Chapter 1

Outline

Campbell (2016)

In this thesis we will examine the use of information and probability theory measures to track the pose, shape, and appearance of non-rigid articulated objects with multiple independently moving cameras. We demonstrate a hierarchical framework that uses the Hilbert-Schmidt independence criterion, to estimate first the shape and appearance, then the pose, then the activity of the dominant actor. At each level different features are employed, however the same kernel function and the same Hilbert-Schmidt norm are used to estimate the independence.

The most elementary level involves a single static camera tracking a single moving non-articulated, rigid object. This requires only that the appearance be tracked. Note that the appearance is defined as the combination of the gray-level intensity, the distance function, and the motion. This example is essentially TLD, which is a very robust and accurate tracker. I have made a contribution to show that my algorithm is an extension to TLD in this case. My tracker models a target as a collection of points. Each point is naturally endowed with x, y coordinates. The features for each point are the intensity at that pixel, the distance to the nearest edge, and the motion of that point from the previous frame. In each frame we find the set of points that maximises the HSIC with the reference (as well as maximising the self similarity and minimising the similarity with the negative training example). It is important to note that the HSIC f́ailsf́or uniform distributions.

To track the pose we use some sort of optimisation scheme to search through potential poses. Options are: hierarchical search, PSO, LM.

1. Single camera tracking - basically done. 2. Multi camera tracking with known calibration - could be done without a prior labelling of the appearance (i.e just using

the MI scheme from previously and then creating a model on the fly.) 3. unknown calibration - investigate the additional parameters that need to be optimised. 4. activity recognition - constant alpha working (temporal synch + clustering / recognition) 5. DTW - allows unknown alpha, therefore may improve the performance.

1. **Introduction**
   (a) Mixed reality
   (b) Pose, shape, appearance
   (c) Articulated objects
   (d) Markerless motion capture
   (e) Appearance modelling
   (f) Activity recognition
   (g) Independently moving cameras
2. **Probability & Information theory**
   (a) Probability spaces
   (b) Random variables
   (c) Distributions
   (d) Moments
   (e) Expectation, variance, covariance
   (f) Entropy & uncertainty
   (g) Mutual information
3. **Hilbert Schmidt independence criterion**
   (a) Euclidean spaces
   (b) Functional analysis
   (c) Hilbert spaces
   (d) Reproducing property
   (e) RKHS
   (f) Kernels - universality / characteristic
   (g) Covariance Measures

    (h) HSIC measure

4. **Appearance tracking**
5. **Pose estimation**
    (a) Bottom-up
    (b) Top-down: Analysis by synthesis
    (c) Calibration free - motion parameter estimation
6. **Activity recognition**
    (a) Temporal synchronisation
    (b) Dynamic time warping
7. **Easter eggs**
    (a) Reference to kiwi (bird) #
    (b) Cite Einstein
    (c) Cite Erdos
    (d)

1.1 Information Theory

What is information theory? Where did it start? What questions does it seek to answer? What are the important concepts? –¿ capacity of a channel to transmit information –¿ what capacity does the system have to transmit information about data from two images?

The focus of this chapter is on the mathematical concept of information theory, which was first proposed by Claude Shannon in his seminal 1948 paper "A Mathematical Theory of Communication". Central to information theory is the concept of *channel capacity* which describes the ability of a *channel* to transmit information from a source to a destination. In particular, the noisy channel coding theorem specifies an upper bound on the rate at which information can be transmitted through a channel in the presence of noise. Notably, this upper bound depends only on the statistical characteristics of the channel.

1.1.1 Entropy & Uncertainty

How are entropy and uncertainty related? Make the relationship between uncertainty and variance etc quite clear. These links are going to be important later when we talk about HSIC, as it is dependent on the covariance, which is functionally related to the notions of uncertainty and therefore entropy.

1.2 Pose estimation

1.2.1 Bottom up

The principle difference between bottom-up and top-down is the use of the model. In BU estimation we do not know the model of the target. As we track a series of points, however, we can estimate the joint positions by computing the minimum spanning tree on the geodesic distance graph. We could do a graph cut type thing. Initially we want to find which edges are connected to which others. There is an edge between two points if the distances are preserved. We then want to find the joint positions. Similar to Steve's work.

