UNIVERSITY OF SOUTHAMPTON

An Investigation into Locating Generalised Articulated Pose Models using Self-Supervised Deep Learning

by

Joshua L. Harris

A thesis submitted in partial fulfilment for the degree of Doctor of Philosophy

in the

Faculty of Engineering and Physical Sciences

School of Electronics and Computer Science

March 2024

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Joshua L. Harris

This thesis investigates how self-supervised learning can be used to locate generalised three-dimensional articulated poses from images. We split this difficult problem into its constituent components, self-supervised keypoint detection to find two-dimensional keypoints from images, and self-supervised pose lifting to infer the depth of those points. We frame this problem as a representation learning problem, where keypoints are a spatially constrained representation, and also consider the semantic properties of keypoints when applied to different use cases. We consider how priors are used to resolve an ill-posed problem such as this, before devising a new prior which leverages the rigidity of limbs found in most articulated objects to both locate better keypoints and to improve the lifting of keypoints. We conclude by describing the wider applicability outside of this specific approach, and suggest future work that logically follows on from this thesis.

Contents

List of Figures

List of Tables

Listings

Acknowledgements

Thanks to all those that have helped me in completing this thesis. This includes VLC PhD students, my wonderful supervisory team of Jon and Adam, friends, family, and most of all Clare. I couldn't have finished it without you.

Chapter 1

Introduction

This thesis looks at techniques for applying self-supervised learning to locate generalised three-dimensional articulation models.

Intuitively, as humans, we can see an articulated object, be it a human, an animal, or a robot in just a few different poses and we can determine the articulation model that defines its movement. Can a machine do the same? Our motivation behind taking a self-supervised approach is that labelling data is expensive, and occasionally inconsistent, especially for pose estimation. It is common to see labelling issues when an image of an articulated model contains occlusion, making it difficult to place ground truth labels. Being able to take a single approach to determine the articulation model of a subject in an image without the need for labelling the dataset beforehand is not just of academic interest. We believe being able to determine 3D structures would be beneficial for many downstream applications including action recognition, surveillance and motion capture for media production.

1.1 Articulation Models

We define articulation models as a general term for a set of keypoints based on an underlying model built of articulated elements with an additional restriction that points are connected by rigid limbs. We create an articulation model by combining an ordered set of keypoints, which represent joints and endpoints of limbs, and a connection matrix that defines which keypoints are connected by a rigid spatial constraint. Through the majority of this thesis, we are interested in 3D articulation models as these not only contain more information about the world, but also give us more information to estimate connectivity.

1.2 Thematic Breakdown

This section breaks down the main themes contained within this thesis and gives an overview of each. This is not a comprehensive list of themes, but gives an outline of the ideas that will be discussed in detail later.

1.2.1 Self-Supervised Learning

Tsai et al. [103] state that self-supervised learning "adopts self-defined signals as supervision and uses the learned representation for downstream tasks". Being able to train a network without requiring labelled data gives us the ability to use any suitable unlabelled dataset, and greatly speeds up the data collection step required before we can start training a new model.

There is an argument that it would be more appropriate to apply semi-supervised learning to this problem, as it can usually get better results by using a combination of a small labelled dataset and a larger unlabelled dataset. But as our focus is generalisability, we would like to remove any unnecessary priors that would be included via the labelling of data, and in turn this lets us apply our approach to any dataset. We are interested in applying self-supervised techniques throughout this thesis, so will be a common theme in Chapters 3, 4, 5, and 6.

1.2.2 Keypoint and Landmark Detection

Keypoints have long been an essential component in the computer vision field, traditionally being used as an anchor point for local descriptor algorithms, but in recent years have become less critical due to the rise in popularity of learned convolutional approaches. But this is not to say that keypoints no longer have their uses. As they are lightweight and easily interpreted it makes keypoints ideal for capturing the shape of a given object in an image. Keypoints are trivial to work with and from them we can derive a wide variety of information about the structure of something in an image.

Landmarks are keypoints with the dedicated purpose of locating a specific feature, most commonly positions in a pose. Whereas a keypoint cannot be incorrect, a landmark is only correct if it aligns with a ground truth point. There are also other subtle differences, usually we assume that landmarks are consistent in their ordering, meaning that we can correlate them between images, but this is not always the case with keypoints. Terminology of keypoints and landmarks are often confused, leading us to one of our research questions, RQ3, to clarify what we mean by keypoint in the context of each section of this thesis. We are primarily looking at this work, in a self supervised and multi-task environment, in Chapter 4.

1.2.3 Pose Estimation

Pose estimation is an application of keypoint or landmark detection, which aims to locate consistent keypoints on a dynamic object. Most commonly objects of interest are humans, faces or hands, but occasionally animals or robots. Typically these approaches only work for the articulation model that is has been tailored for, as strong priors are often used to achieve good results. This thesis is interested in a generalised approach that uses self-supervised learning alongside weak prior knowledge that is not specific to the underlying articulation model. We look at generalised keypoint detection, as an abstraction of pose estimation, in Chapter 4, but go deeper into pose estimation on articulation models in Chapters 5 and 6.

1.2.4 2D to 3D Keypoint Lifting

As humans, we can look at a 2D representation of a 3D structure and infer the depth to develop an internal 3D model. Contained within the human visual processing system are ways of using visual cues to determine the depth of an object, even though the world is primarily viewed as a 2D representation. Taking inspiration from this, we will be looking at how to learn to estimate depth using self-supervised learning, using only 2D inputs taken from single view, monocular images. Learning 3D points from 2D data has massive implications for downstream tasks over simply a two-dimensional pose, and can assist with problems such as occlusion. This thesis will primarily focus on keypoint depth estimation in Chapter 3, but will also continue on this theme in Chapters 5 and 6.

1.2.5 Generative Adversarial Learning

The use of a discriminator network that is optimised to detect real and fake data points is a powerful tool in deep learning. Our work applies this theory to solve the problem of self-supervised depth estimation by imagining 3D poses from different angles and using the discriminator to differentiate the original 2D poses from our 3D estimated ones viewed from random angles, as shown in Chapter 3. This is not to be confused with Adversarial Machine Learning, which focuses on robustness against adversarial examples.

1.2.6 Representation Learning

Self-supervised representation learning has given us an insight into how we can use deep learning to extract information found within an image without the need for labelled data [17, 29, 72]. More specifically, our interests for this thesis are in spatial representations, primarily using keypoints as a spatial constraint to capture structural information

about an image in a lightweight and interpretable representation. We can then map our spatial representation onto an articulation model by assigning concrete interpretation for each keypoint, and applying additional relationships between pairs of keypoints, dictated by a connection matrix. Our spatial representations are first defined and used in Chapter 4, but continue to be used later in Chapters 5 and 6.

1.2.7 Multi-task Learning

Multi-task learning has been proven to give impressive generalisation results across a variety of domains, and we look to apply it within our research on self-supervised keypoint detection, covered in Chapter 4. We use a range of carefully selected, self-supervised tasks that use predicted keypoints and unlabelled images as an input, to provide multiple training signals to the keypoint regressor network as intuitively, a varied set of tasks will allow for generalised keypoints to be estimated. The generalisation found using this approach has wider impact, but in our example, will allow us to determine keypoints on a wide range of articulation models, without overfitting the keypoints to solve one specific task.

1.3 Motivation for Learning Self-Supervised Articulation Models

Our motivations for generalising the detection of articulation models using self-supervised learning techniques are as follows:

- Finding an approach that works for a wide range of articulated objects, including but not limited to, humans, animals and robots.
- Leveraging only information found in images so that we can apply this technique to any dataset, without relying on labelled data. This can be clearly extended to image sequences, but in order to allow one method to apply to a wide range of datasets, we do not require sequential information.
- Locating generalised spatial representations gives us the opportunity to transfer knowledge to solve novel downstream tasks, even if they are not used during training.

1.4 Research Questions

Following on from our motivations and to frame this thesis, we will establish a set of research questions that we seek to answer throughout the content of the thesis.

RQ1. How can we learn a self-supervised articulation model?

Previous approaches for pose estimation rely on heavy prior knowledge of the world, commonly as predetermined articulation models such as Active Shape Models [20], or large labelled datasets [48]. But we are interested in estimating articulation models leveraging only self-defined signals, and using minimal prior knowledge.

This research question is broad, but shapes the entire thesis. Naturally with a difficult problem, we will break it down into smaller, and more manageable, sub-problems, discussing our approach to solving each, before combining each section into one full approach.

RQ2. How can we effectively represent spatial constraints?

In a generalised case, a spatial representation can be simply a set of keypoints. We can assign additional desirable qualities for our keypoints too, such as consistency between examples and robustness to textural change in the image space. We may also choose to represent a spatial constraint as a list, if the index that a keypoint lies in convey important information, or a set, if we deem two permutations of keypoints to be identical. When looking at other, more applied contexts, such as for articulation models, the relationships between keypoints convey essential information that we would like to capture too.

RQ3. In the context of articulation models, what is a keypoint?

Keypoints are one of the most basic spatial representations, but the term can vary greatly with context. In traditional contexts, a keypoint is the root for a local descriptor algorithm but in pose estimation it designates a landmark that we aim to identify, whereas in a self-supervised learning setting, keypoints can be used simply as a spatial representation. We also need to consider if we are looking for landmarks, where an ideal location is desired, or keypoints, where the most information captured about an image is desired. We will aim to answer this research question in each section to apply context to the term keypoint.

RQ4. What is the minimum prior information required to solve self-supervised articulation model estimation?

In a self-supervised environment, we lack ground truth data to learn from, so it is common to supplement an approach with a prior in order to find good results. Selecting the correct prior is essential for success, but selecting the right one comes with difficulties. Determining the best way to leverage the information contained within the dataset, without making too many assumptions about the data that would inhibit generalisability.

We have selected these research questions because we know intuitively that as humans can determine a 3D articulation model without the requirement of ground truth 3D keypoints. This motivates the use of self-supervised learning, as biologically inspired algorithms should be able to operate with the same information that a human can use to solve the same problem.

1.5 Contributions

Here we will briefly discuss original contributions made during the course of this thesis:

1.5.1 Pose Lifting with Bottlenecked Auto-Encoders

As discussed in detail in Section 3.4, we take inspiration from similar approaches to pose lifting using generative adversarial learning and contributed a spatial bottleneck to compress information before estimating a 3D pose. We achieve good results, but find that training can be unreliable due to the nature of adversarial learning and the occasional occurrence of the inverse pose problem.

1.5.2 Multi-Task Approach to Generalised Self-Supervised Keypoint Detection

Our approach to generalised keypoint detection, leveraging the generalisability of multi-task learning is a valuable contribution to spatially constrained representation learning, and is covered in detail in Section 4.4. The main contribution is the generalisability achieved when learning keypoints that satisfy multiple tasks, to the extent that we can test on a disjoint dataset to the training set and still achieve reliable results.

1.5.3 Differentiable Minimum Spanning Tree

The minimum spanning tree algorithm is ubiquitous across many applications, but being able to pass gradients through it allows for more applications in the deep learning field. As the standard implementation of the Prim's algorithm is not differentiable, our contribution found in Section 5.3.1 works around this using an approximation, enabling the ability for use within a neural network.

1.5.4 2D to 3D Pose Lifting using Rigid Bone Prior

Existing 2D to 3D Pose Lifting approaches have shown success with adversarial learning, but can run into the inverse pose problem [61], and can be unreliable during training due to the nature of adversarial learning. Our approach using a self-consistency loss and a rigid bone prior, as covered in Section 5.4, allows for consistent lifting without the difficultly of using generative adversarial learning, and achieving good accuracy.

1.6 Keypoint Clarification

This section will briefly clarify the differences between three key words used in this thesis; keypoint, landmark, and pose within the context of finding generalised articulation models, and why the number of keypoints is a fixed parameter throughout the thesis. Definitions of the keypoint related terms that will be used throughout the thesis are as follows:

- Keypoint: A selected point within an image which is robust and repeatable, and can be used in a list or set to build a generalised spatial representation.
- Landmark: A specialised keypoint that should be placed on a specific feature or location within the image.
- Pose: A collection of landmarks that represent an object in the image and is typically stored in a list where the index denotes the feature that each landmark correlates to.

Throughout this thesis, we will be fixing the number of keypoints used for each experiment (on a per-experiment basis), including those that should be found by our keypoint detection networks, and used in the lifting from 2D to 3D keypoints. The implications of this assumption is discussed in further detail in Section 7.4.2.1.

1.7 Breakdown of Thesis Structure

Tackling a tough problem such as this requires a logical approach. To make our aims achievable, we have decided to break down the problem into achievable sub-problems. Our pipeline for this project has a clear separation, between the stage that derives keypoints from images and the stage that infers depth from those 2D keypoints. There is also an intermediary step between the two, which is made simpler if we have derived an articulation model. Combining the two is not trivial, and our attempts at producing a full pipeline is covered in Chapter 6.

The remainder of this thesis is broken down into the following chapters:

Chapter 2: Literature Review

We will review the related literature in the areas that this thesis covers. We aim to cover a mix of traditional and modern approaches, and discuss the implications of previous work on the contents of this thesis. More specific related work to each chapter will be covered at the start of the relevant chapter.

Chapter 3: Inferring Depth of 2D Keypoints

This chapter will look at existing approaches for inferring depth of dynamic objects, based on 2D keypoints, before contributing our new approach that lifts poses using an adversarial loss function alongside a spatial bottleneck.

Chapter 4: Generalised Keypoint Detection using Self-Supervised Learning

In this chapter, we look at formulating self-supervised keypoint detection as a representation learning problem using a spatial constraint. We look at locating keypoints in a generalised way, using them as a generic spatial representation that applies to any image, and implement a multi-task learning approach to encourage generalised keypoints.

Chapter 5: Using Bone Rigidity as a Generic Prior

After considering the results of the previous chapters, we aim to design and implement a generalised articulation prior to achieve better results without losing generalisability. We then look to return to the problems faced in Chapters 3 and 4, and implement solutions using this prior.

Chapter 6: Towards Learning Articulation Models

This chapter aims to link together the approaches used in Chapters 3, 4 and 5 to create a full pipeline that should be able to take an image and predict a 3D articulation model, all in a self-supervised manner. Results for an entire pipeline are not yet satisfactory, but good progress towards this aim has been made.

Chapter 7: Conclusions and Future Work

Finally, we will discuss the work covered in this thesis, draw conclusions and talk about future work that naturally follows on from this research.

Chapter 2

Related Work

This chapter looks into related work across the range of sub-fields covered in this thesis. We look at existing approaches to solve pose estimation, including both supervised and unsupervised methods. We discuss both modern deep learning based pose estimation, as well as older approaches that apply probabilistic models including shape models. We then consider keypoints and their general role within computer vision and how they align with pose estimation and spatial representations. We then finish this section by discussing representation learning and how spatially constrained representations can be located via keypoint detection. This chapter is necessarily broad to set the scene of the thesis. More focused reviews of relevant literature can be found within the following chapters.

2.1 Pose Estimation

The overarching goal of this thesis is generalising pose estimation in articulated models, taking heavy influences from the sub-fields of self-supervised learning and generalisation. While pose estimation concerns the positioning and shape of any object, static or dynamic, there has historically been a large research focus on human, hand or face poses. This is mainly due to the downstream implications of locating the poses of human features, with numerous downstream applications including biometrics, surveillance, virtual reality and healthcare [22]. Our interest in articulated objects will lead us to focus mostly on human pose estimation, before considering other types of articulation, including animal poses.

2.1.1 Classical Pose Estimation

Pose estimation has classically been a well studied area of computer vision and has been tackled since the very early days of the field. Fischler and Elschlager [30] in the early

1970s aimed to compute an approach for locating a visual object, facial features in their examples, upon being given an image. Their images are represented as a low resolution matrix, due to computational restrictions of 1970s hardware, but they succeed to locate facial features when given specific rules for the patterns that identify them.

Published in 2015, Liu et al. [74] review traditional human pose estimation approaches that use body part parsing at their core. The scope of work covered in this review is vast, covering single and multi person estimation, in videos and in images, and using monocular or multi-view data. They identify the future trend using unsupervised and semi-supervised learning for parsing body parts. Their final take-away is that there is a large gap between theoretical research and real-world applications, a key point to consider as we implement approaches to finding articulated poses.

Bourdev and Malik [5] use poselets, described as "parts that are tightly clustered in both appearance and configuration space" for their 3D human pose estimation approach. This means they consider both structure of body parts, but also appearance when selecting their poselets. Their pipeline proposes poselets from each input image, and pass them through a Max Margin Hough Transform to learn to weight correctly located poselets. Their other main contribution is H3D, a 3D human pose dataset which provides 3D annotations for in-the-wild images of people.

Other well established approaches to tackling this problem opt for using shape models. Traditional shape modelling techniques use Procrustes analysis to align a set of training examples before using PCA to learn the low-dimensional representation to capture valid shapes. Caunce et al. [8, 9], Rogez et al. [92] all look into using shape modelling to find accurate shape models of the human face, and capturing facial articulation. Hasler et al. [40] look at learning a shape model using high detailed 3D meshes, capturing both the pose and appearance of the subject, allowing for a generative model to create novel examples of 3D models with new shapes and poses, demonstrated in Figure 2.1.

Cootes et al. [20] pioneered the concept of an Active Shape Model, which using the distribution of a relatively small training set learns a point distribution model of the shape of a dynamic object. This shape model can then be fit onto an image using an iterative search process, and is capable of robustly fitting to images and is tolerant to some occlusion. They demonstrate their approach on echocardiogram scans of hearts and hands, where a small labelled training set of each is used to learn a point distribution model, which can then be fit onto new unseen examples.

