to learn the probabilistic distribution of a 3D human posture model. [7] Character posing from motion data is represented as a 42-dimensional feature vector that encapsulated joint information of a humanoid body. Learning a model of poses established the relation between joints and identified constraints exhibited in the training data - where unusual postures are given a lower likelihood rating.

1.3.2 Latent Doodle Space

A latent doodle space is the use of a low-dimension latent space that has been applied on simple line drawings. [1] The motivation of a latent doodle space is to generating new drawings that are inspired by the input data. There are two key phases to derive a latent doodle space: the first challenge is to identify line strokes within drawings, a latent variable method is then used to learn a latent space.

1.3.3 Learning a Manifold of Fonts

A study by Campbell & Kautz presented a framework that learns the latent manifold of existing font styles. [3] The process involved universal parametrization of fonts to a polyline representation so that a distance measure is applicable and the generative model can interpolate between styles. Unsupervised learning with the GP-LVM model enabled rapid prototyping and non-experts could create font styles without experience on type design.

1.3.4 Real-time Drawing Assistance through Crowd-sourcing

Drawing assistance powered by large-scale crowd-sourcing explored the potential of data driven drawing to prompt for correction by achieving an artistic consensus. [13] A consensus is found by learning a correction vector field from training drawings. Stroke-correction is applied using the correction vector field to adjust user input dynamically.

1.3.5 AutoHair

Chai, et al introduced AutoHair, a method for automatic modelling of 3D hair from a portrait image. [4] The approach extracts information from images and uses a database of hair meshes to construct a 3D representation of the information conveyed. A hierarchical deep neural network trained on annotated hair images learn to segment hair and estimate growth direction within portraits. Data-driven hair matching and modelling algorithm fit meshes from the database to parameters output by the neural net model to automatically produce 3D hair. The experiment developed a traversable hairstyle space of 50,000 hair models, using training images and 3D exemplars obtained from the internet.

1.4 Challenges

This study faces a number of challenges. Firstly, 3D meshes are difficult to compare. The training data in its raw form will have varying dimensions. Meshes can be viewed as samples of the true geometry, thus meshes that represent the same object could differ drastically in number of data points depending on its level of detail. Typical feature extraction methods do not work well on meshes as artistic products are sensitive to data loss - any change could affect the perception of final result drastically.

Another problem encountered is the lack of training data. Typical machine learning solutions use huge data sets in the order of hundreds of thousands for training, but for 3D meshes the expected size of readily available training data is much smaller. Public repositories of 3D polygonal hair are generally around a few thousand in size. [9] Studios that store and organise past production could likely match the size of public repositories, depending on the size of the company. Independent artists that keep their production will rarely go beyond the range of hundreds.

The application of machine learning methods must also account for subjectivity of evaluating artistic assets. The range of acceptable solutions is ambiguous, likened to how hair styles of characters can change drastically during the design phase, determining the threshold of acceptable solutions will be in itself a chaitllenge to resolve.

As mentioned previously, 3D meshes are delicate and can easily be invalidated from small changes. Thus, reparations to ensure that the output of trained models are acceptable is a topic to explore.

In a production environment, the time required for a technique to return observable result directly affects throughput. For practical efficacy of assisted content generation, the technique should be reasonably fast in presenting observable output.

1.5 Central Objectives

- Resolving the alignment problem of 3D data through a representative generative model.
- Explore the application of GP-LVM for 3D hair geometry in a production pipeline.
- Investigate the use of latent variables for identifying stylistic properties of 3D hair geometry.
- Demonstrate the use of non-linear manifold to generate new hairstyles from training data.
- Enable an intuitive method for non-experts to create 3D hair geometry.
- Observable output demand performance close to real-time for practical use.

Chapter 2

Technical Background

2.1 Principal Component Analysis

In order to comprehend how GPLVM can learn a latent manifold from observed data, first we consider how it was derived, starting from principal component analysis.

In multivariate analysis, principal component analysis (PCA) is a statistical technique used to perform dimensionality reduction. It was originally introduced by Pearson [8], and independently developed by Hotelling [?], where the standard algebraic derivation of PCA was presented in terms of a standardized linear projection.

Consider the properties that define hair structure. Observable variables that can be measured include location, orientation, length, and colour. The data collected may indicate that some variables change together, this relation is measured as the covariance. The PCA technique searches for an ordered set of linear combination of the observed variables that retain maximal variance. Should two observed variables strongly covary linearly, then it is plausible to describe the data with a single variable instead. The more linearly the variables covary, the less data is lost from choosing a smaller set of principal components, thus effectively reducing dimensionality.