1.2.2 Top down

This is the main approach we are using. We have a known model of the target (i.e all the limb lengths and connections) that we wish to fit to the images(s) the best. For any particular pose we can evaluate the cost, then we just take the best pose. Note that we can think of the model as something that acts to preserve geodesic distances between points, i.e. it lets them move, but only in a way that preserves the shape of the target.

1.2.3 Calibration free

This is currently future work. Track the pose independently in both images. Then we want to find the 3DT pose matrix that best matches the two 2D pose matrices. The principle question is how to propose any given 3DT matrix? probably hard given all the constraints etc.

Chapter 2

Mathematical Preliminaries

The prinicple contribution of this thesis is to demonstate the utility of probability and information theory measures for addressing the correspondence problem in certain computer vision applications. In particular we will examine how the *mutual information* (MI) and the *Hilbert Schmidt independence criterion* (HSIC) can be used to evaluate the relationship between two objects. We begin with an introduction to key concepts in probability theory, then demonstrate how these concepts are central to the development of information theory. We conclude with a discussion of the mutual information. In the following chapter we introduce the reader to Hilbert spaces, then demonstrate how Hilbert spaces relate to earlier concepts in probability and information theory.

2.1 Elements of Probability Theory

2.1.1 Probability spaces

A probability space is a triple $(\Omega, \mathcal{F}, P)$ where Ω is a sample, or outcome space, $\mathcal{F}$ is the event space, and $P : \mathcal{F} \to \mathbb{R}$ is a function that maps events to probabilities. The sample space refers to the set of all outcomes that could occur, while the element $A \in \mathcal{F}$ is a subset of the event space such that $A_i \subseteq \Omega$. The function $P : \mathcal{F} \to \mathbb{R}$ assigns a value to elements of the event space that describe the likelihood of that event occuring. The probability function must satisfy the following properties (2013grayentropy):

- *nonnegative:*
$$P(A) \geq 0, \;\; \forall A \in \mathcal{F}; \tag{2.1}$$
- *normalised:*
$$P(\Omega) = 1; \tag{2.2}$$

- *countably additive:*

$$\text{If } A_i \in \mathcal{F}, i = 1, 2, ...\text{are disjoint, then}$$
$$P(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i). \tag{2.3}$$

The first condition states that probability values must be greater than or equal to zero, while the second and third conditions collectively state that

In the context of computer vision, for example, the sample space is defined by the application. For a set of images the sample space will change depending on factors such as the width and height of the image, the number of channels used to describe each pixel and the range of values each channel can take. The sample space for the set of 8-bit gray-scale images with a fixed width and height is given by the set of all possible configurations of pixel intensities.

2.1.2 Random variables & Distributions

(2013durretprobability) Given our probability space we define a random variable $\mathcal{X}$ to be a function that maps the sample space to some measure space, often the space of real numbers, i.e. $\mathcal{X} : \Omega \to \mathbb{R}$. The random variable $\mathcal{X}$ is then said to induce a distribution on $\mathbb{R}$ by letting the probability that $X = a$ be the measure of the set $\{\omega \in \Omega : \mathcal{X}(\omega) = a\}$, which is denoted by $P(\mathcal{X} = a)$. Intuitively, a random variable is an entity that can take on the value of any element of the sample space with probability given by P. Drawing a single sample from a distribution is then akin to selecting a particular value for the random variable, according to the probability space. In this work we will denote by $P(x)$ the probability that $\mathcal{X}$ takes on the particular value x, i.e. that $P(\mathcal{X} = x)$. Given two random variables, $\mathcal{X}$, and $\mathcal{Y}$, we can define the joint probability $P(\mathcal{X}, \mathcal{Y})$ to be the probability distribution for the set of N ordered pairs $\{(x_i, y_i) : x_i \in \mathcal{X}, y_i \in \mathcal{Y}\}$.