2.1.2 Modern Approaches

Recent attempts at pose estimation manage to tackle the problem with impressive results using deep learning. Typically in a supervised environment, pose estimation is a regression task where a network is trained to minimise the L_1 or L_2 error between the

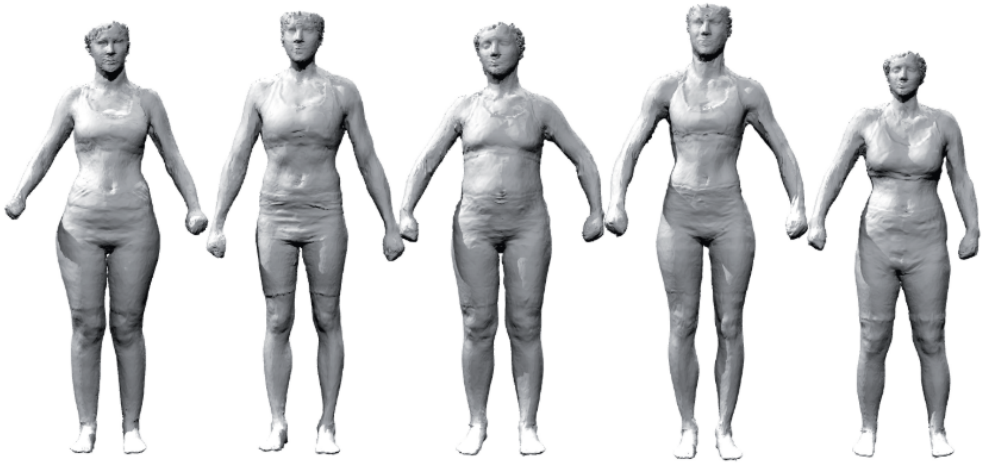


Figure 2.1: Poses and shapes in the form of volumetric models that have been randomly sampled from a learnt distribution of poses from a dataset of people. Taken from Hasler et al. [40].

output and the labelled keypoints. Dang et al. [22] survey deep learning approaches to 2D pose estimation, looking at both single-person and multi-person approaches, while also discussing datasets, metrics and unsolved challenges in this field. They find that convolutional neural networks are typically used, with either linear layers finishing with two times the number of keypoints [102] or a soft-arg-max function to derive numerical keypoints from the peaks of resultant heatmaps from the convolutional layers [77].

A review of monocular deep learning based human pose estimation approaches is carried out by Chen et al. [15], who split the field up into single- and multi-person pose estimation and in both two and three dimensions. They identify that while some pose estimation approaches are keypoint or skeletal based, other look for contour-based models that outline the person or people identified or volumetric approaches which fit a 3D model onto the image to provide 3D pose estimation. They also identify two major strategies within the multi-person pose estimation space. The first is top down, where object detection is used to identify how many people are in the image, cropping those with bounding boxes, and running pose estimation on each of those. The second is bottom up where body parts are located across the whole image, and people are identified via linking those parts together via joint candidate grouping.

Toshev and Szegedy [102] were among the first to apply deep learning techniques to pose estimation and laid the foundations for future deep learning based pose estimation approaches. Their implementation used simple convolutional neural networks with fully connected heads to regress directly to the desired number of keypoints, but cascade multiple networks and feed in a cropped version of the input image around each predicted keypoint in order to improve the accuracy of detection. At time of publication, their approach was state-of-the-art for the Frames Labelled in Cinema dataset [93] and the

Leeds Sports Dataset [57, 58], outperforming traditional approaches that did not leverage the power of deep learning.

In the area of multi-person pose estimation, Cao et al. [6] have been successful for images with many humans, such as the MPII human multi-person dataset, while also running in real-time. They manage this by using part affinity fields to determine which body part belongs to which individual, making use of the image data around each point.

Deep learning based pose estimation does not need to be through supervised learning. Jakab et al. [50] demonstrate a novel self-supervised approach to human pose estimation that makes use of unaligned data, leveraging images and keypoints from unpaired datasets, allowing for the usage of any unlabelled image dataset as long as the unpaired keypoints represent a good range of poses. They demonstrate state-of-the-art unsupervised landmark detection performance, while maintaining an approach that is applicable to a wide range of datasets, considering the strength of the prior used. While this approach demonstrates good performance without direct supervision, our focus is on generalisability, and requiring a labelled dataset of poses for each category of articulation model would take away from our fully self-supervised and generalised approach.

2.1.3 3D Pose Estimation

Wang et al. [107] have surveyed the space of 3D human pose estimation. They cover not only single human estimation, but also multi-person, and consider approaches that operate on singular frames and on sequences. Additionally, datasets, evaluation metrics and performance analysis comparing similar approaches have been detailed and future potential developments outlined. They make the distinction between skeleton based models, skinned multi-person linear model and surface-based models.

Ji et al. [54] have also performed a survey of monocular 3D pose estimation. As part of their analysis, they break down the different approaches into two categories, direct regression of 3D pose and cascaded approaches that regress to 2D pose before performing 3D pose lifting. They find that the general performance of the latter category performs best. This survey, along with other previous approaches to 3D pose estimation, combined with the knowledge of the difficulties in training neural networks using self-supervised learning techniques, lead us to the decision to pursue an approach that splits the pipeline into 2D pose estimation and 2D-3D pose lifting. This will be covered in more detail in Chapter 6.

Pavlakos et al. [88] use a supervised approach that manages to identify shape and 3D pose from a single colour image as a 3D mesh. This heavily supervised technique works as a pipeline, finding the keypoints of joints as heatmaps, and then inferring the 3D structure.

Chen and Ramanan [11] show promising results in 3D pose estimation, and do so by estimating 2D keypoints from the image and then matching to the closest example from a dataset of known valid 3D human poses. This puts the emphasis on locating accurate 2D poses in order to correctly make a match, and exemplar-based matching is done via a reprojection error on nearest matched 3D example. This method ensures that every prediction gives a realistic human pose, but does have limitations to the finite set of known poses, so would perhaps not succeed if given novel examples of unusual looking poses, lacking robustness to out of distribution poses. This lack of generalisation means that there would be failure cases in downstream applications with high variance of poses such as analysing gymnastics data.

Jenni and Favaro [53] solve the problem of self-supervised 3D pose estimation leveraging the multi-view data found in the Human3.6m dataset, but also deviate from strict self-supervision by using a small annotated dataset to fine-tune their self-supervised network. Their self-supervised training uses a classification task which aims to determine if a pair of frames are of the same scene but from a different angle, if a pair of frames are unsynchronised, or if one frame is a flipped version of the other, using a Siamese architecture to derive representations from pairs of images. They then use the Siamese network trained on this self-supervised task as the initial step of a pipeline where representations are derived from images and those representations are regressed onto a small training set of ground truth 3D keypoints, finding comparable results to similar approaches that use full supervision.

When considering depth perception using deep learning, we may want to consider biologically inspired approaches. As humans, our internal depth perception is not foolproof, optical illusions that play on our methods of inferring depth are numerous, many common optical illusions work around exploiting the cues that the human visual system uses to determine depth [108]. From this knowledge, we can see pitfalls of human vision, and may help explain and diagnose similar errors when developing and training a depth estimation neural network. This work has helped inform the addition of an additional self-consistency loss function as seen in Section 5.4, which aims to circumvent errors in depth perception that can occur when insufficient information is available to estimating depth.

2.1.4 Animal Pose

While a great research focus is on pose estimation for humans, being full body poses, faces and hands, because of their plentiful applications, our motivations are to generalise our approach to consider a wider range of articulation models, so we must also consider approaches that are not limited to human pose. Yu et al. [113] have recently collated a dataset for general pose estimation in the wild, taking a wide range of animals of different taxonomies and labelling them alongside providing a benchmark for their preliminary

results. Such a dataset poses a challenge that is relatively unexplored within pose estimation research in a supervised scenario, so a fully optimised solution to the problem has yet to be found. Pereira et al. [89] look at how to quickly detect animal pose when dealing with new and limited data, finding that with only 100 labelled images, they can achieve 90% of their baseline performance. This approach lends itself to unseen downstream applications as labelling a small set of images is a small overhead to get results on an unseen subject. A toolkit has been put together by Graving et al. [36] as a starting block for others who work in the area of animal pose estimation, operating both in the wild and in laboratory settings, with the additional capability to deal with multi-animal images. Liu et al. [73] look at leveraging video information and optical flow in their multi-frame approach for improving the performance of animal pose estimation. Their base model is flexible to the variance found within poses of animals, and once a pose is derived from a frame using their baseline model, multiple frames are passed through and optical flow model to derive the movement from each input video. The optical flow data is then able to correct mistakes in the pose estimation from the baseline model, resulting in a model capable of state-of-the-art results at time of publication.

2.1.5 Challenges

As briefly mentioned by Dang et al. [22], occlusion and self-occlusion are still challenges in human pose estimation. Jalal and Singh [52] have identified three distinct types of occlusion; self-occlusion, where part of an object blocks itself, inter-object occlusion, where one object in the image occludes another, and background occlusion, in which some of the background occludes the object.

While just one example of many, Chen et al. [12] find impressive results within 2D to 3D pose lifting using a self consistency prior, but attribute most of their failure cases to self-occlusion. Solutions to adding robustness to 3D pose estimation when using deep heatmaps have been proposed by Oberweger et al. [86]. They note that convolutional based approaches are highly sensitive to occlusions, but they add robustness via augmenting training data with random occluding objects taken from the LineMOD dataset. This approach is agnostic to any training dataset, and has been demonstrated to be more effective in adding robustness than using random two-dimensional geometric occlusions.

Another challenge in this space, also mentioned by Dang et al. [22], is that existing datasets are large but contain a lack of balance towards rare poses, and no approaches manage to fix this dataset imbalance. They mention data augmentation as a solution, with one option using GANs to generate new data points or unlabelled data to augment in a semi-supervised fashion. Another proposed solution uses special training procedures to learn a better network when using biased data.

Li et al. [65] use an evolutionary algorithm as an approach to remove data bias against extreme poses in 3D human pose estimation. They do this by representing poses as trees, and using evolutionary strategies, augmenting their pose dataset with new data points built from pairs of poses that have been crossed-over and mutated. Their results showed state-of-the-art performance on fully-supervised 3D pose estimation at time of publication, showing the power of enriching a dataset using this approach.

Jiang et al. [55] take a similar strategy, but use a synthetic dataset to level the data bias. Using a conditional variational auto-encoder, they map an existing dataset into a smooth latent space, and then generate new plausible examples by sampling random points from that latent space within a given distribution. The authors of this paper were able to get state-of-the-art results on the Human3.6m dataset at time of publication. However, there is some concern that training a network using synthetic data will result in poor performance on real-world data, and as Human3.6m is data captured in a constrained environment, this approach may not generalise as well as other approaches.

Another challenge identified by Dang et al. [22] is real-time processing of information, which inhibits the usefulness of these approaches when used for real world problems that require the ability to run in real-time, especially when considering video data. To aid this, each stage of the pipeline must be carefully implemented to reduce processing time.

2.1.6 Summary

Pose estimation is a problem that has been researched for approximately 50 years, yet robustly locating poses remains a problem. We have looked at traditional, modern and 3D human pose estimation, while also considering animal poses and identifying the challenges faced when locating poses. When researching and implementing our generalised articulated pose estimation approach, we should consider these challenges and think how best to overcome them, and also take inspiration from previous successful approaches, notably splitting a 3D approach into 2D keypoint estimation before passing to a depth estimator.

2.2 Keypoints

The next major focus of our related work section considers keypoints and their usages both recently and historically, and their relevance to pose estimation. We introduced the idea of keypoints and the terminology used in Section 1.2.2, while outlining the nuanced differences between keypoints and landmarks, the former being generic points and the latter being specific areas on an image to identify.

2.2.1 Traditional Keypoint Detection

Keypoint detection has been at the core of computer vision techniques throughout the history of the field. Traditionally keypoint detection was the first step in detecting local features [38, 76, 84, 94], before analysing these extracted features to solve a task. Typically, the algorithms used to locate keypoints are designed to look for interest points, which are areas of high saliency with the image, for example; corners [38], blobs [76] or regions [82]. Good keypoints should be robust to noise in the image, and repeatable, so similar points are found in similar images. While once ubiquitous, these techniques are much less common today, and focus has switched to using deep learning techniques to train networks to find keypoints. However, there are cases where keypoints are still relevant, albeit mostly in traditional computer vision pipelines where explainability is desired or in safety critical environments. But these traditional approaches are not redundant in recent times, and further research is going into traditional keypoint detection. Cho et al. [16] have recently investigated the benefits of higher order Laplacian of Gaussian keypoint detection techniques and find that when used in combination with higher order Difference of Gaussian, show improvements in multiple keypoint-based computer vision problems.

2.2.2 Deep Learning Keypoint Detection

Keypoint detection still has relevance in the current deep learning research climate. Typically when we consider deep learning based keypoint detection, we are interested in convolutional neural networks in which convolutional kernels have been learned via large amounts of data to identify desirable points.

Neural network architectures of interest in the field of keypoint detection are hourglass and stacked hourglass networks. Examples include unsupervised keypoint detection [118], fully supervised human pose estimation [85] and locating 3D human pose [14]. The power of an hourglass network comes from its ability to fuse local and global information about an image to locate local image features while using the context of where the feature is located within the image. This is in contrast to traditional keypoint detection approaches which rely on local context only. Due to the merits of this architecture, this is something that we will be using as inspiration as we develop and implement our own solutions.

There is a shifting focus towards capturing spatially constrained representations from images instead of selecting keypoints for the basis of a local descriptor algorithm. Thewlis et al. [98, 99, 100] have investigated capturing shape with unsupervised learning in multiple domains. The basis of this concept is to train a network to find invariant points between an original image and an artificially warped version of it, with the aim of training a network to robustly locate keypoints. Zhang et al. [118] take this idea further by adding

a reconstruction task as a training objective that uses local descriptors to reconstruct the image alongside checking for consistency between non-linear image warps. Kulkarni et al. [63] apply a similar method to the control domain, leveraging sequential information to find robust points. A common technique used with self-supervised keypoint detection is to append a linear regressor to the network and training it with a small amount of ground truth data in order to convert keypoints into specific landmarks for pose estimation tasks [100, 118]. While not strictly self-supervised, this additional step gives a good indication of how to convert a generalised representation into landmarks with the goal of locating specific features on the object in the image. This step does however prove the saliency of the keypoints located from the images, regression would not be successful unless a good spatial representation had been identified.

Jakab et al. [51] present a semi-supervised solution that uses disjoint images and labels that belong to the same category. While results are impressive, this technique requires keypoint shapes which are available for some categories of objects such as human pose [48], but fails to generalise to our desired extent, where we wish to find spatial representations of any given articulated object.

While current unsupervised approaches work well, they typically require a lot of hyper-parameter tweaking on a per-dataset basis in order to get satisfactory results [100, 118]. Some methods even require an entirely new network architecture when using a different dataset [118]. Our research in this thesis aims to improve upon this, with the goal of being able to locate robust and reliable keypoints with minimal hyper-parameter tweaking between datasets. We also note that the generalisability of some of these approaches only allows for structures in one category of object to be found at once, opposed to an ideal domain-agnostic scenario.

Bojanić et al. [3] perform an analysis comparing traditional keypoint approaches to deep learning based approaches. They compare the performance of a wide range of keypoint detection algorithms on keypoint verification, image matching and keypoint retrieval. Their findings were that, while deep learning based approaches typically perform better, some combinations of traditional detectors and descriptors outperform deep models.

2.2.3 Applications of Keypoint Detection

There are a wide variety of downstream tasks that use keypoint detection as a foundation. These range from local descriptor algorithms for image recognition [69] to image stitching tasks like panorama creation [97]. A case where the exact locations of keypoints is required, or typically landmarks in this context, is in pose estimation and shape modelling. Cootes et al. [20] have historically looked at the problem of mapping shapes onto images to fit landmarks using Active Shape Models. While these techniques are pre-CNN era, this method is still strong and can be used to reliably add robustness

to a landmark detector. Modern uses of keypoints include pose estimation [6, 23, 111], which is most commonly applied to locating landmarks on people, but hands and faces are also common, as discussed in further detail in Section 2.1. But aside from the literal use of keypoints as landmarks, or as interest points for descriptor algorithms, another major use case is as a generalised spatial representation, which we will discuss further in Section 2.3.1.

2.2.4 Where this field is moving

Recently, there has been research interest in combined deep-learning based keypoint detectors and descriptors, to mimic the keypoint detection and description section found in pre-deep-learning computer vision pipelines. One such approach by Christiansen et al. [18] is UnsuperPoint, building off the previous SuperPoint work [24], which aims to create a self-supervised deep learning based keypoint detector and descriptor. They use a multi-task network architecture, with a common backbone which splits off into separate branches for scores, point locations and descriptors. During optimisation, their approach learns to optimise four loss functions, an unsupervised keypoint consistency loss between transformed pairs of images, a uniform point predictions loss that encourages each image patch to have uniform point placement, a descriptor loss and a de-correlation loss for descriptors to reduce overfitting. The resultant model finds keypoints for any image with real-time speed, making it suitable at the start of a computer vision pipeline, all without the requirement for labelled training data. However, as Bojanić et al. [3] found in their study, the supervised SuperPoint model does not exceed the performance of classical keypoint detector and descriptor approaches when used in verification, matching and retrieval, but does manage to outperform in terms of speed.