Given a set of n observed d-dimensional data in matrix form, $\boldsymbol{X} = [\boldsymbol{x_1, ..., x_n}]^T$, the q principal components $\boldsymbol{w_j}$, $j \in 1, ..., q$, are the orthonormal axes with maximal variance retained. The first principal component is a linear function $\boldsymbol{\alpha}_1^T\boldsymbol{X}$ that retains most variance of $\boldsymbol{X}$, where $\boldsymbol{\alpha} = [\alpha_{11}, \alpha_{12}, ..., \alpha_{1n}]$ is a vector of n constants such that:

$$\boldsymbol{\alpha}_1^T\boldsymbol{X} = \alpha_{11}\boldsymbol{x_1} + \alpha_{12}\boldsymbol{x_2} + ... + \alpha_{1n}\boldsymbol{x_n} = \sum_{i=1}^{n} a_{1i}\boldsymbol{x_i}$$

The following principal components are found by looking for a linear function that is orthogonal to the selected principal components and retain maximum variance.

PCA can be performed by singular value decomposition (SVD) of $\boldsymbol{X}$ in matrix form, [11]

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{L}\boldsymbol{V}^T,$$

where given $r = r(\boldsymbol{X})$ denotes the rank of $\boldsymbol{X}$, then $\boldsymbol{U}$ is a $n \times r$ matrix of orthonormal columns that are the left singular vectors, $\boldsymbol{L}$ is a $r \times r$ diagonal matrix of the singular values of $\boldsymbol{X}$, and $\boldsymbol{V}$ is a $d \times r$ matrix of orthonormal columns that are the right singular vectors.

2.2 Probabilistic Principal Component Analysis

A limitation of standard PCA is the lack of a probabilistic solution. Tipping and Bishop introduced a probabilistic principal component analysis by constraining a latent variable model to be effectively equivalent when its marginal likelihood is maximised. [16]

Suppose that the PCA is applied to a set of data that represents hair structure. Standard PCA searches for the orthonormal axes that retain maximal variance. Extrapolating values using the principal axes can infer new hairstyles, but only within a fixed style embedded by the axes. In the case of 3D content production, it is much more useful to have a selection of plausible designs to choose from. A probabilistic model of PCA will enable exploration of other potential linear embeddings of hair structure.

2.2.1 The Gaussian Distribution

The Gaussian (normal) distribution is a reasonable prior assumption for data that is subject to the central limit theorem, which states that as the sample size of a population tends to infinity, the distribution becomes increasingly Gaussian. [2, p.78] A random variable X that is normally distributed with mean μ and variance σ^2 is denoted as

$$X \sim \mathcal{N}(\mu, \sigma^2).$$

The Gaussian density for a single variable y is expressed as [2, p.78]:

$$\mathcal{N}(y|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right),$$
$$\mathcal{N}(y|\mu,\sigma^2) \equiv p(y|\mu,\sigma^2).$$

Properties of Gaussian Distribution

Notable properties of the Gaussian distribution that are useful include the summation (2.1), scaling (2.2), and product (2.3) operation - all of which yields a result that is also a Gaussian distribution. [15, p.200]

$$\sum_{i=1}^{n} y_i \sim \mathcal{N}(\sum_{i=1}^{n}\mu_i, \sum_{i=1}^{n}\sigma_i^2) \tag{2.1}$$

$$wy \sim \mathcal{N}(w\mu, w^2\sigma^2) \tag{2.2}$$

$$\begin{aligned} \mathcal{N}(\boldsymbol{x}|\boldsymbol{a}, A)\mathcal{N}(\boldsymbol{x}|\boldsymbol{b}, B) &= \mathcal{N}(\boldsymbol{a}|\boldsymbol{b}, A+B)\mathcal{N}(\boldsymbol{x}|\boldsymbol{c}, C), \\ \boldsymbol{c} &= C(A^{-1}\boldsymbol{a} + B^{-1}\boldsymbol{b}), \\ C &= (A^{-1} + B^{-1})^{-1}). \end{aligned} \tag{2.3}$$

Multivariate Gaussian Distribution

move to PPCA derivation?

Let w and h be jointly Gaussian distributed variables, if the variables are independent, then $p(w,h) = p(w)p(h)$. The joint probability density is thus,

$$p(w,h) = \frac{1}{\sqrt{2\pi\sigma_1^2}\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{1}{2}\left(\frac{w-\mu_1)^2}{\sigma_1^2} + \frac{h-\mu_2)^2}{\sigma_2^2}\right)\right).$$

In matrix form, the joint probability is

$$p(w,h) = \frac{1}{2\pi\sqrt{\sigma_1^2\sigma_2^2}} \exp\left(-\frac{1}{2}\left(\begin{bmatrix} w \\ h \end{bmatrix} - \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}\right)^T \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \left(\begin{bmatrix} w \\ h \end{bmatrix} - \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}\right)\right).$$