We may choose our random variable to be a gray scale image, for example. In the absence of any domain information it may be the case that every pixel value is equally likely to occur, and therefore we describe our random variable as having a uniform distribution. In contrast, we may assume that our image is drawn from a set of images of the natural world, so that it may contain a plant or an animal. In this case there is some restriction on the particular pixel values that are likely to occur, and we therefore say that our random variable is governed by some unknown probability distribution function that we may wish to know more about. We may be interested in knowing, for instance, whether our image contains a kiwi (reference). In our sample space from which we draw images of the natural world there is a probability function that describes the particular configurations of pixels that give rise to images that appear to be kiwi's. Given an image, we may wish to know whether the distribution that generated the current image is equivalent to the true distribution that generates images of kiwi's. This example is obviously far from trivial, as one cannot simply prescribe a function for describing any and all flightless birds from New Zealand. Describing probability functions and their relationship to each other has been the subject of an extraordinary amount of work. In computer science we typically call this problem pattern recognition, which reflects the fact that our goal is to detect patterns and make inferences on the state of the world.

2.1.3 Moments

In order to understand probability distribution functions we need a language to describe them. This is achieved by computing the *moments* of the distribution. For a discrete univariate probability density function, with an expected value μ_1, the n^{th} moment of the distribution is given by

$$\mu_n = \sum_{i=1} (x_i - \mu_1)^n P(x_i). \tag{2.4}$$

The expectation value, or first moment, of the distribution is given by

$$\mu_1 = \sum_{i=1} x_i P(x_i), \tag{2.5}$$

which is the sum of the individual elements, weighted by their respective probabilities. For a sample of size N the arithmetic mean is given by

$$\bar{x} = \frac{1}{N} \sum_{i=1} x_i. \tag{2.6}$$

The second moment of the distribution, is given by

$$\mu_2 = \sum_{i=1} (x_i - \mu_1)^2 P(x_i), \tag{2.7}$$

which is the expected value of the squared deviation from the mean. For a discrete sample the second moment is defined as the variance and is given by

$$\sigma = \frac{1}{N-1}\sum_{i=1}(x_i - \bar{x}). \tag{2.8}$$

2.2 Information Theory

Since Claude Shannon's seminal 1948 paper "A Mathematical Theory of Communication", the field of information theory has developed rapidly. [Give some examples etc].

Much of the early work concerning information theory was intended to be used to develop methods for telephony, and in particular to understand theoretical limits for which a message could be transmitted across a noisy channel. Shannon sought a function that would describe the uncertainty associated with a particular random variable. Shannon recognised that the *entropy*, previously found in statistical mechanics as Boltmann's entropy equation (reference), was a suitable function for relating the probability distribution of a random variable to its uncertainty (Shannon (2001)). For a sample $X = \{x_i\}$ from a random variable $\mathcal{X}$ with corresponding probability distribution $P = \{p_1, p_2, ..., p_N\}$ the Shannon entropy is

$$H(X) = \sum_{i}^{N} p_i \log{(p_i)}. \tag{2.9}$$

Further, Renyi (1961) demonstrated that the Shannon entropy is the limiting case as $\alpha \to 1$ in a continuous family of functions that satisfy

$$H_\alpha(X) = \frac{1}{1-\alpha}\log\sum_{i}^{N} {p_i}^{\alpha}. \tag{2.10}$$

The Renyi entropy is defined for $\alpha = 2$ as

$$H_2(X) = -\log\sum_{i}^{N} {p_i}^2. \tag{2.11}$$

The joint entropy of the random variables X, and Y, is given by

$$H(X,Y) = \sum_{i}^{N}\sum_{k}^{N} p_{i,k}\log{(p_{i,k})}, \tag{2.12}$$

where $p_{i,k} = P(x_i, y_k)$ and $Y = \{y_i\}$ is a sample from the random variable $\mathcal{Y}$. One of the key characteristics of the entropy is that it is minimised for events that are either very likely or very unlikely to occur. As a consequence, the entropy is maximised when we are most uncertain about the outcome of an event. Given a Bernoulli trial (such as flipping a coin) the entropy is maximised when the probability of either outcome is equal, as can be seen in (bernoulli entropy figure).