As can be seen from other deep learning disciplines, especially Natural Language Processing, huge pre-trained transformer models are becoming the backbone of deep learning pipelines. Computer vision has already started to follow suit, as seen in vision transformers [26]. But in contexts where regions of information should be considered, a keypoint detector and descriptor based transformer may allow for a wide range of downstream tasks while being generalised such that it works with any image dataset.

Another research direction that keypoints have been essential to is point clouds, allowing 3D shapes to be encoded into a set of keypoints. Guo et al. [37] have developed a transformer for converting images into point clouds representing the 3D structure found within the image. This point cloud can then be used for downstream 3D tasks, including part segmentation and 3D model classification. Hui et al. [45] demonstrate one way 3D point clouds can be further refined by converting them into Superpoints which group together similar points with local geometric structures. They demonstrate the power of this technique in point cloud semantic segmentation, achieving state-of-the-art performance.

2.2.5 Summary

This section has discussed how keypoints have traditionally been located using interest point detectors, identifying small patches of interest with the intended purpose of sampling the local context descriptors for downstream tasks. We then discussed how deep learning has changed the landscape of keypoints due to the lack of requirement when using a convolutional filter approach, and the new role of keypoints as either the goal of a pose estimation task or a spatially constrained representation of an image. We finally theorised where the field of keypoint detection is moving in the future, with respects to deep learning, transfer learning and transformer networks, where a pre-trained model can be used or fine-tuned to extract generic spatial information about an image. Our deep learning approach should also take inspiration from previously network architectures including hourglass networks, which fuse together local and global contexts for keypoint estimation. Robustness should be added to our trained keypoint detectors via an image transformation consistency task, and our estimated self-supervised keypoints can be evaluated using a semi-supervised metric in which estimated points are passed through a linear regressor to provide a distance error.

2.3 General Representations

Representations are at the core of data science. How we represent our data has implications on the approaches we take and the algorithms we use. But with deep learning, we can also learn representations, which is important for this thesis within the spatial domain, especially with respect to Chapter 4. We will be looking at the most effective ways of capturing learnt spatial representations, as a method for answering RQ2.

2.3.1 Representation Learning

Representation Learning is at the core of many different machine learning techniques and keypoints can be viewed as a representation using a strong spatial constraint. Bengio et al. [2] in their high-level survey of representation learning, identify three main categories within representation learning; probabilistic models, reconstruction-based algorithms, and geometrically motivated manifold-learning. While the majority of approaches that we consider in this thesis are reconstruction-based, probabilistic models and geometrical manifold-learning have the ability to capture more nuanced representations of data. Narrowing their scope, Jing and Tian [56] survey visual feature learning using deep neural networks, and consider representations from both images and videos, while also discussing common pre-text and downstream tasks, neural network architectures, and datasets. They identify that within images, context and spatial information are important features to distil into a representation, and videos should contain these

alongside temporal features. Kolesnikov et al. [60] break down the factors that impact the effectiveness of self-supervised visual representation learning, with the key finding that both neural network architectures and methods of training must not be considered in isolation, due to inter-dependency between the two elements when learning representations.

Self-supervised learning is a common method for learning representations from an unlabelled dataset. Choi et al. [17] use a simple training task of predicting image rotations to learn representations of images. Despite being a simple task, the semantic features learnt using this approach allow the researchers to get state-of-the-art performance at time of publication for unsupervised feature learning on ImageNet classification, PASCAL classification, detection and segmentation, and CIFAR-10 classification. Doersch and Zisserman [25] apply multi-task learning for self-supervised visual feature extraction, using supervised tasks where data can be collected without the use of manual labelling. They find that deeper networks improve the ability to extract representations in a self-supervised fashion over shallow networks, and that using multiple training tasks improves performance over only using a singular task, while also speeding up training. Their results show a shrinking of the gap in ImageNet classification performance between supervised and self-supervised learning. Self-supervised representation learning also aids in model robustness as demonstrated by Hendrycks et al. [44], who show increased robustness to adversarial examples, label corruption, and common input corruptions. Additionally, classification of out-of-, but near-distribution outliers is improved, allowing for a final model that exceeds supervised approaches.

One important research area within representation learning is in evaluation of representations. Goodfellow et al. [34] claim that a "good representation is one that makes a subsequent learning task easier" and that the "choice of representation will usually depend on the choice of the subsequent learning task". As a general rule, a good representation is one that maximises the amount of information extracted from the input data, if a system exists for measuring the information captured. This will be considered later when we discuss how to represent spatial constraints in order to answer RQ2.

Typically, techniques look to learn a global representation for each image, typically in the form of a vector, but our application requires a spatial constraint to be applied. This operates as a bottleneck and one option for deriving keypoints is to use the soft arg-max function, discussed further in Section 4.5.2, to locate the highest pixel activation in a heatmap [100]. As this spatial representation is simply a list or set of X and Y pairs, it removes non-spatial information while being highly interpretable and generalised for compatibility with any image.

Constrastive learning [72, 101] aims to find representations where the magnitude of difference between representations is proportionate to the difference between data points while maintaining similar representations between similar data points. When applied

to a spatially constrained representation such as keypoints, we would like objects with different structures to be captured with different shapes of keypoints, but similar structured objects with different textural appearances to have similar keypoints. In a keypoint detection context, this is typically considered when we discuss robustness and repeatability.

2.4 Summary

This chapter has explored existing work in the areas concerned by this thesis. We have discussed approaches to solving the problem of pose estimation, ranging from traditional methods using shape models to newer methods using neural networks. We have also seen that 3D pose estimation can be simplified by solving two separate tasks, finding keypoints that relate to joints on an image and inferring depth from those points, to predict a three-dimensional skeletal model.

With previous work in this field in mind, our approach to solving this problem will draw inspiration from some of these methods and techniques. We will split the approach into two composite sub-problems, and find generalised self-supervised approaches for solving both, before building a full pipeline that attempts to solve both sections simultaneously. Framing keypoint detection as a representation learning problem allows us to generalise keypoint detection while endeavouring to maximise information captured by our points. Our keypoint detector network architecture will take inspiration from hourglass networks, which fuse local and global information, and using consistency between transformed images as a self-supervised training signal will be used to locate robust keypoints.

Chapter 3

Inferring Depth from 2D Keypoints using Self-Supervised Learning

This chapter covers the stage in the articulation model estimation pipeline that infers the depth of 3D keypoints from 2D inputs. Necessary for our objectives, monocular 2D to 3D lifting is a powerful tool, as single-view image datasets do not contain any explicit depth. The depth of each keypoint must be inferred from the limited information contained within the 2D data if we are to maintain a self-supervised learning approach.

3.1 Introduction

2D to 3D Pose Lifting aims to infer the three-dimensional structure given an input of two-dimensional keypoints. This objective is complicated further by the dynamic nature of our articulated subjects. The subject can be represented in many different poses, thus shows the need to learn to represent the dynamic pose and the 3D structure concurrently. We must also consider the difficulties of locating the structure of a symmetrical articulated object, can all this be done using only 2D information?

Finding the 3D keypoints that are defined by an articulation model is a step towards answering RQ1, and doing this in a self-supervised environment is the challenging step in this process. While there is a large field of work that considers only 2D keypoints for pose estimation, the benefits of learning a 3D model gives a large amount of information when considering occluded points. Reconsidering RQ3, the term keypoint in this chapter is specifically to resemble the landmarks that designate joints within our articulated model. The order of the keypoints is also important here, with each index corresponding to a specific articulation point, and must be consistent between examples. Considering the

spatial constraints in RQ2, this chapter looks at using a bottlenecked auto-encoder, to compress spatial information, before estimating 3D information from a smaller latent space.

3.2 Related Work

This section will cover related work to the specific areas covered in this chapter. We break the section down into three sections: traditional depth lifting that predate the deep learning revolution, deep learning approaches, and datasets that we can use to both train our model and test our approach.

3.2.1 Traditional Pose Lifting

Traditional approaches to estimating 3D from 2D data required multiple viewpoints, in order to create a well-formed problem which can be solved numerically as shown in Hartley and Zisserman [39]. While this is both elegant and effective, it requires one of two constraints. The first is at least two cameras with a significant distance between them, and for those cameras to have their parameters known, which creates limitations for this approach outside of a laboratory setting. The second constraint is one camera with multiple snapshots of the same object or scene taken while the camera is in motion. Monocular pose lifting is preferable, and early work in this includes Shape from Shading, using lighting cues within the image to estimate depth of a surface [78].

Cootes et al. [20] introduced Active Shape Models as a simple statistical method for capturing the variance of poses, and is easily extended into three-dimensions [7, 9, 46]. The simplicity of aligning the given data and capturing variance about it follows our self-supervised methodology, and the results give robust and reliable pose estimation in both two and three dimensions.

But as we are looking at solving this problem without the use of multiple views, a more appropriate approach would apply deep learning to this problem. We are interested in how subtle information can be found from different scales, give us a system of estimating the depth of keypoints when we have monocular image data to work with.

3.2.2 Deep Learning based Lifting

The problem of finding a 3D pose assumes that we have 2D keypoints of a pose, which requires the other section of the pipeline to determine these from an image. Another option is to simply leverage the accuracy of 2D ground truth data, creating a semi-supervised approach if we are interested in 3D points but only have 2D labels. Martinez

et al. [79] give a good benchmark for a solution to this problem by using a neural network and a dataset of 2D and 3D training examples. They get successful results with a simple neural network architecture by using a few well known tricks such as skip connections [43] and batch norm [47]. Kudo et al. [61] have shown that this task can be solved using unsupervised learning, although their approach does require a strong prior on the angles between joints in order to prevent the inverse pose problem. As this prior is specific to the articulation of humans, it would not generalise beyond human pose lifting. Their successes come from adversarial learning, and the use of a discriminator to determine if the given pose is from the dataset or generated from their pose lifting network. A very similar approach was devised by Drover et al. [27], but using weakly-supervised learning, with an interest in training a supervised 2D keypoint detector alongside the 3D pose lifting. In addition to adversarial learning, Chen et al. [13] use a self-consistency loss function to create an approach that improves upon the accuracy found in other unsupervised and self-supervised pose lifting approaches. Another approach by Wandt et al. [106] achieves monocular self-supervised pose lifting using noisy data taken from an off-the-shelf joint estimator, but requires multiple views at the training stage to prevent an ill-posed problem from occurring. While not a comprehensive study of all pose lifting via deep neural networks, these previous approaches show the applicability of deep learning to the pose lifting problem, especially when we have only 2D keypoints from monocular images.

3.2.3 Datasets

These approaches are reliant on having data to both train a network, but also test its performance. A number of datasets of articulated dynamic objects, primarily humans but also animals. Ionescu et al. [48] created the popular Human3.6m dataset, which contains videos of actors from 4 different angles, along with 2D keypoints for 32 body parts for each video, and 3D keypoints derived from the 2D data. Charles et al. [10] put together a dataset of sign language poses, taken from BBC footage with the aim of human pose estimation. Due to the nature of this data source, there is a constraint of the image subject always facing towards the camera and only having the upper half of their body visible. There is also a dynamic background, being the footage that is being interpreted is sign language, which can add a lot of noise to the images. However, we may want to train or test our network on data that is less constrained than these two datasets which have many fixed parameters. Von Marcard et al. [104] address these concerns and have created a 3D pose in-the-wild dataset, giving 3D pose data outside of a laboratory setting.

While we have focused on human poses, we aim to design an approach that is generalised to any articulated object. The 3D menagerie dataset [120] gives us the option to test the generalisability of our approach on an articulated model that is not just human beings.

While this would be a good dataset for testing our approach, we have not been able to get access to the labelled images with keypoints used to build this dataset.

We must also consider that the datasets that we have identified for use with our research may be biased and using a data-driven technique for solving this problem will learn the biases found in our dataset. These biases can lead to potentially discriminatory models, which is a consideration we must bear in mind.

Bias in the selection of subjects in Human3.6m means that a model will likely overfit to lighter skin colours. This is an issue in the image space, and specifically applies to the keypoint detection stage of the pipeline. The Human3.6m dataset also contains no subjects that have any visible disabilities. This may lead to model failures at inference time if provided with an example of a person with a different skeletal structure.

3.3 Factorised Auto-Encoding

To solve the problem of 2D to 3D Pose Lifting, we will attempt to factorise the 2D keypoint data into 3D keypoints and camera parameters, which project the 3D keypoints back into the 2D keypoints. As our keypoints are derived from an underlying model based on an articulated subject, we should be able to represent the our underlying articulation model using a smaller number of parameters than the raw 3D point data. From this assumption, we have devised a lifting approach that uses a bottlenecked auto-encoder, to compress our joint information into a smaller latent space and learning depth from those latent vectors. The rationale for this is that we hope to leverage the reconstruction loss function as an encoding approach for our keypoints, and if our latent space has encoded the necessary information to reconstruct an input with compressed data, then we hope that depth is also implicitly represented within that latent space as depth should be an integral parts of the underlying articulation model.

3.3.1 Concept

The underlying concept behind this section of work is a factorisation based auto-encoder (FAE). This is a type of auto-encoder network that aims to split the input data into its logical components. These components should be selected such that they can be intuitively recombined to create a reconstruction of the input, which naturally lends itself to a simple self-supervised reconstruction loss to train the network via gradient descent. This is similar to a standard auto-encoder, but rather than a single latent space that can have little restriction on the data format stored, we aim to assign meaning to the latent spaces and not require a learnt decoder to convert back into the input data format.

This idea has many potential applications beyond pose lifting, including factorising shape from texture and disentangling image and temporal factors in video. However, we will be using this concept to solve the problem of finding 3D keypoints and camera positions that correspond to 2D data.

3.3.2 Applying Factorised Auto-Encoding to 3D Pose Lifting

Now we have introduced the idea of Factorised Auto-Encoding, we will demonstrate how it can be used to solve the 3D Pose Lifting problem. In the context of factorisation, we can view real-life images as a two-dimensional representation of our three-dimensional world. One three dimensional object could be represented by many two-dimensional images, where we can adjust the position and rotation, or other parameters such as aperture and lens shape, of the camera used to produce the final image. In order to learn the structure, we are required to disentangle the 3D world from the 2D examples. If we can isolate the parameters of the camera from the 3D world, then it is intuitive to combine the two to get a reconstruction of the 2D ground truth, giving a simple self-supervised loss function, as outlined in Figure 3.1. However, difficulties in this approach quickly become apparent: there are many different mappings from a 2D pose to a 3D representation and thus some restriction on this 3D space is required to get an accurate 3D pose.

To enforce this restriction, we will train a neural network to take the 3D estimation and flatten it into a 2D pose with different camera parameters, essentially imagining the object from a different angle. This new 2D view is then passed through the network again to obtain the 3D estimation from this angle. If our network is successful then this 3D estimation will be identical to the one that was flattened in the first place. This provides a reconstruction loss function for our neural network, minimising the Euclidean distance between the two 2D structures. This simple idea was theoretically good, but in practice did not give accurate results, as the network could get similar 2D reconstructions but they would not be realistic as a 3D pose.

An attempted solution to the problem of poor 3D poses was a restriction that applies small changes to the camera parameters before flattening back to 2D, which should give a very similar 2D reconstruction if the 3D space was accurate.

3.3.3 Implementation

Our deep learning based keypoint lifting implementation uses only fully connected layers with non-linear activation functions, owing to the data format of numerical keypoints.

Our implementation also opted for learning a mean 3D pose along with a pose residual that converts the mean pose to a specific example. Our approach splits the spatial

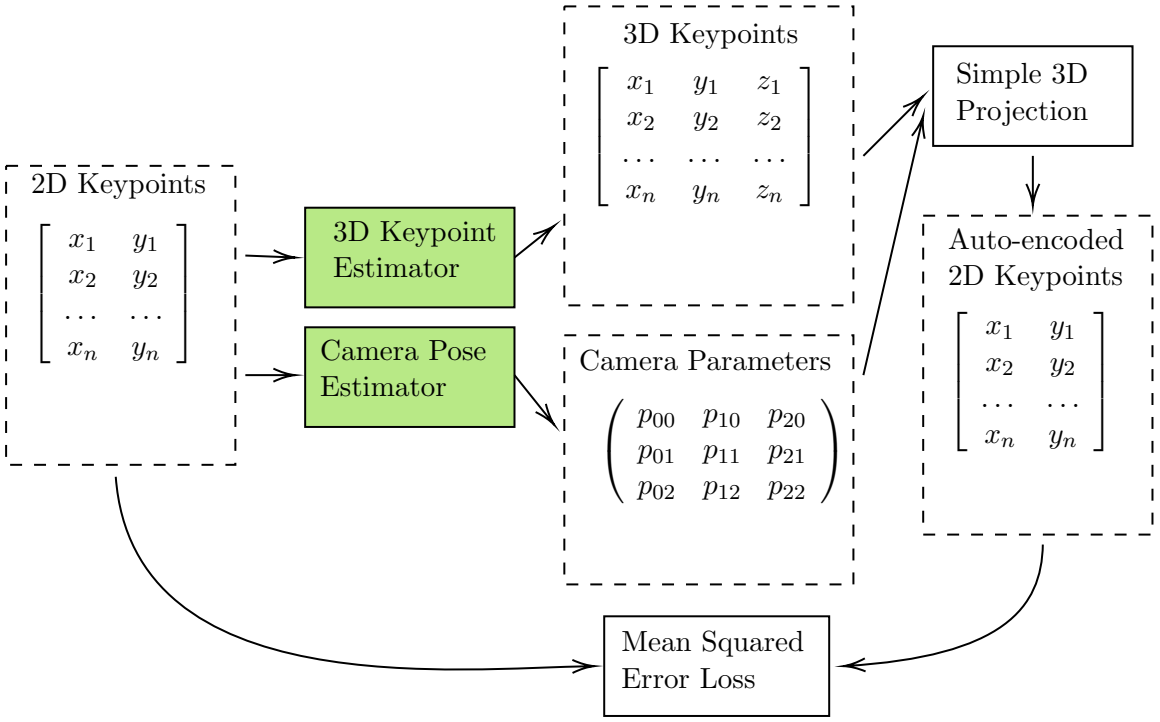


Figure 3.1: A block diagram showing how the components of this approach are linked together to create a trainable auto-encoder. 2D keypoints sampled from the source data are fed into two different networks, one to estimate the 3D points and one to estimate the camera parameters used to capture the 3D structure in that specific view. These are then combined with a projection to create a reconstruction of the 2D data, which is used with the original 2D data in a mean squared error loss function in order to provide gradients to optimise the networks. Blocks in green are trainable neural networks, and blocks in dashed boxes are data at each stage of processing.

representation like this as to more clearly distill the pose information in the latent space vector, theoretically simplifying the depth inference. Part of the rationale behind this implementation choice is that being a derivative from a base pose should prevent major errors in depth inference.