For a n-dimensional vector $\boldsymbol{y}$, the joint probability density is expressed as

$$p(\boldsymbol{y}) = \frac{1}{2\pi|\boldsymbol{D}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu})^T\boldsymbol{D}^{-1}(\boldsymbol{y}-\boldsymbol{u})\right), [2, p.78]$$

where $\boldsymbol{D} \in \Re^{n\times 1}$ is the diagonal matrix of the variances Σ. An arbitrary rotation matrix $\boldsymbol{R}^T \in \Re^{n\times n}$ applied to the basis forms a correlated Gaussian,

$$p(\boldsymbol{y}) = \frac{1}{2\pi|\boldsymbol{D}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{R}^T\boldsymbol{y}-\boldsymbol{R}^T\boldsymbol{\mu})^T\boldsymbol{D}^{-1}(\boldsymbol{R}^T\boldsymbol{y}-\boldsymbol{R}^T u)\right).$$

This gives an eigenvalue decomposition of the inverse covariance matrix,

$$\boldsymbol{C}^{-1} = \boldsymbol{R}\boldsymbol{D}^{-1}\boldsymbol{R}^T,$$

thus the covariance matrix,

$$\boldsymbol{C} = \boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^T.$$

As a consequence, if $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$, then $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{W}\boldsymbol{\mu}, \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{W}^T)$.

Given $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\sigma}^2)$, and $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{\epsilon}$, then $\boldsymbol{W}\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{W}\boldsymbol{W}^T)$, $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{0}, bmC)$, where $\boldsymbol{C} = \boldsymbol{W}\boldsymbol{W}^T\beta^{-1}\boldsymbol{I}$, and $\boldsymbol{I}$ is the identity matrix. [12]

2.2.2 Latent Variable Models and Factor Analysis

A latent variable model transforms a set of n d-dimensional observed variables encoded as a data matrix, $\boldsymbol{Y}^T \in \Re^{n\times d}$, to a set of n q-dimensional latent (unobserved) variables, $\boldsymbol{X} \in \Re^{n\times q}$. Latent variables are parsimonious, it is generally the case that $q \ll d$, explaining the original data with fewer variables. A notable latent variable model is that of factor analysis, one that assumes linearity in relation of the observed data set,

$$\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{X} + \boldsymbol{\mu} + \boldsymbol{\epsilon}. \tag{2.4}$$

$\boldsymbol{W}$ represents a matrix that specifies the linear relation between the observed data-space with the latent-space. The parameter μ allows for non-zero mean, and the ϵ parameter represents noise within the model. Standard PCA can be viewed as a variant of factor analysis where the noise parameter is not accounted for. The maximum-likelihood estimates of $\boldsymbol{W}$ will thus generally not correspond to the principal subspace. Tipping and Bishop develop a latent variable model that performs principal component analysis by modelling the parameter ϵ of equation 2.4 as an isotropic, spherical Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \beta^{-1}\boldsymbol{I})$. The conditional probability distribution is thus,

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \mathcal{N}(\boldsymbol{W}\boldsymbol{X} + \boldsymbol{\mu}, \beta^{-1}\boldsymbol{I}). \tag{2.5}$$

The marginal distribution over the latent variables are standard Gaussian, defined as $\boldsymbol{X \sim \mathcal{N}(0, I)}$. The marginal distribution for the observed data $\boldsymbol{Y}$ is obtained by integrating out the latent variables,

$$\boldsymbol{Y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{C}).$$

The observation covariance model is $\boldsymbol{C} = \boldsymbol{W}\boldsymbol{W}^T + \beta^{-1}\boldsymbol{I}$, and corresponding log-likelihood

$$\mathcal{L} = \frac{n}{2}(dln(2\pi) + ln|\boldsymbol{C}| + tr(\boldsymbol{C}^{-1}\boldsymbol{S})), \tag{2.6}$$

$$\boldsymbol{S} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{y}_i - \mu)(\boldsymbol{y}_i - \mu)^T$$

2.2.3 A Probabilistic Model for PCA

Consider a data matrix of n centred d-dimensional vectors $\boldsymbol{Y} = [\boldsymbol{y_1, ..., y_n}]^T$. For each observed data point, $1 \le i \le n$, there is an associated q-dimensional latent variable $\boldsymbol{x_i}$.

The original data can be represented in terms of the latent variable with noise value,

$$\boldsymbol{y_i} = \boldsymbol{W}\boldsymbol{x_i} + \boldsymbol{\epsilon_i}.$$

The matrix $\boldsymbol{W} \in \boldsymbol{\Re^{d\times q}}$ represents the linear relationship between the latent-space with the data-space. The noise values, $\epsilon_i \in \Re^{d\times 1}$, are sampled from a independent spherical Gaussian distribution, $\epsilon_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\beta^{-1}I})$.