One of the principle measures to utilise the entropy is the mutual information (reference (shannon?)), defined by

$$I(X,Y) = H(X) + H(Y) - H(X,Y).0 \tag{2.13}$$

The mutual information is the reduction in uncertainty in X, given observations of Y (and vice versa). Notably, the mutual information can be used to characterise the dependency between X and Y. We can express the mutual information in terms of probabilities as

$$I(X,Y) = \sum_{i}^{N}\sum_{k}^{N} p_{i,k} \log\left(\frac{p_{i,k}}{p_i p_k}\right), \tag{2.14}$$

which is zero if and only if $\mathcal{X}$ and $\mathcal{Y}$ are independent, i.e. if and only if $p_{i,k} = p_i\, p_k$.

We can see that the mutual information will be greater when the respective entropies are maximised and when the joint entropy is maximised. Consider, for example, the outcome of two Bernoulli trials, such as flipping two coins T_1 and T_2 in succession. In order for the mutual information to be maximised we wish to reduce our uncertainty about the outcome of T_2 after observing T_1 by as much as possible. This occurs when our initial uncertainty, i.e our uncertainty about the outcome of T_1, is maximised and our uncertainty about T_2 given T_1 is minimised. We can see that this occurs, for example, when $P(\mathrm{T}_1 = \textit{heads}) = 0.5$ and $P(\mathrm{T}_1 = \textit{heads}, \mathrm{T}_2 = \textit{heads}) = 1$ (with all other joint probabilities equal to zero).

In order to compute the mutual information we need a mechanism to compute the probabilities p_i. It is common to use either histograms or Parzen window techniques to approximate the underlying densities. Given samples $X = \{x_i\}$ and $Y = \{y_i\}$, the histogram is constructed by tallying the frequency of occurance of each particular value. We then compute the entropy as per 2.11.

We compute the Parzen density estimate of the mutual information according to the framework laid out by Wells, Viola, Atsumi, Nakajima, and Kikinis (1996). The Parzen window technique (Duda, Hart, and Stork (2012)) aproximates a probability distribution as a superposition of functions centered on each of the elements of a sample. Any differentiable function that integrates to one can be used in Parzen density estimation. In this work, as in Wells *et al.* (1996), we will use the Gaussian function:

$$G_\psi(x) = \frac{1}{\sqrt{2\pi|\psi|}} \exp\left(\frac{1}{2}x^{\mathrm{T}}\psi^{-1}x\right), \tag{2.15}$$

where ψ is an empirically chosen covariance matrix. In order to approximate the entropy we create two new samples A and B from X, with $N_A, N_B < N$ their respective sizes. The entropy is approximated according to

$$H(X) \approx H^*(X) = \frac{-1}{N_B} \sum_{x_i \in B} \log \frac{1}{N_A} \sum_{x_j \in A} G_\psi(x_i - x_j), \tag{2.16}$$

whereby an individual probability is approximated according to

$$p(x) \approx p^*(x) = \frac{1}{N_A} \sum_{x_j \in A} G_\psi(x - x_j). \tag{2.17}$$

In his paper, Shannon (2001) states that "The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point".

2.3 Functional Analysis

The principle theoretical framework we are using in this thesis is that of reproducing kernel Hilbert spaces (RKHS). Specifically, we are interested in the Hilbert-Schmidt independence criterion (HSIC), which measures the cross-covariance between two RKHS's associated with two random variables. We begin this section with an introduction to Hilbert spaces and to the more general theory of a function space. We then introduce the reproducing property and demonstrate its utility in evaluating functions from a Hilbert space. Finally, we discuss how the theory of RKHS motivates the use of the HSIC to measure independence.

2.3.1 Hilbert spaces

Most readers will be familiar with the concept of the space of real numbers, $\mathcal{R}$, which is an example of a vector space. We can characterise this space as a set of dimensions, whereby points in the space take on a single real value from each of the dimensions and for which operations such as addition and scalar multiplication are permitted. Furthermore, we can endow points in the space with certain properties, such as a norm, which typically represents the size, or magnitude of the vector from the origin to the point. Interestingly, we can also represent *functions* as a vector space. Specifically, a function space $\mathcal{F}$ is defined as a collection of functions which supports the following definitions:

Definition 2.3.1. (from gdurret03) An inner product is a function $\langle \cdot, \cdot \rangle : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{R}$ that satisfies, for every $f, g \in \mathcal{F}$ and $\alpha \in \mathcal{R}$:

- Symmetric: $\langle f, g \rangle = \langle g, f \rangle$
- Linear: $\langle f, g \rangle = \langle g, f \rangle$
- Symmetric: $\langle f, g \rangle = \langle g, f \rangle$

Definition 2.3.2. (from gdurret03) A norm is a non-negative function $\langle \cdot, \cdot \rangle : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{R}$ that satisfies, for every $f, g \in \mathcal{F}$ and $\alpha \in \mathcal{R}$:

- Symmetric: $\langle f, g \rangle = \langle g, f \rangle$
- Symmetric: $\langle f, g \rangle = \langle g, f \rangle$
- Symmetric: $\langle f, g \rangle = \langle g, f \rangle$

.

Note that there exist a rich class of both inner products and norms that characterise many different types of spaces. The Euclidean norm, $||x|| = \sqrt{\sum x_i{}^2}$, and the Euclidean dot product $\langle x, x \rangle = xx^T$ are simple examples. It is important to note that dealing with the abstract notion of a norm or inner product between functions is no more trivial in theory than dealing with the norm or inner product between vectors of real numbers (although it may well be in practice).

In this work we consider a function space known as a Hilbert space.

Chapter 3

Object recognition

The problem of object recognition can generally be divided into two categories, that of image classification (does this image contain the object?) and localisation (where in the image is the object?). The latter naturally extends to tracking, which requires that objects be localised over consecutive frames. In all cases we must be given some form of reference to match. In the problem of classification this is often a class label on images containing the subject of interest. The problem is then to learn a function that will respond in a particular manner to previously unseen images containing that object. Image classification has been dealt with extensively and there exist robust algorithms that achieve very low errors on many classic datasets, for example convolutional deep neural networks (examples?). Although there exist many interesting problems in image classification it will remain outside the scope of this thesis.

Image localisation extends classification by imposing an additional spatial constraint on the output. Rather than simply labelling the class of the image, we now require that the location of the object within the image be specified. In the simplest case, the target is defined by a (typically rectangular) template, from which we can extract a number of features. Tracking is then defined as the problem of finding, for every pair of frames in the video, the transformation that minimises the disagreement between templates in successive images.

State of the art trackers tend to adhere to the following paradigm:

1. In the first frame, specify the location of the target.
2. Given this initial location, train a classifier.

3. For each subsequent frame, find the most probable location of the target and update the classifier.

This approach is beneficial as it removes the need for an extensive library of labelled examples, which is often prohibitive to acquire, and it allows for the classifier to be trained in response to changes in the appearance of the target. One of the most acclaimed tracking algorithms, TLD (tracking-learning-detect), adheres to this online learning paradigm. TLD uses a labelled set of image patches, given by the initial location of the template, from which it trains a classifier. In subsequent frames the output of the tracking process is used to augment the set of labelled examples (both positive and negative) from which the performance of the classifier is improved. Given the initial bounding box location, () sample from the image to generate a number of base classifiers. The input to each classifier is a set of pixels, the locations of which are unique for every base classifier. That is, no two classifiers share any pixel locations in common. This is necessary to ensure the classifiers generate independent predictions. Each of the base classifiers is used to define an ensemble classifier. The ensemble consists of a posterior probability distribution $P_i(y|x)$ with $y \in \{p, n\}$, which defines the probability of observing any base classifier i. An observation of a base classifier is found by making comparisons between each of its pixels and concatenating the binary response into a code, x. The probability of a base classifier corresponding to a template patch is then defined as $P_i(y|x) = \frac{\#p}{\#p+\#n}$ where $\#p$ and $\#n$ are the number of positive and negative patches respectively that were observed for the given response.

For any given bounding box location, TLD then generates a response for each base classifier, which indexes into the ensemble of predictors $P_i(y|x)$. The response to each predictor is summed and the image patch labelled a match if the response is greater than 50%. For every frame, TLD tests every possible bounding box location across a number of scales and records the response of the ensemble. Every patch that scores above 50% is recorded as a detected response. The final output patch is the detected response that lies closest to the output of the optical flow algorithm, which estimates the location of the target by finding the median of the motion vectors that originate in the previous bounding box. TLD is a powerful algorithm because it uses the location of the detected response that was accepted as the true location as a further positive example, and uses the detected examples that were not accepted as negative training examples. This means that over time the algorithm learns to discriminate between the true target and patches that lie close to the decision boundary.