2D keypoints are fed into the camera parameter network, the mean pose network and the pose residual encoder network. The pose residual encoder then produces a latent vector of size l, which we in turn pass into our pose residual decoder, and sum with the output of the mean pose network to get the estimated 3D pose. Projecting this with the estimated camera parameters will then produce a reconstruction of the 2D input, which we then use with a mean squared error function to produce a final reconstruction loss for backpropagation.

We also require other losses to break the ill-formed problem that we face when factorising the camera from the 3D keypoints. The summed magnitude of the pose residuals are used as an additional loss, forcing the mean pose to learn a good mean pose structure which requires the minimum amount of other information to achieve the final poses. As

previously discussed we also have added another loss function that adds marginal noise to the camera parameters in the hope that it will prevent the network from learning a very large range of depth values, stretching the pose out in the z dimension.

Full details of the network parameters can be found in Appendix A.

3.3.4 Initial Experiments

The results from this experiment show promise, but are not yet satisfactory. While we were able to successfully reconstruct the 2D inputs from the latent space, showing spatial information has been preserved through a bottleneck, the 3D structures produced were implausible. This has occurred as there is no restriction on the 3D space to represent the correct pose, only to encode the pose in an arbitrary set of points, which can then be used to reconstruct the original 2D pose, as shown in Table 3.1. Moving forward from this point, we can assume that greater restrictions will need to placed on the 3D structures to see improved results.

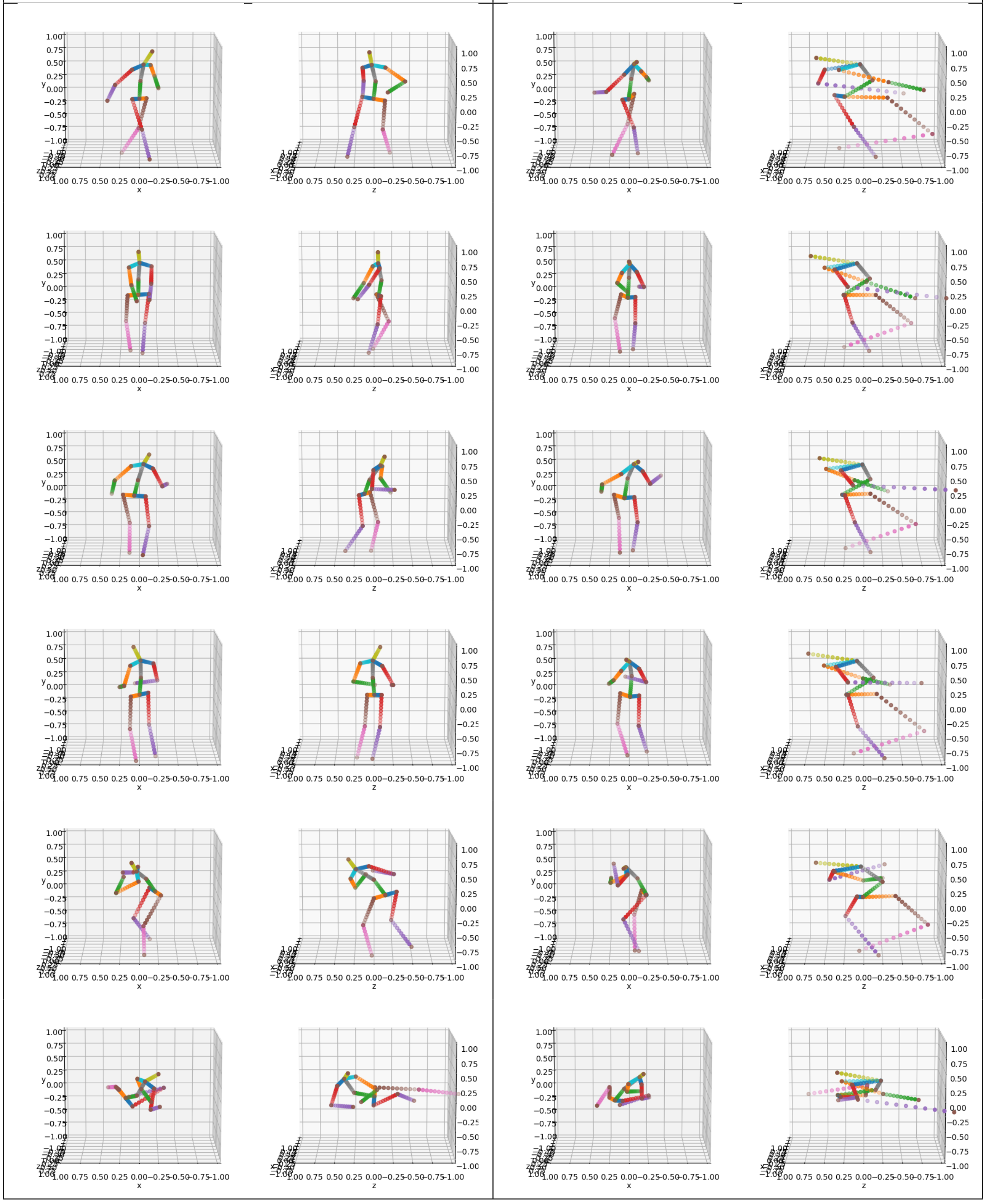
3.3.5 Required Modifications

The idea behind this approach has potential, but the problem we are trying to solve is difficult. While we are able to produce accurate 2D reconstructions using a simple Factorised Auto-Encoder, the 3D estimations are far from realistic. Inspired by Kudo et al. [61], we have attempted to use an adversarial approach to apply a restriction to the generated 3D poses.

Discriminator networks have proven to be successful for many applications in self-supervised deep learning, particularly in generative models, but by changing the generative network to a depth estimator, we can leverage the power of a discriminator network for predicting if a pose is from the original dataset or created using predicted z co-ordinates.

Table 3.1: Results showing 3D estimations taken from the Factorised Auto-encoder network trained on 2D data from the Human3.6m dataset [48]. Results show that 2D reconstructions are successful, but depth estimations do not align with the 3D data found in the input.

3D Ground Truth		3D Estimation	
Front View	Side View	Front View	Side View



3.4 Applying Adversarial Learning

Popularised by Generative Adversarial Networks [35], adversarial learning typically consists of adding a discriminator network that is trained to identify real and fake training examples. We train it in tandem with the generator network, with a separate optimiser for both the depth estimation and discriminator networks. Our training objective for the discriminator network is to correctly identify the real and fake examples, and our objective for the generator is to produce fake examples that the discriminator classifies as being real.

An adversarial loss applies to this scenario as we can use the discriminator network to learn valid 2D poses. Our real examples are taken as our inputs to the network, 2D poses sampled as random rotations from the 3D dataset. We then generate a 3D structure using the generator network, and sample it from new random angles to generate our fake 2D poses. We are choosing to do this to force the generator network learn to generate a 3D structure that looks realistic when viewed from any angle.

3.4.1 Motivation

Solving this problem using a discriminator has already been shown to work by Kudo et al. [61], but our approach differs in a few ways. We do not require strict constraints to the 3D human pose due to the bottleneck in the pose encoder which forces a latent space with spatial constraints between the points in three dimensions. We also do not use such heavy normalisation on the dataset in order to train effectively.

We take inspiration from the factorised auto-encoder in this design by factorising the 3D shape into a mean shape, pose residuals, and pose rotation, which is simplified from the camera parameters as used initially. We find that this gives the freedom to learn the shape, pose and alignment independently and combines the three in order to create a 2D reconstruction.

Our discriminator network takes our 2D reconstructions and the ground truth data, and is given the task of identifying the real and fake examples. Our pipeline has two separate optimisers, one for the discriminator network and the other for the keypoint lifting network. The discriminator network is trained to minimise the errors of the adversarial predictions for both real and fake examples. The keypoint lifting network has two loss terms, the first is a simple reconstruction loss using our flattened 2D reconstructions. The second loss term is to maximise the error rate of the discriminator when a fake example is the input. These losses are summed together with a multiplier applied to the reconstruction loss to balance the terms. Figure 3.2 shows an outline of our approach.

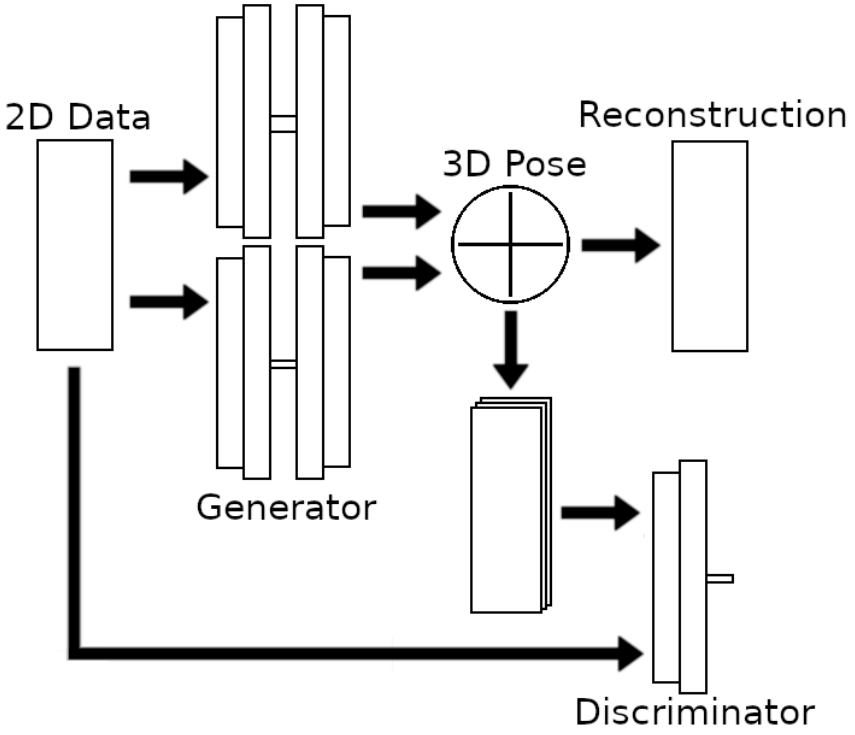


Figure 3.2: Outline of the network architectures for the components of this method. A dataset of 2D data is split into two networks, one to learn a mean shape and one to learn a pose residual, which are summed to create an estimated 3D pose. This 3D pose is flattened to create a reconstruction loss, and flattened from a random angle to be used with the discriminator network to create the adversarial loss function.

3.4.2 Implementation

Our PyTorch[87] based approach is comprised of several sub-networks built from fully connected layers. The first of these sub-networks aims to find the mean 3D pose, pushing the 2D point data through a bottleneck of size 1, leaving a small amount of freedom for scale. The second network finds the pose residuals, and also has a bottleneck with variable size which is a hyper-parameter to be optimised during experimentation, as the size of this bottleneck would vary with different underlying articulation models. Finally, we use a network to predict the rotation of the 2D input, and outputs a numerical value representing the number of radians to rotate the 3D structure before flattening to 2D.

As this is an adversarial approach, we also need a discriminator network. This is simply a network that takes a set of 2D keypoints and gives an output of size 2 that is then passed through the softmax operation, giving the networks prediction confidence that an input is real or fake respectively.

The pose bottleneck in the generator is a key part of this design and choosing the value for its size is a trade-off. If the bottleneck is too narrow then the network is unable to locate good reconstructions of the vast number of valid poses, but if too wide then there

is too much flexibility such that compressed skeletal constraints will not be learnt. The optimal value varies with the complexity of the underlying articulation mode that we are learning to represent, but from experimentation, a good starting value is half the number of keypoints.

Balancing the two loss terms is another difficult problem. If we focus too much on getting a good 2D reconstruction then we find the 3D estimation becomes less accurate. If we focus too much on the 3D estimation then we find that the poses are realistic, but differ from the ground truth. This is due to our information bottleneck having to encode our skeletal information into a compressed latent space. We are assuming that our model is learning to encode the pose as a set of points which are restricted in distance to other connected points, inferring the skeleton as a set of joints with rigid bones between them. This will naturally cause some issues with accuracy due to the large variance in our dataset of poses. Balancing multiple loss terms is investigated further in Chapter 4.

3.4.2.1 Network Architectures

This network architecture is split into two sections, the generator and the discriminator. The generator network has a few different components in order to recreate a 3D pose from a 2D input. The first part learns a mean shape for the subjects. This network takes n inputs and flattens the 2D input into a vector of size $2n$, it then passes through 4 fully connected layers, each with ReLU activation apart from the final layer which has a TanH activation so that the mean shape lies in the range $[-1, 1]$. These layers have sizes $25n$, 1, $25n$, $3n$ respectively and the final output is shaped into a matrix of size $(3, n)$ to represent points in 3D space.

The network then also learns the pose residuals which are summed with the mean shape above to generate the final pose. This network also has an input of size $2n$ and passes through 6 fully connected layers. Layers 4 and 6 have TanH activation so that our latent space and outputs are both in range $[-1, 1]$, and the other layers have ReLU activations. The layers for this network have the following sizes; $25n$, $50n$, $25n$, λ_p, $25n$, $3n$. λ_p represents the chosen latent space size, which for experiments on the human3.6m dataset, is set to 7. Further hyperparameters relating to the training of this network are available in Appendix A.

The discriminator network consists of 4 fully connected layers, all with ReLU activation apart from the final layer, and the output from the network has softmax applied to it. These layers have sizes $50n$, $25n$, $5n$, 2, where the output is the prediction of real or fake.

3.4.3 Experiments

Using this implementation, we run our experiments using the Human3.6M dataset [49], sampling only 16 of the 32 keypoints to get a simple outline of human pose. We do not use the 2D data included in this dataset, instead we take the 3D data and sample from random angles to create a larger 2D dataset with no bias towards seeing poses from only the four viewpoints as found in the dataset. Each epoch we shuffle the dataset and rotate each example by a new randomly selected angle. The hyper-parameters used for this experiment are shown in Table A.1, found in Appendix A.

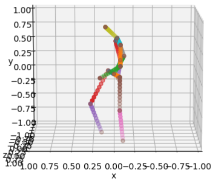
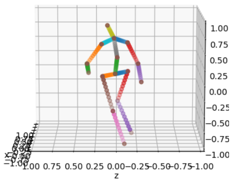
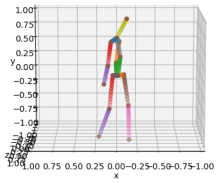
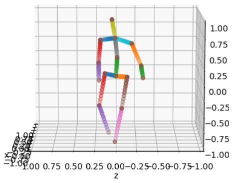
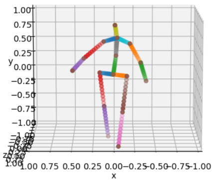
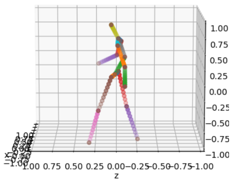
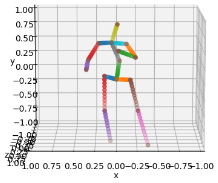
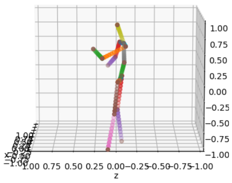
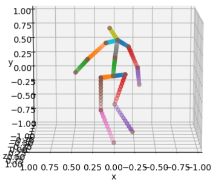
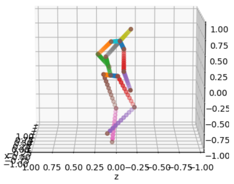
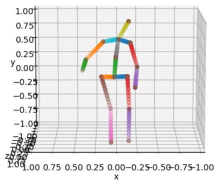
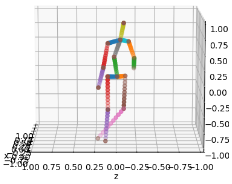
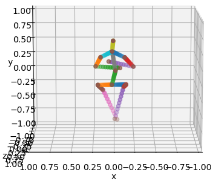
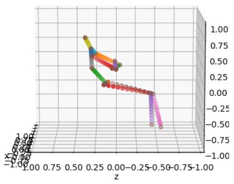
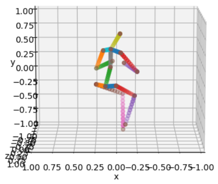
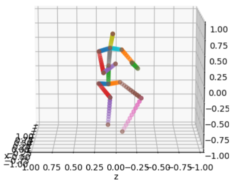
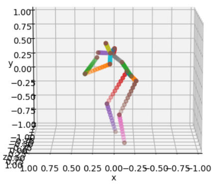
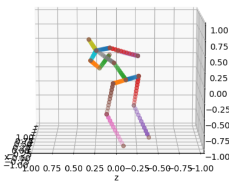
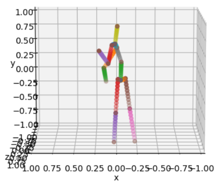
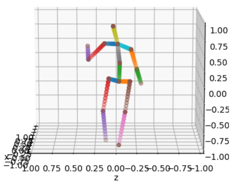
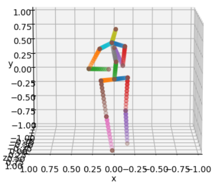
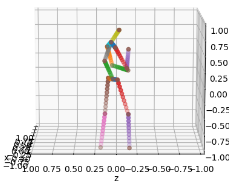
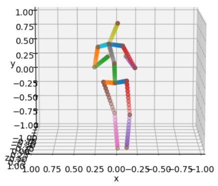
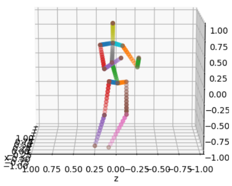
3.4.3.1 Qualitative Results

Our results are shown in Table 3.2, showing the ground truth 3D data alongside our 3D predictions.