By the product property of Gaussian distribution, the likelihood of a data point is thus

$$p(\boldsymbol{y_i}|\boldsymbol{x_i}, \boldsymbol{W}, \boldsymbol{\beta}) = \boldsymbol{\mathcal{N}}(\boldsymbol{y_i}|\boldsymbol{W}\boldsymbol{x_i}, \boldsymbol{\beta^{-1}I}). \tag{2.7}$$

Integrating over the latent variables gives the marginal likelihood,

$$p(\boldsymbol{y_i}|\boldsymbol{W}, \boldsymbol{\beta}) = \int p(\boldsymbol{y_i}|\boldsymbol{x_i}, \boldsymbol{W}, \boldsymbol{\beta})p(\boldsymbol{x_i})d\boldsymbol{x_i}.$$

As the prior of probabilistic PCA is modelled as a standard Gaussian distribution, $p(\boldsymbol{x_i}) = \mathcal{N}(\boldsymbol{x_i}|\boldsymbol{0}, \boldsymbol{I})$, marginalisation of the integral obtains the marginal likelihood of each data point as

$$p(\boldsymbol{y_i}|\boldsymbol{W}, \boldsymbol{\beta}) = \mathcal{N}(\boldsymbol{y_i}|\boldsymbol{0}, \boldsymbol{W}\boldsymbol{W^T} + \boldsymbol{\beta^{-1}}\boldsymbol{I}).$$

Assuming that the data points are independent, the likelihood of the full data set is the product of each marginal likelihood,

$$p(\boldsymbol{Y}|\boldsymbol{W}, \boldsymbol{\beta}) = \prod_{i=1}^{n} p(\boldsymbol{y_i}|\boldsymbol{W}, \boldsymbol{\beta}).$$

2.2.4 The Principal Subspace of PPCA

Tipping and Bishop[16] showed that all potential solutions for $\boldsymbol{W}$, the likelihood (2.6), is of the form

$$\boldsymbol{W} = \boldsymbol{U_q}(\boldsymbol{K_q} - \boldsymbol{\sigma^2 I})^{\frac{1}{2}}\boldsymbol{R}.$$

One particular case of interest is when the likelihood is maximised,

$$\boldsymbol{W}_{ML} = \boldsymbol{U}_q\boldsymbol{L}\boldsymbol{R}, \tag{2.8}$$

$$\boldsymbol{L} = (\Lambda_q - \sigma^2\boldsymbol{I})^{\frac{1}{2}}$$

The matrix $\boldsymbol{U_q} \in \Re^{d\times q}$ contains the column vectors that are the principal eigenvectors, $\Lambda_q = [\lambda_1, ..., \lambda_q]$ represents the diagonal matrix of the corresponding eigenvalues, and $\boldsymbol{R} \in \Re^{q\times q}$ represent an arbitrary orthogonal rotation matrix.

Maximising the likelihood of $\boldsymbol{W}$ by equation 2.8 on the latent variable model defined by equation 2.4 maps the latent-space to the principal subspace of the observed data. Selecting $\boldsymbol{W}_{ML}$, the latent variable model is effectively equivalent to standard principal component analysis.

2.3 Gaussian Process Latent Variable Model

The Gaussian Process Latent Variable Model (GP-LVM) is a non-linear latent variable model derived from a dual of the probabilistic PCA by replacing the inner product kernel with Gaussian processes (Lawrence 2005) [12]. The use of an inner product kernel that allows for non-linear functions enable a non-linear embedding of the data-space to obtain a lower dimension latent-space.

PPCA presents a space of linearly embedded hairstyles, however, it is unlikely that linear embeddings of observed variables for hair is effective for high dimensional data such as hair structure. GP-LVM replaces the covariance matrix with a non-linear kernel, allowing non-linear embedding of the observed variables to capture the complexity of hair structure.

2.3.1 Dual Probabilistic PCA

The dual probabilistic PCA introduced by Lawrence marginalises the parameters, $\boldsymbol{W}$, and optimises with respect to latent variables, $\boldsymbol{X}$. This is the dual approach of the standard probabilistic PCA where the parameters are optimised and the latent variables are marginalised.