While TLD demonstrates superior performance on a number of training examples, it is not clear that updating the classifier based on the spatial layout of likely positive and negative examples is correct in principle. In particular, the use of heuristics to determine which samples to use to update the classifier can lead to incorrect patch labelling, which, over time, can cause the classifier to drift. Grabner (2008) address this problem by introducing a prior over likely patch labels, which they use to adapt an online boosting algorithm. Similarly, Saffari (2010) use a combination of multiple views and priors to enforce consistency amongst a set of classifiers. Leistner (2009) focus on developing a robust loss function to improve the classifier learning phase. In contrast, Hare (2011) learn a function that predicts the transformation of the target between frames, rather than training a classifier that will give a binary response on an input patch. Their method, which they term STRUCK, utilises a support vector machine (SVM) to estimate the location of the target in a given frame. They use the output of this estimation process to update the SVM, with additional budgeting constraints to restrict the growth in the number of support vectors. In this manner, STRUCK incorporates the patch labelling decision into the classification and learning process, rather than leaving it as an additional step after classification.

One of the principle limitations of the methods introduced above is speed. All of these algorithms rely on costly template representations and update strategies. Moreover, the prohibitive speeds affect the number of examples that can be used to update, affecting the overall learning quality. While these trackers can generally be formulated to operate in near real-time, it is possible to achieve significant improvements in speed without major precision sacrifices using *correlation filters*. Correlation filters operate in a similar fashion to the trackers mentioned above, however they make use of the Fast Fourier Transform (FFT, ref) to improve the tracking speed. Given an input image f and a 2D filter h, the output of the correlation process is an image g. In the ideal case g would be specified by a Gaussian function with a peak at the true target location. The Convolution Theorem states that the correlation can be given by an element-wise multiplication in the Fourier domain (Bolme (2010)). Given the Fourier transforms $\mathrm{F} = \mathcal{F}(f)$ and $\mathrm{H} = \mathcal{F}(h)$ the correlation output is

$$\mathrm{G} = \mathrm{F} \odot \mathrm{H}^* \tag{3.1}$$

where $\odot$ denotes element-wise multiplication and * the complex conjugate. The optimal filter given the training examples is

$$\mathrm{H}^* = \frac{\sum_i \mathrm{G}_i \odot \mathrm{F}_i^*}{\sum_i \mathrm{F}_i \odot \mathrm{F}_i^*} \tag{3.2}$$

where F_i is a training patch in the Fourier domain and G_i is the desired response. For a full derivation of this expression please see Bolme (2010). The desired response is taken to be a 2D Gaussian centred on the training patch. The detected response in frame t is used to update the filter, according to

$$\mathrm{H}_t^* = \frac{\mathrm{A}_t}{\mathrm{B}_t} \tag{3.3}$$

$$\mathrm{A}_t = \mu \mathrm{G}_t \odot \mathrm{F}_t^* + (1-\mu)\mathrm{A}_{t-1} \tag{3.4}$$

$$\mathrm{B}_t = \mu \mathrm{F}_t \odot \mathrm{F}_t^* + (1-\mu)\mathrm{A}_{t-1} \tag{3.5}$$

where μ is a learning rate and $\frac{\mathrm{A}_t}{\mathrm{B}_t}$ is an element-wise division. The MOSSE filter demonstrates robust tracking performance on many standard datasets. Notably, Bolme (2010) report a median frame rate of 669 frames per second with a filter of size 64x64 pixels. Correlation filters can be further improved by using complementary cues to develop a robust model of the target. Bertinetto (2015) introduce the STAPLE tracker, which uses two models in parallel to achieve state of the art tracking performance at approximately 90 frames per second. STAPLE outperforms all of the entries in the VOT2014 challenge (Bertinetto 2015) by integrating a colour histogram and a correlation filter with histograms of oriented gradients (HOG) as the features.