3.4.3.2 Quantitative Results

Following previous approaches to this problem [61, 79], we evaluate our approach quantitatively by measuring distance from our predicted 3D keypoints to the ground truth. Our self-supervised approach obtains a higher error than the weakly supervised and supervised approaches, and a slightly higher error than similar self-supervised approaches. The reason for a drop in accuracy most likely stems from the bottlenecking and factorisation losing some accuracy in the X and Y dimensions, along with the expected error in the Z dimension.

Table 3.2: Results showing 3D estimations from sets of 2D keypoints. Both 2D reconstructions and 3D depth estimation resemble the inputs used when using the adversarial pose lifting approach.

3D Ground Truth		3D Estimation	
Front View	Side View	Front View	Side View
			
			
			
			
			
			

Approach	Mean accuracy (mm)
Martinez et al. [79] (supervised)	45.5
Drover et al. [27] (weakly supervised)	64.6
Kudo et al. [61] (self-supervised)	130.9
Chen et al. [12] (self-supervised)	51.0
Ours (self-supervised)	155.5

Table 3.3: Mean distance between predicted and ground truth poses in the human3.6m dataset.

3.5 Discussion

We have shown the application of factorised auto-encoding to the area of 2D to 3D keypoint lifting, where data is split into composite components before being recombined to create a reconstruction loss. Upon finding that we have an ill-posed problem, the addition of a discriminator network to differentiate between real and predicted poses allows for a strong enough constraint to be applied to our problem to achieve good results.

However, adversarial learning as an approach to solving the ill-posed problem is not foolproof. Training time for a generator and discriminator pair is generally higher, and can be unreliable. In addition, there are a greater number of hyper-parameters required to be tailored in order to successfully train our network.

3.5.1 The Inverse Pose Problem

Another problem that is commonly faced when using adversarial learning to lift symmetrical dynamic objects such as human poses is the Inverse Pose Problem. Inferring 3D structure from 2D examples with no supervision is a challenge due to multiple viable 3D structures that could represent one 2D pose. Due to the natural symmetry of human poses, but also many other articulated objects found in nature, self-supervised pose lifting can lead to valid looking poses, but the order of points show an inverted skeleton. This problem was encountered by Kudo et al. [61] and solved by enforcing certain restraints in angles between pairs of joints. This approach to solving the issue goes against the end goal of general pose estimation, so we have used a different solution to the problem. During our research, we found that this problem sometimes arose when training our adversarial pose lifting network, but was an unpredictable occurrence. Our final results do not show the inverted pose problem, but no explicit steps were taken to remove this problem from our final model. The rationale behind this not being present is, as discussed in our implementation section, the additional loss term to minimise the absolute values in the pose residuals. If we assume that our average shape is truly the average pose in the 3D space, then the true 3D pose will be closer to the average than the inverted version.

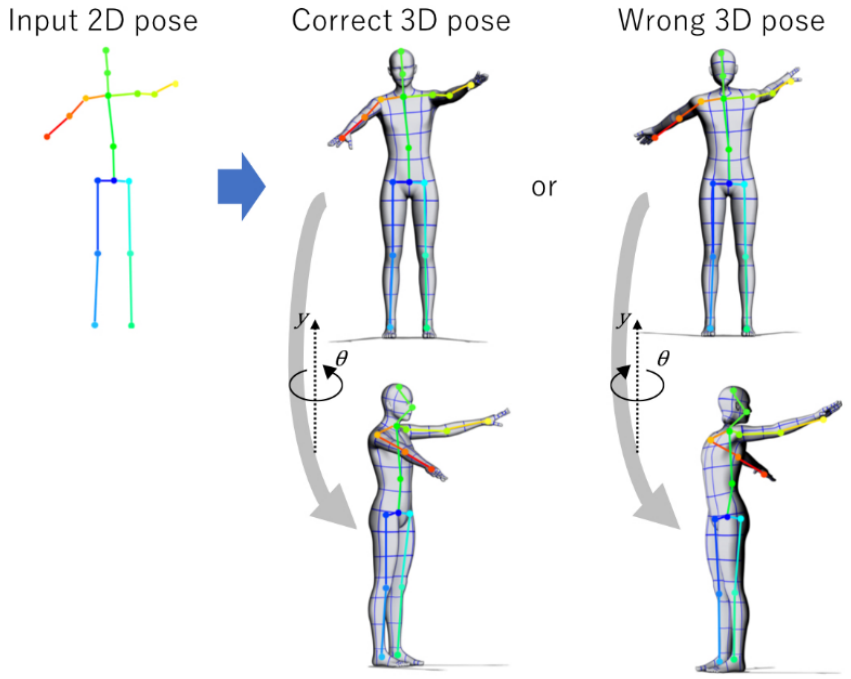


Figure 3.3: Figure taken from Kudo et al. [61], a visual demonstration of the inverse pose problem.

3.5.2 More Applications of Factorised Auto-Encoding

The concept of a factorised auto-encoder has many potential applications; further researching may yield interesting research possibilities.

- Factorising grammar from vocabulary to learn sentence structures
- Factorising shape from texture in both 2D and 3D scenarios
- Factorising voice from the words in speech recording
- Factorising facial appearance from visual emotions

As seen by potential applications, this conceptual idea is promising for a wide range of disciplines and applications. It is also clear how these can be used for generative models; creating previously unseen novel examples by combining different components. These novel examples could be applied to the creation of artificial datasets, for data augmentation purposes.

Constraints on one or more factorised elements may be required in most cases, where there may be multiple sets of elements that give the correct output when recombined,

but individually are not correct solutions, as is the case in our initial 3D pose estimator approach.

3.6 Summary

In this chapter we have discussed the concept of a factorised auto-encoder and attempted to apply it to solving our problem of finding 3D keypoints from a 2D dataset. Upon finding that it does not give satisfactory results, we have modified the design to use an adversarial approach to get promising results. However, as discussed earlier in this section, this approach is not ideal and could be improved. We will revisit this problem in Chapter 5, and look to create a solution with the addition of a generalised prior to add a suitable restriction on the 3D points that are predicted, with the aim of improving these results.

This is only one part of the articulation model estimation pipeline, as we do not always have 2D keypoints available to us. In the next section, we will look at how we can capture 2D keypoints from images in a self-supervised fashion, with the aim of piecing 2D keypoint estimation with this section to create a full image to 3D keypoint pipeline.

Chapter 4

Self-Supervised Learning of Generalised Spatial Representations

Self-supervised keypoint detection is an important part of representation learning, with the aim to learn a spatially constrained representation of an object in an image. This problem is an abstraction of the initial stage of our articulation model pipeline, where we look for landmarks placed on points of articulation in images, but in learning an abstraction, we can later enforce properties using prior knowledge to meet our aims. Previous approaches have come close to being able to distill solid keypoints with no supervision, but typically lack generalisability to different datasets and tasks. After extensive research of different approaches to solve this problem, we propose a new technique for self-supervised keypoint detection that leverages the generalisability benefits of multi-task learning, selecting a small set of downstream tasks to aid in the training process. We find that our keypoints can capture a wide variety of structures and are generalised to a much greater extent, such that we can test on a different dataset to the one trained on with no noticeable drop in performance. Finally, we present a detailed discussion on the state of the field of keypoint detection, analysing some common pitfalls and suggest some areas of interest for future research.

4.1 Introduction

Keypoints have long been an essential component in the computer vision field, traditionally being used as an anchor point for local descriptor algorithms, but in recent years, have become less critical due to the rise in popularity of learned convolutional approaches. But this is not to say that keypoints no longer have their uses. They are easily interpreted so are ideal for capturing the shape of a given object in an image, and

as discussed when introducing RQ2, can be used as a spatial bottleneck. Keypoints are trivial to work with and from them we can derive a wide variety of information about the spatial properties of the object in an image.

Pose estimation and face tracking are areas where keypoints are still ubiquitous to this day, and keypoints are contextually interpreted as landmarks, as they locate features on the object. As discussed in RQ3, the usage of the term keypoints varies greatly on the context. While sometimes used synonymously, keypoints and landmarks are subtly different: landmarks represent a desired feature, for example an eye on a face or a hand on a body. In the context of this chapter, we define keypoints as a way of capturing the shape of an object but do not necessarily line up with any specific features. This implicitly means that keypoints taken from an image cannot be incorrect, but simply sub-optimal in terms of spatial information captured. However, we may want to enforce that they should not lie outside of the bounds of the object that the shape is aiming to capture. Because of this, evaluation of self-supervised generalised keypoints remains difficult and we discuss this further in Section 4.6.

Locating landmarks when provided with labelled data poses a relatively simple regression problem, whereas a self-supervised method for finding landmarks is a greater challenge. Previous self-supervised approaches [98, 99, 100, 118] use a semi-supervised approach, learning keypoints with no supervision before training a simple linear regressor with a small amount of ground truth data in order to translate rough keypoints into landmarks. The benefits of a semi-supervised approach combines some of the benefits of both the supervised and self-supervised approaches, where less labelled data is required but a representation is still learnt from a larger unlabelled dataset.

When keypoints are found using a self-supervised deep learning approach, we tend to see that the points capture the required information to solve the downstream task, but we are at risk of overfitting our keypoints to optimise this task over locating generalised keypoints that capture strong structure. To circumvent this, we propose an approach that leverages multi-task learning for its ability to find generalised solutions that satisfy a set of tasks, preventing the likelihood of overfitting one single task. If a set of keypoints is able to be used to solve a range of tasks, then we would expect these points to capture the structure of an object in a generalised way.

This chapter proposes a novel approach to self-supervised keypoint detection using multi-task training after finding failures with simple and naïve implementations. We show how training with more tasks can lead to better generalisation than single task training. We also studied different approaches to multi-task learning and settled on an approach that we have identified that consistently provides good results. Finally, we hypothesise the qualities of tasks required in a multi-task learning scenario for learning to detect desirable keypoints.

4.2 Related Work

Chapter 2 gives a further outline of related work, but this section will cover a focused analysis of related work specific to this chapter.

4.2.1 Multi-Task Learning

The problem of self-supervised keypoint detection is non-trivial, so we aim to leverage the power of multi-task learning to aid in the discovery of points that not only represent our structures but do so in a generalised way. Zhang and Yang [117, 116] survey the area of multi-task learning and identify a wide range of domains that multi-task learning has been applied to. Hassani and Haley [41] show how using a combination of a reconstruction task, a clustering task and a prediction task can learn features on point clouds in an unsupervised setting, outperforming prior unsupervised approaches. In the area of supervised multi-task learning, Zhang et al. [119] have shown the effectiveness of leveraging classification style tasks alongside keypoint regression tasks to find facial landmarks, observing greater robustness in cases with occlusion and larger pose variance. A study by Standley et al. [96] looks at which tasks should be learned together using multi-task learning, given a limited computation budget with a goal to maximise the efficiency of the learning operation.

However, training in a multi-task environment can be difficult due to the requirement of balancing multiple losses; care must be taken so that one task is not over-optimised while the others are ignored. Sener and Koltun [95] balance multiple tasks using an approach that searches for the Pareto optimal of each objective function. This is then applied to the gradient of each parameter in order to optimise the network towards all tasks at once. Cipolla et al. [19] use an alternative approach that learns an extra parameter for each task which represents the uncertainty for that task. These are learnt alongside the parameters of the network and losses are balanced using this term before the backpropagation step. Another approach to managing losses by Yu et al. [114] looks at conflicting gradients and for any pair found, projects one onto the other such that they no longer conflict. A simpler but still effective technique proposed by Liang and Zhang [68] simply passes each loss through a non-linear monotonically increasing function to map each loss into a different range, with the aim of minimising large discrepancies between the loss values. We explore the merits of these approaches in Section 4.4.2.2 before selecting the best method for our application in Section 4.6.1.

4.3 Initial Ideas and Tests

This section covers initial attempts at training a neural network to detect robust, repeatable and meaningful keypoints with no supervision. Supervised keypoint detection is a well researched field, especially in the context of pose estimation [6, 22, 85]. But when given no ground truth data, the challenge is finding keypoints that are relevant for the context of the application. When the context is not known, our approach is reshaped into one that aims to locate strong spatial representations that can be used for downstream tasks, or even regressed to ground truth points in a semi-supervised fashion. Our initial approaches did not manage to get satisfactory results in finding robust and repeatable points, but we manage to find some success when re-implementing a previous approach.

4.3.1 Keypoint Detection by Image Triangulation

We propose an approach that aims to locate consistent keypoints between two sequential video frames. We aim to train a keypoint regressor using a loss function based around keypoints consistency via image triangulation to find consistent pixel-wise triangles. If we take two sequential video frames, we would expect there to be a small amount of difference between the appearance of those frames, meaning that a keypoint detector should predict similar points, but following the natural change found in that frame change. We aim to capture this frame change by splitting the images into meshes of triangles, assuming keypoints are aligned between each example, and applying the necessary affine transform to align those two triangles. The meshes are determined by the Delaunay triangulation algorithm using a set of points regressed from an image with our keypoint detector network, resulting in a set of triangular segments of the image. We are able to create a simple loss function that calculates the per pixel mean squared error after aligning corresponding triangles using a simple affine transform.

If our keypoints were found to be consistent between both images, then we would expect a small loss. While the objects in our video sequences will be dynamic, this movement can be approximated by using many small triangles, giving a transform between the two images that should capture natural transformations.

However, this approach is fundamentally flawed as can be seen from the results in Table 4.1. With only the affine consistency loss, we observe a keypoint collapse in our network, where every point is placed on the same area of the image, and even with a keypoint separation constraint, our keypoints do not capture a good spatial representation. We also note that this approach is not invariant to lighting between video frames, which can be a big problem outside of laboratory conditions. Another problem with this approach is occlusion, or large jumps between frames, which might occur if either the camera or subject moves quickly. Finally, image blur could cause features in adjacent

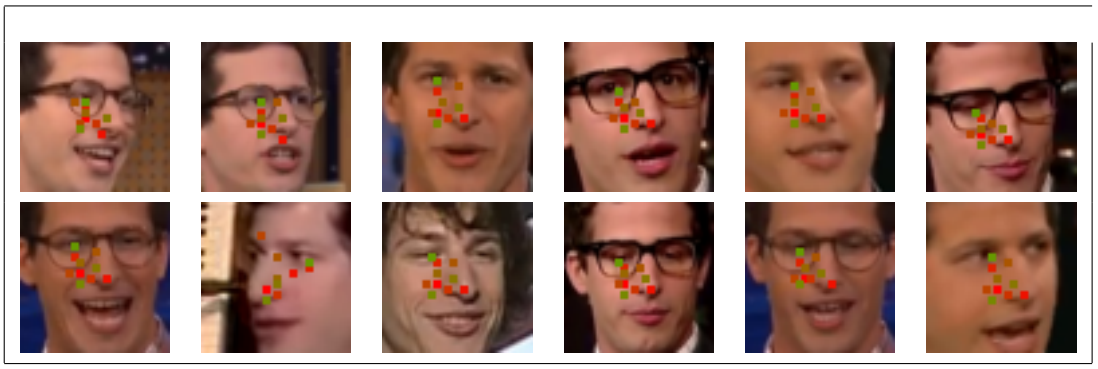


Table 4.1: Results showing a failure to find robust, repeatable and meaningful keypoints using the triangulation method. Keypoints are clumped in the centre of the images and represent a static shape.

frames to appear visually different, which would make this loss function find a high loss in cases that keypoints were placed correctly.

From these failures, we see that we require a more sophisticated approach to capturing spatial representations. In the next section, we turn to other literature to re-implement a successful approach, so that we can learn what elements are required to solve this problem.