First, a conjugate prior to the likelihood of probabilistic PCA (2.7) is taken to be a spherical Gaussian distribution,

$$p(\boldsymbol{W}) = \prod_{i=1}^{d} \mathcal{N}(\boldsymbol{w}_i|\boldsymbol{0}, \boldsymbol{I}).$$

As marginalisation of both $\boldsymbol{W}$ and $\boldsymbol{X}$ is intractable, $\boldsymbol{W}$ is selected for marginalisation as the conjugate prior is Gaussian distributed, thus, it can be integrated analytically. The marginalised likelihood of $\boldsymbol{W}$ is

$$p(\boldsymbol{Y}|\boldsymbol{X}, \beta) = \prod_{i=1}^{d} p(\boldsymbol{y}_{:,i}|\boldsymbol{X}, \beta),$$

The $\boldsymbol{y}_{:,i}$ parameter represents the i^{it} column of $\boldsymbol{Y}$, where

$$p(\boldsymbol{y}_{:,i}|\boldsymbol{X}, \beta) = \mathcal{N}(\boldsymbol{y}_{:,i}|\boldsymbol{0}, \boldsymbol{X}\boldsymbol{X}^T + \beta^{-1}\boldsymbol{I}).$$

The objective function is the log-likelihood

$$L = -\frac{dn}{2}ln2\pi - \frac{d}{2}ln|\boldsymbol{K}| - \frac{1}{2}tr(\boldsymbol{K^{-1}}\boldsymbol{Y}\boldsymbol{Y}^T), \tag{2.9}$$

$$\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^T + \beta^{-1}I.$$

In the original paper, Lawrence found the gradients of the log-likelihood (2.9) with respect to $\boldsymbol{X}$ as

$$\frac{\sigma L}{\sigma \boldsymbol{X}} = \boldsymbol{K^{-1}}\boldsymbol{Y}\boldsymbol{Y}^T\boldsymbol{K^{-1}}\boldsymbol{X} - d\boldsymbol{K^{-1}}\boldsymbol{X}.$$

A stationary point where the gradients are zero is given by

$$\frac{1}{d}\boldsymbol{Y}\boldsymbol{Y}^T\boldsymbol{K}^{-\mathbf{1}}\boldsymbol{X} = \boldsymbol{X}.$$

The values for $\boldsymbol{X}$ which maximise the likelihood are given by singular value decomposition of $\boldsymbol{X}$,

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{L}\boldsymbol{V}^T.$$

$\boldsymbol{U}$ is an $n \times q$ matrix whose orthonormal column vectors are the first eigenvectors of $\boldsymbol{Y}\boldsymbol{Y}^T$. $\boldsymbol{L}$ is a $q \times q$ diagonal matrix of singular values, whose j^{th} element is $l_j = (\lambda_j - \frac{1}{\beta})^{-\frac{1}{2}}$, where λ_j is the eigenvalue associated with the j^{th} eigenvector $d^{-1}\boldsymbol{Y}\boldsymbol{Y}^T$. $\boldsymbol{V}$ is an arbitrary $q \times q$ rotation matrix. Lawrence showed that the eigenvalue problem developed here is equivalent to the eigenvalue problem solved in PPCA, and thus, DPPCA is also effectively equal to standard PCA when the likelihood is maximised.

Dual probabilistic PCA assumes that the output dimensions are linear, independent, and identically distributed. Infringing upon these assumptions derive new probabilistic models.

2.3.2 Gaussian Processes

(O'Hagan, 1992; Williams, 1998)

Gaussian processes are a class of probabilistic models that generalizes a Gaussian probability distribution. The Gaussian process approximates a distribution function in a space of infinite values to a finite range that is tractable.

A Gaussian process first requires specifying a prior, parametrised by a mean and covariance. A simple prior over the space of functions that are linear but corrupted by Gaussian noise of variance $\beta^{-1}\boldsymbol{I}$ is

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T\boldsymbol{x}_j + \beta^{-1}\delta_{ij}. \tag{2.10}$$

$\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are vectors from the space of inputs to the function and $\sigma_1 ij$ is the Knronecker delta. If these inputs were taken from our embedding matrix, $\boldsymbol{X}$, and the covariance function was evaluated at each of the N points, we would recover the covariance matrix of the form

$$\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^T + \beta^{-1}\boldsymbol{I}$$

where the element at ith row and jth column of $\boldsymbol{K}$ is given by the prior (2.10). This is recognised as the covariance associated with each factor of the marginal likelihood for dual probabilistic PCA. The marginal likelihood for dual probabilistic PCA is therefore a product of d independent Gaussian processes. In PCA we are optimising parameters and input positions of a Gaussian process prior distribution where the (linear) covariance function for each dimension is given by $\boldsymbol{K}$.

**Explain better & kernels

2.4 Bayesian Gaussian Process Latent Variable Model

The Bayesian Gaussian Process Latent Variable Model (Bayesian GP-LVM) [17] extends the GP-LVM by variationally integrating out the input variables of the Gaussian process to approximate the marginal likelihood of a fully marginalised model. The approximated marginal likelihood can be used to compute a lower bound that is robust to overfitting. A fully marginalised model establishes a Bayesian perspective that copes well with uncertainty caused by missing data and can automatically determine latent dimensions within the observed data set.