The principle benefit of correlation filters arises from the fact that they can learn from a large amount of training data while maintaining very high frame rates. In particular, correlation filters are trained on data that can be stored in circulant matrices. A circulant matrix is formed by shifting a data vector by one position and storing the resulting vector as a row in the matrix. Notably, circulant matrices can be diagonalised using Fourier transforms and used to provide a rapid solution in ridge regression. Henriques (2014) introduce a kernelised filter which is obtained as a solution to a ridge regression problem. They demonstrate that under certain conditions the kernel matrix is circulant, and therefore a fast solution can be obtained for large training datasets and non-linear features.

Kernelised circulant correlation filters are fast and can be trained on a large number of training examples. They are limited, however, to tracking objects that undergo minimal changes in pose, for some arbitrary definition of minimal. Tracking performance can be improved by using HOG filters, which are robust to changes in pose, however by definition a square bounding box around an articulated target will naturally include background features. We therefore turn to articulated pose estimation, which takes principles from object recognition and applies them to articulated objects. In particular, we demonstrate how the HSIC can be used to develop an articulated tracker that can learn from very few training examples. Although we have not demonstrated how the results of Henriques (2014) could be applied to our work, we consider hypothetical examples of how this may occur. (is HS-DTW the circulant link?)

[how does struck perform against TLD? Is it weaker because it's computationally heavy and can't support as many features?]

[talk about that paper that describes the best trackers].

One of the more subtle benefits of an algorithm such as TLD is that it is invariant to camera motion. This is due to the fact that TLD localises the target relative to the camera reference frame, rather than to a world frame.

Chapter 4

Pose Estimation

In this chapter we discuss our approach to detecting and tracking the pose of an articulated object through a series of images. We begin with a discussion of our approach and demonstrate its performance in a number of different test cases. We conclude with a discussion of prior work in the fields of object recognition, segmentation, and pose estimation.

4.1 Background

Augmented and virtual reality, which we refer to in this thesis as *mixed reality*, are progressing at a rapid pace. This advancement is due largely to improvements in hardware and in our understanding of the human visual system. Collectively, these advancements mean that mixed reality devices are becoming ergonomic and are capable of delivering content that seamlessly blends with the surrounding world. In particular, this progress means that it is now reasonable to develop mixed reality applications for users interacting in dynamic, unpredictable environments. Consequently, there are two situations for which pose tracking is required. The first is to track the user as she navigates an environment in order to deliver the current pose of the device, necessary for delivering content. The second situation involves tracking both people and objects that occupy the environment surrounding the user. Many mixed reality platform providers (references?) address the first problem (of finding the position of the headset) by tracking the internal dynamics of the headset with embedded inertial sensors. For better performance and for more complex behaviours many providers also require that the user operates within a controlled environment that features a number of cameras around its perimeter. These cameras track the user as they move around the environment, which allows for more interactivity between the user and the application, and improves the positional tracking system within the application, which improves the display performance. In most cases, however, this visual tracking system relies on distinctive visual markers placed on the user. In addition, few systems include capabilities to track external actors. Applications would become far more immersive if mixed reality platforms were capable of tracking arbitrary objects in their environment.

4.1.1 Terminology

In this thesis we refer to an *actor* as any object, whether it is a person, a car, or a [endemic NZ bird]. We differentiate between the cases where the actor is *rigid* or *non-rigid*, and whether they are *articulated* or not. A human hand, for instance, is a non-rigid articulated object, due to the fact that the fingers can move independently (typically under certain constraints). In contrast, a solid object such as a desk is not articulated (if one ignores any drawers or movable appendages). A human hand is non-rigid because the skin undergoes complex non-linear motions in response to an underlying rigid motion. In this work however we ignore these non-linear effects and treat the hand as a rigid body object. Similarly we treat a walking person as a rigid-body, articulated actor. In the previous chapter we saw that a bounding box could be used to define a template for an actor and to specify its location in the image. This presents a number of difficulties for tracking articulated actors. An articulated actor occupies only a small percentage of the space within a bounding box. By definition, therefore, a filter that tracks an articulated actor will either fail to correctly identify the target, or the classifier will naturally drift as more and more background features are included in the representation. To mitigate these effects we need to use a more natural representation of articulated actors. We therefore use the following definition:

Definition 4.1.1. Given an object represented as a graph over a collection of nodes, the object is rigid and non-articulated if the Euclidean distances between the nodes are fixed. A rigid object is articulated if the Euclidean distances between nodes are free to change, while the Geodesic distances on the mesh remain constant.