4.3.2 Re-implementing an Existing Approach

We look to re-implement the work introduced by Thewlis et al. [100], which was briefly discussed in Chapter 2. It explores detection of robust and repeatable keypoints on images with no supervision. The key idea is that if we take an image α_1, and apply a random non-linear transformation to it to make a second image α_2, we should be able to find keypoints on α_1 that correspond to the same location of the image in α_2. As we know the transformation applied, we can check if the predictions line up by applying the same transformation to the estimated keypoints and measuring the L2 loss between them. This naturally gives us an intuitive loss function for our neural network, as this loss function will reach 0 as keypoints perfectly align.

While we can find pairs of points that correspond to each other using this theory, there is no guarantee that these points represent landmarks that contain an accurate spatial representation. To counter this, one approach used in the paper is to locate a large number of keypoints and then use a semi-supervised linear regression step to predict ground truth points using the self-supervised points as an input. While this does defy the concept of self-supervised learning, this approach gives good results and can do so with a very small number of labelled images from the dataset.

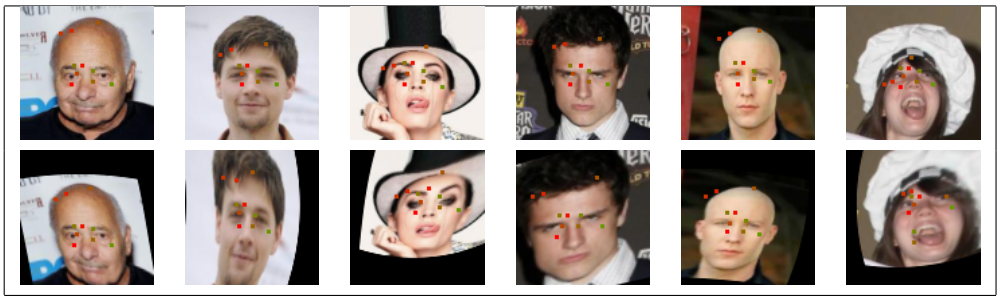


Table 4.2: Results showing consistency between pairs of images with the TPS transform applied.

4.3.2.1 Thin Plate Splines

Thewlis et al. [100] make use of Thin Plate Splines [28] as the non-linear transformation to both keypoints and image data. This algorithm uses a set of fixed control points with weights in which all pixels in the image are modified by based on their distance to each control point. Any set of points can be used, but our implementation uses 25 points arranged in an equally spaced 5 x 5 grid. The weights are sampled randomly from a Gaussian distribution with a mean of 0 and a standard deviation which can be modified to change the intensity of the transform. Each pixel or keypoint is then warped corresponding to its distance from each control point and their corresponding weight. The effect is a transform that mimics a thin plate of metal being bent for a natural looking non-linear transformation, with no sudden changes in pixel translation.

4.3.2.2 Results

Table 4.2 shows some results from this implementation. As can be seen, the keypoints found are fairly consistent between pairs of images as well as between examples of faces. There are a few quirks though, rather than finding object structure, it seems to be focusing on edges around the top of the head. This means that these located points are at the top of the head for bald people and are on the hairline for those with hair. The last example is a failure case, where the keypoints are not consistent between the pair of images, which is most likely due to the different viewpoint, as well as occlusion due to the hat.

4.3.3 Discussion

In this section we have discussed some initial attempted experiments to capture self-supervised spatial representations through the use of a keypoint detector. Our triangulation approach faces issues due to keypoint collapse, but we have been successful in re-implementing another paper with a similar aim, managing to get some promising results.

As our focus is on locating spatial representations, we will move forward from this point by formulating our problem in a different way, placing the focus on solving downstream tasks in order to train a keypoint detector to learn a strong generalised representation.

4.4 Multi-task Learning: Motivation and Approach

We are aiming to extract a generalised shape from an image as a list of keypoints $s = ([x_i, y_i] \,|\, i = 1, 2, \ldots, k)$, where s represents one shape, k is number of keypoints and $x_i, y_i \in \mathbb{R}$ represent the x and y coordinate of the i-th keypoint. We should be able to represent any image I as a list of keypoints and as it is a list and not a set, the order of points matter such that the feature in I at each index of s should correlate between examples. Ordering is not an inherent property of keypoints, but we are imposing this restriction for ease of analysis, however the importance of ordering keypoints is discussed in Section 4.7.1.

While self-supervised keypoint detection is not a novel concept, similar approaches use multiple additional constraints to force the network to learn correctly and to avoid a collapse. An example of this is separation constraints that force keypoints apart to ensure that the points capture a good structure from all areas of the image. While we could copy these methods, we leverage multi-task learning to train a network to perform equally, naturally separating the keypoints as this captures the most amount of information to solve the range of tasks. Because of this, we do not require properties to be explicitly defined, as they emerge while the network optimises.

4.4.1 Representation Learning

We can think of keypoint detection as a representation learning problem, with a spatial restriction on the learnt representation in order to locate geometric semantics from an image. Representation learning naturally lends itself to self-supervised applications, due to the ability to use loss functions that do not rely on labelled data. In order for the network to find a good reconstruction, it must learn how to compress an image into a set of keypoints that describe the shape of the object. An auto-encoder is the simplest way of learning a representation in this fashion and by converting the latent space into a list of keypoints, we should be able to capture spatially significant information about an image. However, this method is prone to over-fitting, and an auto-encoder alone will find keypoints that do not resemble the desired shape, instead they appear to be hashing the image data into the keypoint data to minimise our reconstruction loss. To prevent our representation over-fitting, we will take inspiration from multi-task learning to increase generalisability in our representation, due to the criteria of solving multiple tasks with one representation.

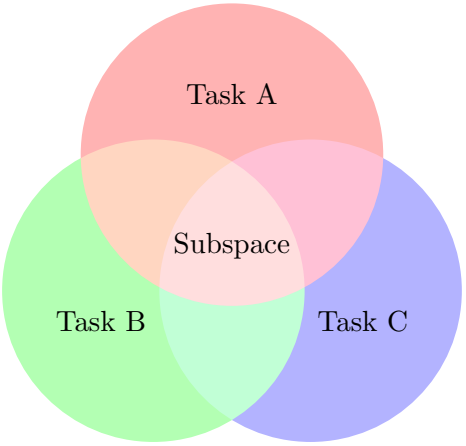


Figure 4.1: Venn Diagram showing how unrelated tasks can find an intersection that contains desirable solutions

4.4.2 Multi-Task Learning

As single-task learning for capturing shape semantics will certainly overfit to the training task at the cost of finding good keypoints, we look to multi-task learning to learn a good representation. If a list of keypoints can be used to solve a variety of tasks, then we can assume that those keypoints have captured a generalised structure found in the image and has not encoded the image into a specific representation to optimise a single downstream task. If we imagine a space of keypoints that suitably solve a given self-supervised task, there will be many valid solutions. The intersection of the space of solutions between a pair of tasks will narrow down to a smaller subset that contains keypoints that satisfy both tasks. As we add more tasks, assuming they have enough variety in order to minimise overlap between point spaces, then our optimisation has a limited set of shapes that can satisfy all of the tasks. This is demonstrated in Figure 4.1.

The idea of a intersection of shapes that satisfy each task leads us towards the intuition of selecting tasks to train with. First, we need a good number of tasks and secondly, those tasks need to be varied such that the intersection covers a small area. The result is an optimisation that should have a small set global minima that satisfies all the chosen tasks. If the intersection is too small or does not exist then we may need to consider a trade off, where we use a solution that is suboptimal for some or all tasks but achieves the best results when losses from each task is given a weighted average. However we may find that optimising in a multi-task environment increases how difficult it is to find the optimal solution.

4.4.2.1 Choice of Tasks

Choosing the correct downstream tasks for the training signal for our keypoint regressor network is crucial to the success of this method as discussed later in Section 4.7. Below

is a brief description of which tasks we are using and why.

Reconstruction with Global and Local Descriptors. Representation learning naturally lends itself to auto-encoder style architectures where the task given to the network is to reconstruct the input image from a learnt representation, which in our case is a set of points. Keypoints alone can encode sufficient information to successfully reconstruct the input image when the dataset of images contains little textural information, such as MNIST, but when images have texture, extra information is required. For reconstructing images with texture, we use local descriptors taken from a circular crop around each keypoint and compressing that information through a simple learnt CNN into a small descriptor vector. We do the same process on the whole image to obtain a global descriptor and then reconstruct the image from the stacked local descriptors and the global descriptors.

The aim of using a reconstruction task is to capture structure in the image and to bias the keypoints towards areas with more texture. An area with more texture is more likely to resemble a robust and repeatable point of interest.

Choosing the hyper-parameters for the circular crop sizes and the global and local descriptor vectors is essential for success using this method. If we select too large descriptors or patches, then the information bottleneck is not tight enough to force the network into learning a solid representation.

Referential Game using Distractor Images. Classification tasks typically require a labelled dataset, but by making a change to the formulation of the task, we can use this style of task without labelled data. Taking inspiration from Havrylov and Titov [42], we train a downstream network to decide which image was used to create the list of keypoints, and selecting an image from a stack of images alongside distractor images, taken from the same dataset. If our keypoints convey enough information about the input image, then we should be able to accurately predict which image corresponds to the found keypoints. This task varies in difficulty based on the dataset that we are using, as some datasets have high variance in shapes in the images while others are aligned, so a hyper-parameter of how many distractor images can be modified on a dataset basis.

Middle Frame Predictor. While other approaches to self-supervised keypoint detection use sequential information or artificial warps [99, 100, 118] to check keypoints consistency, we find that such a strong constraint is not required to distil robustness into our keypoint detector. While these techniques give good results, better generalisability can be obtained by learning tasks that are unrelated to the desirable keypoint properties and finding emergent properties instead. We take inspiration from Misra et al. [83], who use the ordering of shuffled video frames as an unsupervised training signal. Our

implementation of this task takes the input image and applies a random Thin Plate Spline warp [28], as previously discussed in Section 4.3.2.1, once to create the middle image, then applies the same warp to the warped image to create a third. Keypoints are extracted from each image and stacked before being shuffled and then we train a downstream network to predict which is the middle of the three images.

4.4.2.2 Loss Balancing

As we are using tasks that have varying loss functions and different loss landscapes, in order to combine tasks such that no one task dominates, we must carefully balance our losses. The obvious way of doing this is manually selecting loss alphas to bring each loss into a similar scale, but this requires hyper-parameter searching and alphas will not always be optimal when we modify other hyper-parameters or when we use different data.

Pareto Optimal Gradient Tweaking. One option for automatically balancing losses is a method described by Sener and Koltun [95] that aims to find a Pareto optimal solution. When we attempted this implementation, we found that even though this method optimises towards the Pareto front, this does not help to find a set of network parameters that give desirable keypoints. This method has the additional downside of being difficult to implement and cumbersome to train due to the requirement for two back-propagation steps.

Learnable Task Uncertainty. To prevent lengthy hyper-parameter tweaking, Cipolla et al. [19] describe an approach that learns an extra parameter for each task that represents its uncertainty. By using the same optimiser for the network and these parameters, this approach strives to automatically balance tasks with varying loss functions and with different loss variances and is shown to model the uncertainty with accuracy. In practice we have found that while we can balance tasks successfully, we observe a high variance in task performance over multiple runs, as shown in detail in Section 4.6.1.

Balanced Multi-Task Learning. Another approach as described by Liang and Zhang [68] uses a fixed monotonically increasing function that is applied to the losses from every task in order to bring the loss values and training gradients into the same region. They use $e^{l/50}$ as their mapping function and we have found that this gives good results when used on simple datasets where our descriptor reconstructor task is not used. While intuitively $\log(1 + l)$ should similarly map all of our losses into the same region, our experiments would suggest that because the gradients of a log function accelerate towards zero as the loss tends to zero. This results in easier tasks being over-optimised

at the cost of the more difficult tasks, and the optimisation getting stuck in a local optima with poor keypoints. In a similar fashion to the Adaboost algorithm [31], the exponential function puts a greater emphasis on tasks with high losses, meaning the gradient descent for tasks that are struggling will have a greater signal to the network weights than that of tasks that are being solved easily. This technique helps us a lot as we are using a varied set of tasks where some will be easier to solve than others. In Section 4.6.1, we go into more detail about these techniques to determine which performs best in this scenario.

4.4.3 Heatmap Concentration Constraint

We use one additional constraint alongside our tasks that encourages the network to learn keypoints on images through patches of texture or though shapes and not learning vague shapes using image borders or random noise. We use the l_2 loss between output heatmaps from the keypoint detector network and Gaussian blobs around the derived numerical keypoint. If this loss is minimised to zero, then it means that every heatmap is a perfect Gaussian blob around each keypoint. We discuss how this constraint is implemented in Section 4.5.6.

4.4.4 Summary

We have formulated our problem as a representation learning problem with a spatial constraint on our latent space and then have applied the theory of multi-task learning to aid in the discovery of generalised points. We have chosen a selection of tasks to optimise in parallel and methods of combining the losses from these tasks in order to learn keypoints that resemble a robust structure found in the image. Finally, we have introduced an additional constraint that assists our network in locating suitable keypoints.

4.5 Implementation

An outline of how our approach is implemented is as follows, and shown schematically in Figure 4.2):

- Images are fed through a keypoint regressor and outputs k heatmaps.
- Soft arg-max (described further in Section 4.5.2) turns heatmaps into keypoints as x, y pairs.
- Each task then use the output keypoints and calculates a loss.

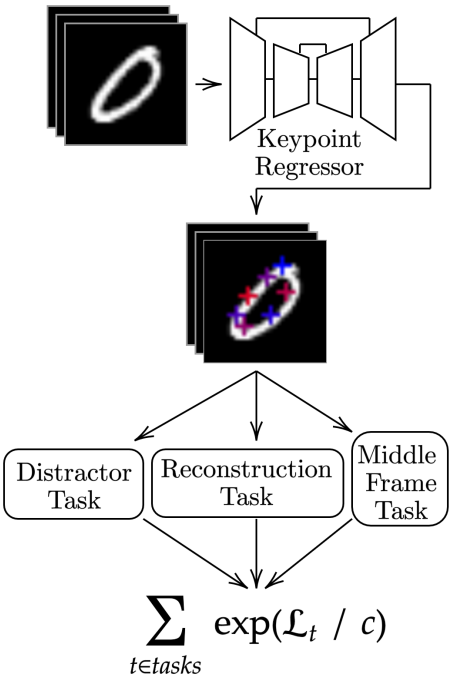


Figure 4.2: Outline of the components in our implementation. Input images are passed through a keypoint regressor using a soft arg-max operator to locate numerical keypoints. These points are passed into each task to derive a list of losses, which are then in turn balanced to create a final loss used for backpropagation.

- The losses from all tasks are combined using a loss balancing method as described in Section 4.4.2.2. Weights in all networks are updated at once.

All code has been written in Python and PyTorch has been used for the neural network implementation.

4.5.1 Keypoint Detection Network

As mentioned in Section 2.2.2, deep learning based keypoint detection and pose estimation techniques commonly use hourglass style networks for extracting keypoints from images, whether or not supervision is available. The benefit of this network architecture is having multiple channels of convolutions that operate on different scales of feature maps. As a result, we gain the ability to fuse information derived from both local and global features found in the images. Local features can then learn to locate small image patches that resemble typical interest points while global information takes into account spatial context of each image patch. Stacking hourglass networks [85] has shown to further improve accuracy for pose estimation, but we have chosen to use a single hourglass to aid in generalisation, as deeper networks can be prone to overfitting. We have found that the keypoint detector hourglass network within this approach is very robust and manages to get good results with no fine-tuning between datasets.

4.5.2 Spatial Soft Arg-Max

To convert the output of our feature detector, m, into a list of keypoints ($[x_i, y_i] \,|\, i = 1, 2, \ldots, k$), we need to use the spatial soft arg-max operator which approximates the arg-max function over two dimensions whilst retaining differentiability, in order to train our network end-to-end. This allows us to find the numerical coordinates of the peak of each heatmap, which we will use for downstream tasks. This is done using Equation 4.1, where β is an optional temperature parameter, which we have set $\beta = 1$ for all experiments in this chapter, and h is the activation heatmap taken from the output of the network.

$$\sum_{ij} \frac{e^{\beta h_{ij}}}{\sum_j e^{\beta h_{ij}}} i \tag{4.1}$$

Equation 4.1 essentially calculates the mean of the activation within the two-dimensional input, converting a heatmap into a 2D keypoint. This process takes the dot product of a co-ordinate grid, our implementation uses a normalised grid with values between -1 and 1, and finds the mean over the x and y dimensions to produce a keypoint.

As this is an approximation, we can see some undesirable qualities. The first being when a heatmap does not have a peak, i.e. all pixel values are equal. In this case we will find that our keypoint defaults to the centre of the image, giving a keypoint of (0, 0), assuming our image range is between -1 and 1. This can lead to the training of our network getting stuck in a local optima where all heatmaps are outputted as being flat and all keypoints being chosen at (0, 0). The second failure case is when we have more than two peaks, and in this case the soft arg-max algorithm will choose a keypoint that lies between the two peaks. This is problematic, especially when using datasets that have images containing symmetry, as we may place a keypoint between two areas that are likely features, resulting in an area with no feature present. An alternative to this algorithm would be to regress our keypoints in a method similar to Liu et al. [70], where the image and x, y coordinates are stacked. We believe that this spatially constrained information bottleneck has value in distilling object structure from a dataset of images, but found soft arg-max to also be an adequate solution.