The marginalised likelihood of the GPLVM can be represented in the form

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{i=1}^{d} p(\boldsymbol{y}_{:,i}|\boldsymbol{X})$$

where $\boldsymbol{y}_{:,i}$ represents the i^{th} column of $\boldsymbol{Y}$ and

$$p(\boldsymbol{y}_{:,i}|\boldsymbol{X}) = \mathcal{N}(\boldsymbol{y}_{\boldsymbol{:},\boldsymbol{i}}|\boldsymbol{0}, K_{nn} + \beta^{-1}\boldsymbol{I}_n).$$

K_{nn} is the $n \times n$ covariance matrix defined by the kernel function $k(\boldsymbol{x}, \boldsymbol{x'})$. The latent variable X is assigned a prior density given by the standard Gaussian distribution,

$$p(\boldsymbol{X}) = \prod_{i=1}^{n} \mathcal{N}(\boldsymbol{x_i}|\boldsymbol{0}, \boldsymbol{I_Q}).$$

Each $\boldsymbol{x_i}$ is the i^{th} row of $\boldsymbol{X}$. The joint probability model for the GP-LVM is

$$p(\boldsymbol{Y}, \boldsymbol{X}) = p(\boldsymbol{Y}|\boldsymbol{X})p(\boldsymbol{X})$$

The standard GP-LVM method trains by finding the MAP estimate of $\boldsymbol{X}$ whilst jointly maximizing with respect to the hyperparameters. Bayesian GP-LVM performs variational inference to marginalise the latent variables. This method enables optimisation of the resulting lower bound on the marginal likelihood with respect to the hyperparameters.

2.4.1 Variational Inference

In order to apply variational Bayesian methods to GP-LVM, the latent/input variables that appear non-linearly must first be approximately integrated out.

The marginal likelihood of the observed data is obtained by integrating out the latent variables:

$$p(\boldsymbol{Y}) = \int p(\boldsymbol{Y}|\boldsymbol{X})p(\boldsymbol{X})d\boldsymbol{X}.$$

Computationally, this integration is intractable in practice. Variaitonal Bayesian methods can instead be used by using variational distribution $q(\boldsymbol{X})$ to approximate the posterior distribution over the latent variables, $p(\boldsymbol{X}|\boldsymbol{Y})$.

$$q(\boldsymbol{X}) = \prod_{i=1}^{n} \mathcal{N}(\boldsymbol{x_n}|\boldsymbol{\mu_n}, \boldsymbol{S_n}).$$

The variational parameters are $\boldsymbol{\{\mu_n, S_n\}_{i=1}^n}$ and $\boldsymbol{S_n}$ is a diagonal covariance matrix.

The variational distribution can then be used to obtain a Jensen's lower bound on $\log p(\boldsymbol{Y})$:

$$F(q) = \int q(\boldsymbol{X}) \log \frac{p(\boldsymbol{Y}|\boldsymbol{X})p(\boldsymbol{X})}{q(\boldsymbol{X})} d\boldsymbol{X}$$

$$= \int q(\boldsymbol{X}) \log p(\boldsymbol{Y}|\boldsymbol{X})p(\boldsymbol{X})d\boldsymbol{X} - \int q(x) \log \frac{q(X)}{p(X)} dX$$

$$= \tilde{F}(q) - KL(q||p).$$

The $KL(q||p)$ term is the negative KL divergence between the variational posterior distribution $q(X)$ and the prior distribution p(X) over the latent variables. Since the distributions are Gaussian, the negative KL divergence is tractable. The problematic term is $\tilde{F}(q)$, where variational sparse Gaussian process regression is applied for approximation.

2.5 Formal Definition of 3D Polygon Mesh Representation

Polygon mesh representation of 3D surfaces are composed of vertices, edges, and faces. Let polygon mesh $P = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{F})$, where $\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{F}$ represents the set of vertices, edges, and faces respectively. In practice, polygonal meshes contain more components that affect surface appearance such as texture coordinates and vertex normals, however, the components described are sufficient for geometric processing.

2.5.1 Mesh Vertices

A mesh vertex v is a 3D point of the form $\forall(x, y, z) \in \Re, v = (x, y, z)$. The set of vertices is a point cloud representation of the geometry.

2.5.2 Edges

An edge e is an unordered pair that connects two vertices. Formally, it is described in the form $\forall(v_1, v_2) \in \boldsymbol{V}, e = \{v_1, v_2\}$. Vertices connected by edges form a wireframe of the geometry.