[actor -¿ graph -¿ model]

Actors are represented as a graph $G = (V, E)$. We have that $V = \{v_i\}$ is a set of nodes, $E = \{e_{ij}\}$ is a set of edges and $e_{ij} > 0$ denotes an edge between node i and node j. Our graph is weighted, such that that weight on each edge represents the Euclidean distance between it's two nodes, i.e. $e_{ij} = d(i, j)$. The geodesic distances between any two nodes is the sum of the edges that lie on the shortest path between them. Intuitively this works in the same way as a skeleton. At any point in time separate joints that are not connected through a bone may be any aribitrary Euclidean distance from each other. If we were to find the shortest path between the ends of each of our thumbs, however, we would find that the distance we travel along the bones remains constant at any point in time. A piece of future work, therefore, is to explore whether we can build an unsupervised model by exploiting this property. This idea will be discussed further in Chapter x.

[probably a good place for a diagram]

In order to reduce the complexity of the problem we assume that any actor we wish to track can be specified by a graph G ahead of time. This dramatically reduces our search space, as many of the nodes in our graph are now constrained to move in a particular fashion.

Given a model of the actor, represented as a graph, how do we represent the pose? There are a number of alternative measures for this, such as exponential maps (), quaternions, and Euler angles. We choose instead to use rotation and translation matrices as these are convenient and we are only dealing with small graphs, so we are not operating under memory constraints.

[Go into more detail about how the model is put together and how the rotation matrices work etc. + homogeneous coordinates.]

We wish to find the current pose of the actor in every frame of the video. We are choosing in this work to rely on top-down approaches, whereby we place a model into the scene and record the degree to which it fits the image. We then adjust the pose of the model in such a way that after a certain number of iterations we can be confident the pose of the model will align with the true underlying actor.

We begin by specifying the true location of the actor in the first frame, and from this build our initial model of the actor. Evaluating the model at any given pose means sampling, for every point in the model, the image intensity at that pixel, the distance to the nearest edge and the optical flow information at that point. The similarity between any current pose of our model and the set of positive (or negative) examples is given by the Hilbert-Schmidt independence criterion, as outlined in section x. Using the HSIC as a measure of the similarity between two random variables is advantageous as it captures all of the functional dependencies between the two samples. There are of course many other measures we could use for the similarity, such as the normalised cross correlation (NCC) and the mutual information. The NCC is a linear measure, however, and therefore fails to capture non-linear effects such as dynamic changes in lighting. This is significant, as it is rarely the case that positive changes in one variable directly correlate with positive changes in another variable (or negative changes). Changes tend to be subtle and difficult to discern. Indeed while humans are remarkably capable of identifying which objects are similar to another object, we still struggle to identify subtle differences between two samples. In theory the mutual information is the most robust measure for capturing and understanding these non-linear dependencies. The mutual information requires, however, that the underlying probability distributions of the sample from the random variable be estimated. This can be both computationally expensive and can lead to errors in the approximation. Notably, to represent a probability distribution with a histogram, we typically need the number of samples to be much greater than both the dimensionality of the features we are sampling and the size of their respective domains. We can refer to this problem as *coverage* (curse of dimensionality?) which refers to the fact that we need enough samples to cover the domain of every dimension, otherwise the probability distribution can be unnecessarily concentrated in certain regions.

[1. curse of dimensionality diagram?] [2. example of sampling procedure (intensities, edges etc)] [3. would be good to show example of the energy surface] [4. do a comparison of different optimisation procedures]

References

Campbell, J. (2016). *How to do Anything.* Dunedin, New Zealand: Non-existent Publishers.

Duda, R. O., Hart, P. E., and Stork, D. G. (2012). *Pattern classification.* John Wiley & Sons.

Renyi, A. (1961). On measures of entropy and information.

Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, *5*(1), 3–55.

Wells, W. M., Viola, P., Atsumi, H., Nakajima, S., and Kikinis, R. (1996). Multi-modal volume registration by maximization of mutual information. *Medical image analysis*, *1*(1), 35–51.