4.5.3 Heatmap Cleaning

The output of our keypoint regressor network is a set of heatmaps which we then convert into keypoints using the soft arg-max function. As we are aiming to distil a spatially restricted representation, using a numerical input in our downstream tasks does not guarantee a spatial bottleneck, so we must use a heatmap as the input for some of our tasks. However if we were to use the original heatmaps, we would be prone to information

leakage [51], where the heatmap can encode extra information as pixel values. To fix this issue, we reconstruct heatmaps using the keypoint locations as the means for a Gaussian peak placed on a blank heatmap, using a fixed standard deviation, which we could change dependent on dataset. The reason for fixing the standard variation for the Gaussian peak is primarily that it simplifies the training of the keypoint regressor. We acknowledge that the visual information that leads to discovery of a specific keypoint may span either a wider or smaller area, and while this information may be ascertained by the activations in the feature map prior to the soft arg-max algorithm, as seen in Table 4.21 and Table 4.22. If we chose the standard deviation of the Gaussian peak to estimate a best fit to these feature maps, then this is an extra algorithmic step which adds complexity to our training. This complexity could assist in solving downstream tasks as the information captured would not only be the location of the point, but also the size of the feature at that keypoint. While this area could be part of a generalised representation of image-space size of the keypoint, this extra information does not necessarily adhere to the traditional definition of a keypoint, which is no more than a location on an image. Additionally, we may observe a keypoint detector that identifies keypoints of poorer quality when trained using a variable standard deviation in this stage, as the spatial bottleneck is more relaxed.

4.5.4 Keypoint Regressor Architecture

The hourglass network we use for all datasets during this chapter is outlined graphically in Figure 4.3, with parameters for each layer specified further in Table 4.5.4.

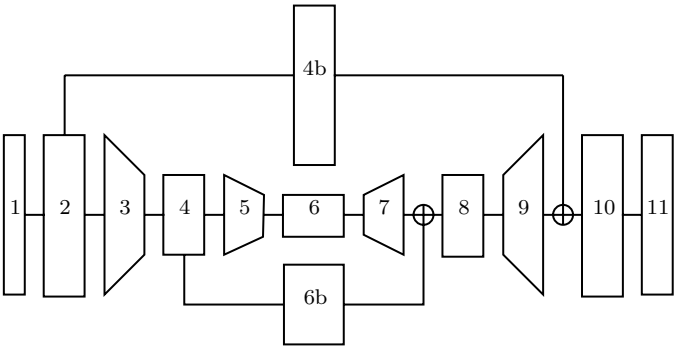


Figure 4.3: A diagram showing how each of the layers in the hourglass network are connected. Direct sum symbols are element wise additions on the feature maps being outputted from the previous layers.

#	Layer Type	Input Size	Output Size	Kernel	Activation
1	Conv2D	3	20	3x3	LeakyReLU
2	Conv2D	20	48	3x3	LeakyReLU
3	MaxPool	-	-	2x2	-
4	Conv2D	48	64	3x3	LeakyReLU
4b	Conv2D	48	48	3x3	LeakyReLU
5	MaxPool	-	-	2x2	-
6	Conv2D	64	64	3x3	LeakyReLU
6b	Conv2D	64	64	3x3	LeakyReLU
7	UnPool	-	-	2x2	-
8	Conv2D	64	48	3x3	LeakyReLU
9	UnPool	-	-	2x2	-
10	Conv2D	48	20	3x3	LeakyReLU
11	Conv2D	20	k	3x3	-

Table 4.3: A table of the parameters of each layer of the keypoint regressor network, specifying their layer type, input size, output size, kernel and activation. The number of each layer corresponds to the numbers shown graphically in Figure 4.3. For all LeakyReLU activations, the alpha value is set to $\alpha = 0.1$.

4.5.5 Downstream Task Implementations

This subsection covers how we have implemented each of our downstream tasks that we wish to train with.

4.5.5.1 Reconstruction Task

Our reconstruction task has two modes, depending on the dataset used. The first mode simply aims to reconstruct the input image from the set of keypoints, after being converted into cleaned heatmaps. This process is shown in Figure 4.4.

When reconstructing from cleaned heatmaps using only convolutions, each keypoint has a receptive field surrounding it and if we use too few keypoints or have a reconstructor with receptive fields that are too small then the keypoints will spread out to maximise the receptive field in order to get a good reconstruction, at the cost of capturing structure in the image. To circumvent this, we use a reconstruction network that splits the convolutions into two channels, one for low (L) frequency, representing approximate colour in the background pixels, and one for high (H) frequency, representing more detailed reconstructions in close proximity to each keypoint. Low frequency reconstruction is done with 5x5 convolutions and one application of 2x2 max pooling, before upsampling at the end when summed with high frequency channel, which only uses 3x3 convolutions. This architecture has an adequate receptive field to reconstruct the input and also allows the reconstruction near keypoints to convey stronger high frequency information. The final reconstruction is made by interpolating L using a bilinear interpolation with scale factor

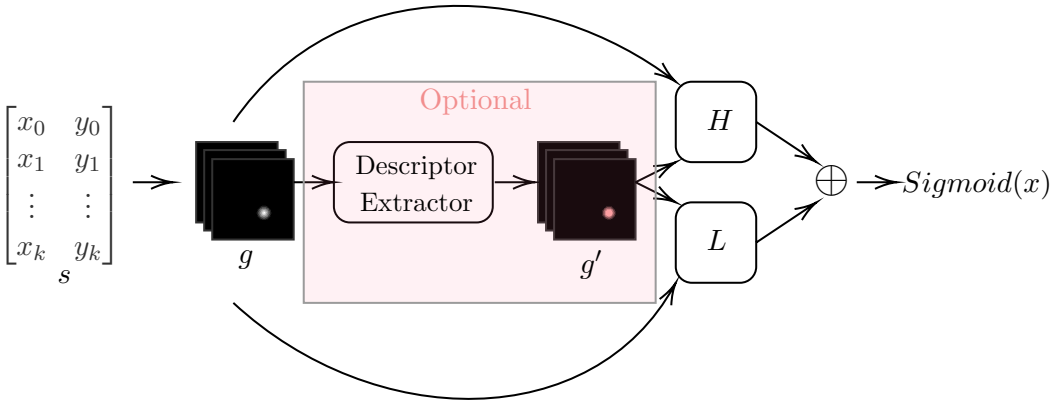


Figure 4.4: Outline of our approach to reconstructing images from keypoints. Input (s) is a list of keypoints, which is converted into Gaussian peaks (g) on a blank heatmap and then fed into the high frequency reconstructor (H) and low frequency reconstructor (L). The final reconstruction is made by summing the outputs of the two sun-networks and passing through a sigmoid function. We have an additional section used to convert Gaussian peaks into feature vectors and then reshaped back into Gaussian peaks with textural information (g') which are fed into H and L instead of the original Gaussian peaks.

2 and summing with H, and then passed through a sigmoid activation layer to push the distribution into a valid pixel range of $[0, 1]$

The reconstructor networks we use for both reconstruction modes are fully sequential networks, and the parameters for each layer are defined in Tables 4.4 and 4.5.

High Frequency Information Reconstructor (H):

Layer Type	Input Size	Output Size	Kernel	Activation
Conv2D	k	20	3x3	LeakyReLU
Conv2D	20	48	3x3	LeakyReLU
Conv2D	48	64	3x3	LeakyReLU
Conv2D	64	32	3x3	LeakyReLU
Conv2D	32	3	3x3	-

Table 4.4: A table of the parameters of each layer of the high frequency reconstructor network, specifying their layer type, input size, output size, kernel and activation. Where k is the number of keypoints, which changes dependent on dataset, and LeakyReLU uses $\alpha = 0.1$.

Low Frequency Information Reconstructor (L):

Layer Type	**Input Size**	**Output Size**	**Kernel**	**Activation**
Conv2D	k	5	5x5	LeakyReLU
Conv2D	5	10	5x5	LeakyReLU
Conv2D	10	12	5x5	LeakyReLU
Conv2D	12	12	5x5	LeakyReLU
Conv2D	12	12	5x5	LeakyReLU
MaxPool	-	-	2x2	-
Conv2D	12	5	5x5	LeakyReLU
Conv2D	5	3	5x5	-

Table 4.5: A table of the parameters of each layer of the low frequency reconstructor network, specifying their layer type, input size, output size, kernel and activation. Where k is the number of keypoints and LeakyReLU uses $\alpha = 0.1$.

In the second reconstruction mode, we train an additional local descriptor extractor so that we are able to reconstruct datasets that contain more detailed textures. Inspired by traditional local descriptor algorithms, we take a circular crop from each keypoint, located via a square crop with diameter $= h = w$, and pixels outside of the circle with diameter of h are zeroed out. This is then fed into a small descriptor extractor network with a bottleneck that is trained from scratch in an end-to-end fashion alongside the other networks. The output of this descriptor network is the same size as the input and these outputs are then shifted back into the locations that the crops were taken from to create textured heatmaps. These textured heatmaps are then fed into the same reconstructor network as we used in the simple reconstruction case.

The descriptor extractor network is implemented as a sequential network and the details of the network parameters for each layer is defined in Table 4.6.

Layer Type	**Input Size**	**Output Size**	**Kernel**	**Activation**
Linear	$3p^2$	512	-	ReLU
Linear	512	256	-	ReLU
Linear	256	v	-	ReLU
Linear	v	$8v$	-	ReLU
Linear	$8v$	$16v$	-	ReLU
Linear	$16v$	p^2	-	Sigmoid

Table 4.6: A table of the parameters of each layer of the local descriptor extractor network, specifying their layer type, input size, output size, kernel and activation. Where p is the size of the circular crop patch and v is the size of the feature vector.

Layer Type	Input Size	Output Size	Kernel	Activation
Conv2D	$k + 3d$	64	3x3	ReLU
Conv2D	64	128	3x3	ReLU
Conv2D	128	32	3x3	ReLU
Conv2D	32	4	3x3	ReLU
Data viewed as b vectors of size $4 * h * w$				
Linear	$4 * h * w$	4096	-	ReLU
Linear	4096	256	-	ReLU
Linear	256	d	-	-

Table 4.7: A table of the parameters of each layer of the distractor predictor network, specifying their layer type, input size, output size, kernel and activation. Where d is the number of images used, b is batch size, h is image height and w is image width.

4.5.5.2 Distractor Image Prediction Task

This task operates as an alternative to a supervised classification, where no labels are required. The idea is to train a network to identify which image the keypoints were extracted from when provided with the true image along with some distractor images. A simple downstream network, is then given the task of looking at the stacked keypoints and images and selecting which index contains the image that corresponds to the keypoints given. We use a multi-margin loss for the classification stage as we find it trains more reliably than using a negative log-likelihood loss.

This task is difficult for our predictor to learn, so typically we use only one distractor image in the prediction. This gives a binary decision for which image the keypoints belong to.

The distractor predictor network we use is a sequential network with layers defined as per Table 4.7.

4.5.5.3 Middle Frame Prediction Task

Predicting which frame is the middle of a sequence is a common task in self-supervised learning, but is normally applied to sequential frames taken from a video. As we would like to use non-sequential data, we mimic the effect of video frames using the TPS warp [28], discussed previously in Section 4.3.2.1, to apply non-linear transforms to our keypoints. We use a non-linear warp instead of a simple affine transform as we gives the network a more difficult job in deciding which frame is the middle. For each image, we apply two warps, where the parameters of the first warp are half of the second, to obtain a set of three images, the original and two warped versions. We then pass all three images through our keypoint detector to get three sets of keypoints, which are

Layer Type	**Input Size**	**Output Size**	**Kernel**	**Activation**
Linear	$6k$	512	-	ReLU
Linear	512	1024	-	ReLU
Linear	1024	2048	-	ReLU
Linear	2048	512	-	ReLU
Linear	512	3	-	-

Table 4.8: A table of the parameters of each layer of the middle frame predictor network, specifying their layer type, input size, output size, kernel and activation. Where k is the number of keypoints.

then shuffled and passed into a simple linear MLP to predict which index contains the middle frame of the warps. A cross entropy loss is used with these predictions and the true indices to obtain our loss.

The reason we use a task to predict a middle frame of the warp instead of a hard constraint on keypoints aligning between warps is to encourage better generalisability by learning the same behaviour as an emergent property.

The middle frame predictor network we use is a sequential network with layers as defined by Table 4.8.

4.5.6 Concentration Constraint

For this additional constraint, we are minimising Equation 4.2 where m is the heatmap prior to the soft arg-max function and g is our cleaned heatmaps. This constraint is only used to discourage the soft arg-max algorithm from exploiting any random noise or border pixels when selecting keypoints, thus encouraging the discovery of a peaky Gaussian heatmap.

$$\sum_{x,y} (h_{x,y} - g_{x,y})^2 \tag{4.2}$$

4.5.7 Hyper-Parameters

We have selected a few hyper-parameters manually to aid our network in the training process. We set our batch size to 64 and train for a total of 15 epochs. We train using the Adam optimiser, with a initial learning rate of 0.001, which is reduced every 5 epochs by a factor of 10. The number of keypoints is selected per dataset and is selected based on the complexity of the shapes found in the dataset.

4.6 Experiments

This section covers the experiments undertook to decide the best loss balancing approach to use, followed by experiments demonstrating success on different datasets and in different scenarios. Further experiments are undertaken to improve our understanding of what the keypoint regressor network has learnt and testing how well our method can generalise to new data.

To evaluate the proposed approach we have selected a range of different datasets with varying properties, including MNIST [64], FashionMNIST [110], Shoes from UT Zappos50k [112], Chairs [1] and Human3.6m [48].

Section 4.6.1 compares different approaches to balancing the range of losses used in our multi-task approach. Section 4.6.2 shows comparison between how our network optimises using single tasks against multiple tasks. Section 4.6.3 looks at the performance of our technique on simplistic toy datasets as a proof of concept. Section 4.6.4 looks at complex datasets that would have real-life applications. Quantitative analysis of our keypoints is detailed in Section 4.6.5. We look into the inputs for the soft-arg-max algorithms in Section 4.6.6, in order to analyse how confident the keypoint detector network is for each keypoint. Finally, Section 4.6.7 is verifying the generalisability of our approach by analysing results out of the training data distribution. Further implementation details can be found in Appendix B.

4.6.1 Loss Balancing Method Selection

As discussed in section 4.4.2.2, we have found multiple different techniques for balancing the losses of each task. After initial experiments found that the Pareto Optimal method [95] gave poor results, as well as being far slower as we require two back-propagation passes, we narrowed down our selection to two approaches, uncertainty estimation and balanced learning with an exponential function mapping.

To test which approach finds better solutions, we have trained a network on the MNIST dataset using the three tasks and one constraint described above, for 15 epochs. We drop the learning rate every 5 epochs and to test reliability, we have completed 10 runs for each approach. The results from this comparison can be seen in Figure 4.5. We have plotted training loss against epoch with a 95% confidence interval, and we can clearly see that the exponential function mapping not only gives a lower loss for all tasks, but also gives more consistent results. To validate these results, we have run the same experiment but using the Fashion MNIST dataset [110] as seen in Figure 4.6. While the results here are not as clear cut, we still see that the overall loss is lower when we train using the exponential function mapping and that we find more reliable results over the 10 runs, with the confidence interval being significantly narrower. We

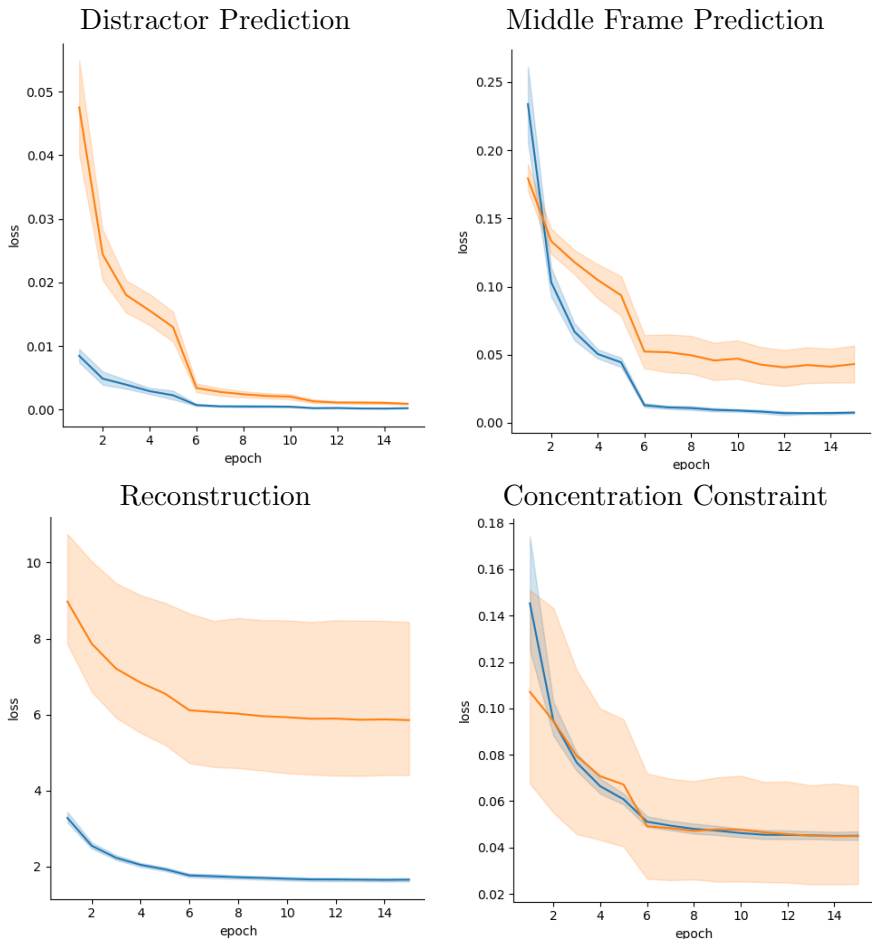


Figure 4.5: Comparisons of task losses when using different loss balancing options, where learnt uncertainty is in orange and exponential function mapping is in blue.

do see that the uncertainty parameter technique does get a slightly lower loss for the concentration constraint, perhaps over optimising this constraint over the reconstruction task. While this may not be an issue, we may find that the keypoints do not capture as much information about the image as the focus is on the feature heatmap constraint instead of solving one of the given tasks.