2.5.3 Polygon Faces

A polygon face can be formed from an arbitrary number of vertices $\forall(v_1, v_2, ..., v_n) \in \boldsymbol{V}, f_n = (v_1, v_2, ..., v_n)$, however, in this context we are only concerned with tri-faces $\forall(v_1, v_2, v_3) \in \boldsymbol{V}, f_3 = (v_1, v_2, v_3)$ and quad-faces $\forall(v_1, v_2, v_3, v_4) \in \boldsymbol{V}, f_4 = (v_1, v_2, v_3, v_4)$. Faces describe the geometric surface of an object.

2.5.4 Edge Loops

3D programs often allow edge loop selection which are useful properties of the geometry. An edge loop is defined (on blender) as a set of connected edges that either forms a loop or the end vertices are poles (vertices that do not have edges). Edge loops are useful for extracting more information on the structure of the mesh. [14]

2.5.5 Topology

Tris, quads, poles, etc

2.6 Graph Theory

The construction of 3D polygonal meshes resemble graphs very much. As it turns out, methods of graph theory are useful for processing meshes.

Chapter 3

Project Execution

3.1 Training Data Set

3.1.1 Acquiring Training Data

To begin the training process, it is necessary to obtain suitable 3D hair geometry for use as input training data. Compared to the abundance of images freely available, appropriate 3D models are much less common. Courtesy of Electronic Arts, there exists an active community that produces free custom content for their gaming software - which includes polygonal hair. [9] The acquired files are encoded in the *PACKAGE* format for the latest game of *The Sims* franchise. Open-source community software *s4pe* is used to read the package file and extract geometry in *SIMGEOM* format. [**?**] The *SIMGEOM* format is then converted to *OBJ* format using yet another open-source program *S4CASTools*. [**?**] The geometry extracted is already standardised in scale and orientation.

3.1.2 Repairing Training Data

As mentioned, it is often the case for 3D mesh topology to be organised by quad-faces. In video games, quad-faced meshes are converted to triangle-faced before the rendering pipeline. Mesh reconstruction is performed on all input data to convert the geometry from triangle-face meshes to quad-faces. The Blender API offers a tool to automate this process, executed by the following algorithm:

```
for $i = 0$ upto $n$ do
    $t_i \leftarrow 0$
end
```

Algorithm 3.1: Tri-to-quad using Blender API

Current solutions for converting a triangle meshes to a quad mesh is imperfect. After the procedure, the remaining triangle faces are converted manually. The conversion process alters the geometry very marginally, but the hair representation is preserved - the quad-faced meshes are used for training, and are assumed to be correct in the context that it representative of a valid hairstyle.

3.2 Generative Model of Hair

A generative model is developed for the purpose of approximating the raw data and output generation. Mesh data is difficult to compare as topology and fidelity alters both structure and dimensionality. In order to resolve the data alignment problem, the mesh is approximated to generative parameters of the closest output possible in the generative model. Learning is trained using these generative parameters obtained from approximating the mesh data.

Spherical coordinates Sphere Reference Sample X*Y roots uniformly on surface of sphere From these roots, grow Z points to form strand splines

```
[[0.0,0.0,0.0]]
```

Listing 3.1: Generative Hair Parameter Format.

3.3 Approximating Generative Parameters from Input Data

3.3.1 Parsing OBJ File

Mesh geometry data of *OBJ* files are read into a graph data structure of nodes and edges. Each vertex read swaps the Y and Z axis and the Y axis is negated to align the coordinate space used by Blender and *OBJ* format.

```
for $i = 0$ upto $n$ do
    $t_i \leftarrow 0$
end
```

Algorithm 3.2: Parsing OBJ format.

3.3.2 Spline Estimation

Hair structure estimation begins by splitting the hair mesh into sub-meshes of hair segments determined by connectivity. For each segment, its border edge loops are extracted by finding corner vertices (having exactly two neighbour vertices). A root border edge loop is determined by finding the border edge that is closest to the root. Using the root border edge loop, spline edge loops are selected as strands of hair.

A repair operator performs reparation on extracted splines: attaching floating splines to closest ends of root splines. Splines that are too short are also removed.

3.3.3 Structure Estimation

Each root is associated to a spline by a selection operator. Attributes that make a spline more desirable to a particular root are attributes such as proximity or uniqueness compared to previously selected splines. The motivation is to use a set amount of roots to represent the hair structure without losing too much information.

3.4 Learning a Manifold with Bayesian GP-LVM

Add offset so top of head is "mean", so uncertain hair moves towards the top (avoids hair moving inside head or other unlikely areas) Training with GPLVM. Kernel used. Give algorithm.

3.5 Generation of Output

GPy native matplotlib conflicts with Blender as TKInter is disabled. Plot image of latent space to be imported into image editor of Blender. Pickle model to be imported by Blender's python. Select latent variables from manifold (Blender event API) Read model and predict output by selected latent variables Generate guide splines following format of latent model and generative model Extrude by spline object Rotate by head surface normals Convert to Poly

3.6 Reparation

Normals, clusters, intersections

3.7 Project Management

3.7.1 Source Control

Git is used for source control of the project implementation. The branching feature is useful for separating development of features. Maintaining multiple versions of the code base prevented issues caused by the interaction of incomplete features. The merging and rebasing tools helped conflict resolution. Descriptive atomic commits keep a log of progress and supports roll-back to older versions when necessary. A private repository backup was set up on a hosting service provider to prevent data loss and enable development on multiple machines with ease.

3.7.2 Issue Tracking

Trello

3.7.3 Time-line

Calender. Internal Deadlines.

foo	bar	baz
0	0	0

Table 3.1: This is an example table.

Chapter 4

Critical Evaluation

4.1 Functional Testing

functional testing, including analysis and explanation of failure cases. 5 pages

4.2 Behavioural Testing

behavioural testing, often including analysis of any results that draw some form of conclusion wrt. the aims and objectives 5 pages

4.3 Evaluation

evaluation of options and decisions within the project, and/or a comparison with alternatives. 2 page

4.4 Applications

MMOs, games, etc 1 page

Chapter 5

Conclusion

5.1 Summary

(Re)summarise the main contributions and achievements, in essence summing up the content. 2 pages

5.2 Project Status

Clearly state the current project status (e.g., “X is working, Y is not”) and evaluate what has been achieved with respect to the initial aims and objectives (e.g., “I completed aim X outlined previously, the evidence for this is within Chapter Y”). There is no problem including aims which were not completed, but it is important to evaluate and/or justify why this is the case. 2 pages

5.3 Future Work

Outline any open problems or future plans. Rather than treat this only as an exercise in what you *could* have done given more time, try to focus on any unexplored options or interesting outcomes (e.g., “my experiment for X gave counter-intuitive results, this could be because Y and would form an interesting area for further study” or “users found feature Z of my software difficult to use, which is obvious in hindsight but not during at design stage; to resolve this, I could clearly apply the technique of Smith [7]”). 1 page

Bibliography

[1] W. Baxter and K. Anjyo. *Latent Doodle Space.* Eurographics 2006. Vol. 25, 3., 2006.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[3] N. D. Campbell and J. Kautz. *Learning a manifold of fonts.* ACM Transactions on Graphics (TOG) 33, 4, 91., 2014.

[4] Shaq T.-Wu H. Weng Y. Chai, M. and K Zhou. *AutoHair: Fully automatic hair modeling from a single image.* ACM Trans. Graph. 35, 4 (July), 116:1116:12, 2016.

[5] F. Bevilacqua D. M. D. Carli, C. T. Pozzer and M. C. d Ornellas. *A survey of procedural content generation techniques suitable to game development.* X Simposio Brasileiro de Games e Entretenimento Digital, 2011.

[6] R. Milo. et al. *Number of hairs on human head.* BioNumbers. BNID 101509, 2017.

[7] et al Grochow, K. *Style-Based Inverse Kinematics.* SIGGRAPH '04 ACM SIGGRAPH 2004 Papers. pp.522-531, 2004.

[8] [PCA 1901] http://stat.smmu.edu.cn/history/pearson1901.pdf.

[9] [TSR] https://www.thesimsresource.com/.

[10] [MRI] http://www.stat.columbia.edu/ martin/Papers/STS282.pdf.

[11] I. T. Jolliffe. *Principal Component Analysis.* Springer Series in Statistics, Springer-Verlag, 2nd edition, 2002.

[12] N. D. Lawrence. *Probabilistic non-linear principal component analysis with Gaussian process latent variable models.* The Journal of Machine Learning Research 6, 17831816, 2004.

[13] et al Limpaecher, A. *Real-time Drawing Assistance through Crowdsourcing.* ACM Trans. Graph. 32, 4 (July), 2013.

[14] [Edge loops] https://sites.ualberta.ca/ cwant/blender/derived surfaces.pdf.

[15] C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning.* MIT Press, 2006.

[16] M. E. Tipping and C. M. Bishop. *Probabilistic Principal Component Analysis.* Microsoft Research, Cambridge, UK, 1998.

[17] M. K. Titsias and N. D. Lawrence. *Bayesian Gaussian Process Latent Variable Model.* AISTATS. Vol. 9, 2010.

22

Appendix A

An Example Appendix

Content which is not central to, but may enhance the dissertation can be included in one or more appendices; examples include, but are not limited to

- lengthy mathematical proofs, numerical or graphical results which are summarised in the main body,
- sample or example calculations, and
- results of user studies or questionnaires.

Note that in line with most research conferences, the marking panel is not obliged to read such appendices.