4.6.2 Single and Multi Task Comparison

Table 4.9 shows how our network optimises to find keypoints to solve the given tasks when solving each single task and when solving multiple tasks at the same time. When using only the distractor task, our found keypoints barely represent that shape which implies that our network is operating like a hashing function for the image as a whole, which is then used to identify the correct image.

The reconstruction task gives the best single task performance, distributing the keypoints more evenly than the other single task methods, but still has issues with clumping points in smaller regions. The middle frame prediction task gives keypoints that have

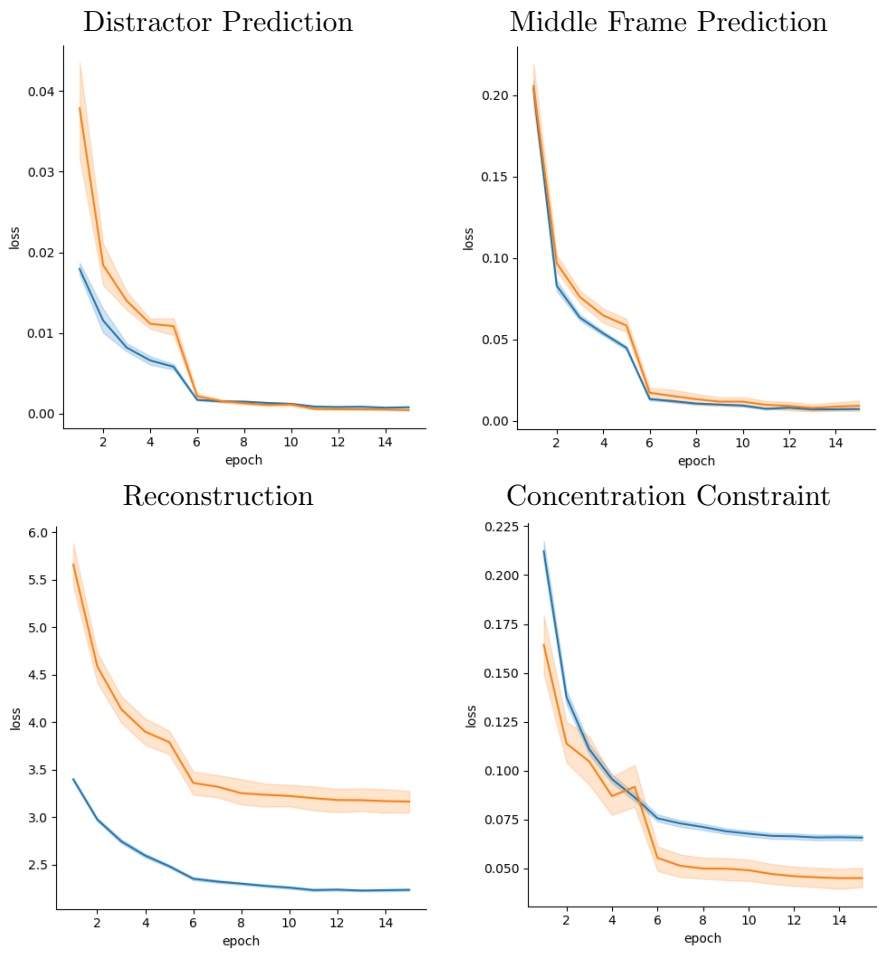


Figure 4.6: Comparisons of task losses when using different loss balancing options on FashionMNIST, where learnt uncertainty is in orange and exponential function mapping is in blue.

little travel from the centre of each image, but are not completely central. This is due to our network needing to place points in areas that move to capture the transform but this capture has no need to be in diverse areas of the image, so simply finds a solution that locates one easy to locate feature near the centre of the image. Finally we have our combined multi-task method that spreads the keypoints evenly around the shapes in the image and also represent the image with enough detail to select correctly in our distractor task. These keypoints are also able to trace the transforms to accurately predict our middle frames.

Not only are the qualitative results showing an improvement on found keypoints, multi-task learning leverages extra information so solve the individual tasks, we observe better task performance when all tasks are trained together as seen in Table 4.10. The same pattern can be seen in Table 4.11, which compares training losses in the single and multi task cases.

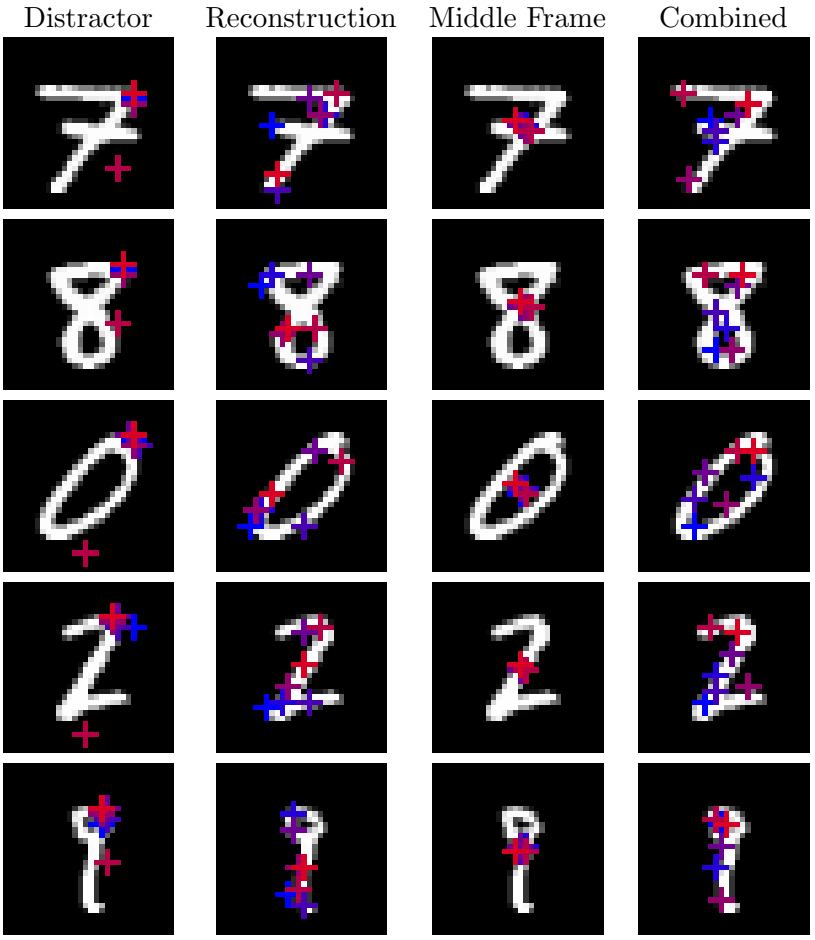


Table 4.9: Comparisons of keypoints found to solve different tasks individually and when located using multi-task learning. Qualitatively, the multi-task learning approach shows keypoints that capture the best structure, followed by the reconstruction task. Colours are a scale between blue for keypoint at index 0 to red for keypoint at index k, and are consistent between examples.

Table 4.10: Comparisons of task performance when the network is trained on the individual tasks and when trained on all tasks combined.

	Distractor Accuracy (%)	Reconstruction Error	Middle Frame Accuracy (%)
Single	98.75	0.0588	98.30
Multi	99.75	0.0291	99.65

Table 4.11: Comparisons of training losses of single task training and multi task training.

	Distractor Loss	Reconstruction Loss	Middle Frame Loss
Single	0.0089	3.7941	0.0506
Multi	0.000518	1.8487	0.0109

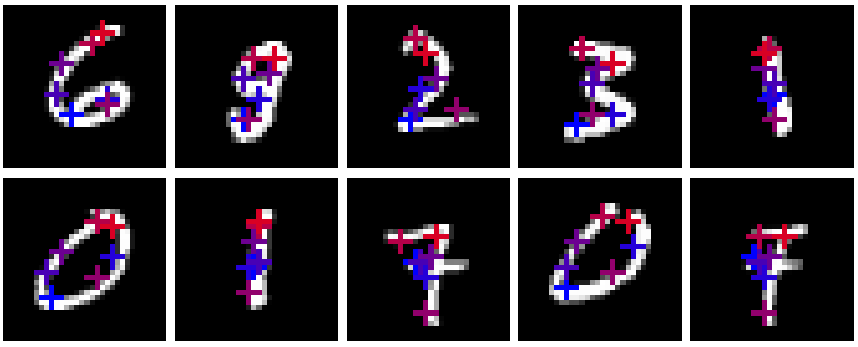


Table 4.12: Examples of keypoints found on the MNIST dataset. Colours are a scale between blue for keypoint at index 0 to red for keypoint at index k. Further examples available in Appendix B.1.1

4.6.3 Results on Toy Datasets

This subsection looks into our results when we use simple datasets, which consist of greyscale pixels and are small in size (28x28 pixels). Due to the simple nature of the data, we do not require the feature extractor in the reconstruction task. We are using a value of $\sigma = 0.9$ for constructing Gaussian blob heatmaps for these simple datasets.

4.6.3.1 MNIST

The MNIST dataset gives us a good insight into how useful this technique can be. We train our keypoint detector as described above using the entire dataset rather than training on individual characters. As a result, we are not just optimising for one single structure, but a wide variety of shapes, giving the network a much harder problem to solve. As seen in Table 4.12, we have managed to find a convincing set of keypoints that convey the shape of each and every different digit in the dataset.

4.6.3.2 Fashion MNIST

Just as we have done with the MNIST dataset, we are aiming to find keypoints that represent the structures of all of the different types of clothing in the Fashion MNIST dataset. We show in Table 4.13 that while there is a massive variance of shapes, each item of clothing has keypoints that successfully represents their structure.

4.6.3.3 Quantitative Evaluation

To validate our qualitative results shown above, we have used a novel downstream classification task. We train a simple neural network to classify our learnt keypoints using

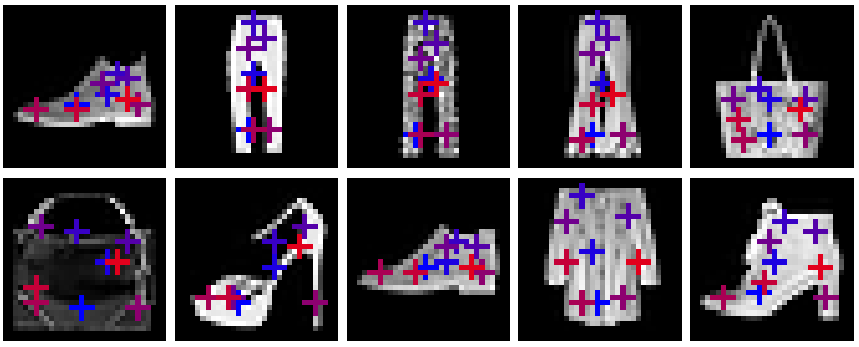


Table 4.13: Examples of keypoints found on the Fashion MNIST dataset. Colours are a scale between blue for keypoint at index 0 to red for keypoint at index k. Further examples available in Appendix B.1.2

labels found in the dataset and we are able to classify 7 keypoints taken from MNIST with 97.7% accuracy and 9 keypoints from FashionMNIST with 84.5% accuracy.

4.6.4 Results on Complex Datasets

The previous experiments were run on datasets that had no colour and limited textural information, which meant we could use a simple reconstruction task that requires only keypoints to reconstruct the image. However, most datasets are not as easy to work with and require extra information in order to solve our reconstruction task. For the following examples, we have extracted patches around the keypoints and learnt descriptor vectors for each to feed into the reconstructor network. We are using a value of $\sigma = 1.0$ for constructing Gaussian blob heatmaps for these real world datasets.

4.6.4.1 Shoes

We can train our network using shoes from the UT-Zap50k dataset [112], as seen in Table 4.14, to find keypoints on a variety of different shoes that represent the shape of the individual image. Individual keypoints are well matched between the shoes, and we are clearly able to implicitly learn the generalised structure and how to fit that structure to any example.

4.6.4.2 Human3.6m

This approach also gives good results for human pose estimation, using the Human3.6m dataset [48], as seen in Table 4.15. We can see that this technique consistently locates body parts, such as the head with the purple keypoint, and spreads the keypoints sufficiently to capture the structure of the human body. Although these are not perfect results, with the most obvious error being the issue of where to place the keypoints on

Table 4.14: Examples of keypoints found on the Shoes dataset. Colours are a scale between blue for keypoint at index 0 to red for keypoint at index k. Further examples available in Appendix B.1.3

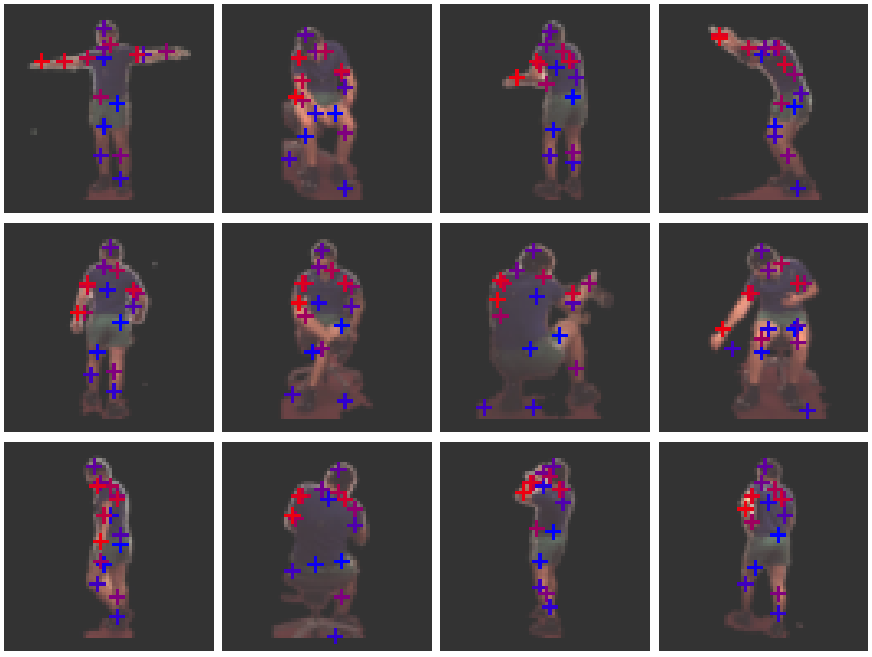


Table 4.15: Examples of self-supervised keypoints found on the Human3.6m dataset. Colours are a scale between blue for keypoint at index 0 to red for keypoint at index k. Further examples available in Appendix B.1.4

a human who is only showing their back. The labels in the dataset invert the keypoint order over the y-axis when the back of a subject is shown, but this is not the case using this approach as no prior informs the network that these are three-dimensional objects.

4.6.4.3 Semi-supervised Regression Metric

In line with Zhang et al. [118] and Thewlis et al. [100], we have tested our approach using a semi-supervised step that regresses our predicted keypoints to ground truth poses using a simple linear regressor. We then measure the error as a percentage of image size per keypoint. As our keypoint detector network cannot deal with the differentiating

Table 4.16: Comparison of regression errors with comparable papers. Distances are measured as % error of image size.

	Human3.6M
Thewlis et al. [100]	7.51
Zhang et al. [118]	4.14
Ours	6.35

Landmark Location	Mean Error	Standard Deviation
Middle Hip	5.26	4.96
Left Hip	5.69	5.12
Left Knee	5.94	4.93
Left Foot	8.59	7.06
Right Hip	5.46	5.00
Right Knee	6.59	5.50
Right Foot	8.61	7.17
Mid Back	3.81	2.82
Head	4.86	4.75
Mid Shoulder	3.37	2.96
Right Shoulder	3.77	3.09
Right Elbow	6.94	5.27
Right Hand	10.94	7.55
Left Shoulder	3.54	3.00
Left Elbow	6.75	5.11
Left Hand	11.53	7.51

Table 4.17: Mean regression errors on a per landmark basis. Errors are measured as % of image size.

front/back in our images, we are manually flipping the input images and corresponding ground truth points in examples where the left shoulder keypoint is further right in the image than the right shoulder.

We can further break this down to a mean error on a per keypoint basis, as shown in Table 4.17. We also show these results visually in Table 4.18, comparing the regressed keypoints to their ground truth counterparts.

4.6.4.4 Action Recognition Metric

For the Human3.6m dataset, we do a similar classification task to the novel task classification as described in Section 4.6.3.3. But as the dataset only contains action recognition labels, we classify on sequences of keypoints taken from the video frames and feed the sequence through an LSTM before classifying based on the hidden state of the final LSTM layer. Figure 4.7 shows the action recognition accuracy using this approach with our predicted keypoints taken from our keypoint regressor compared to the ground truth keypoints taken from the dataset.

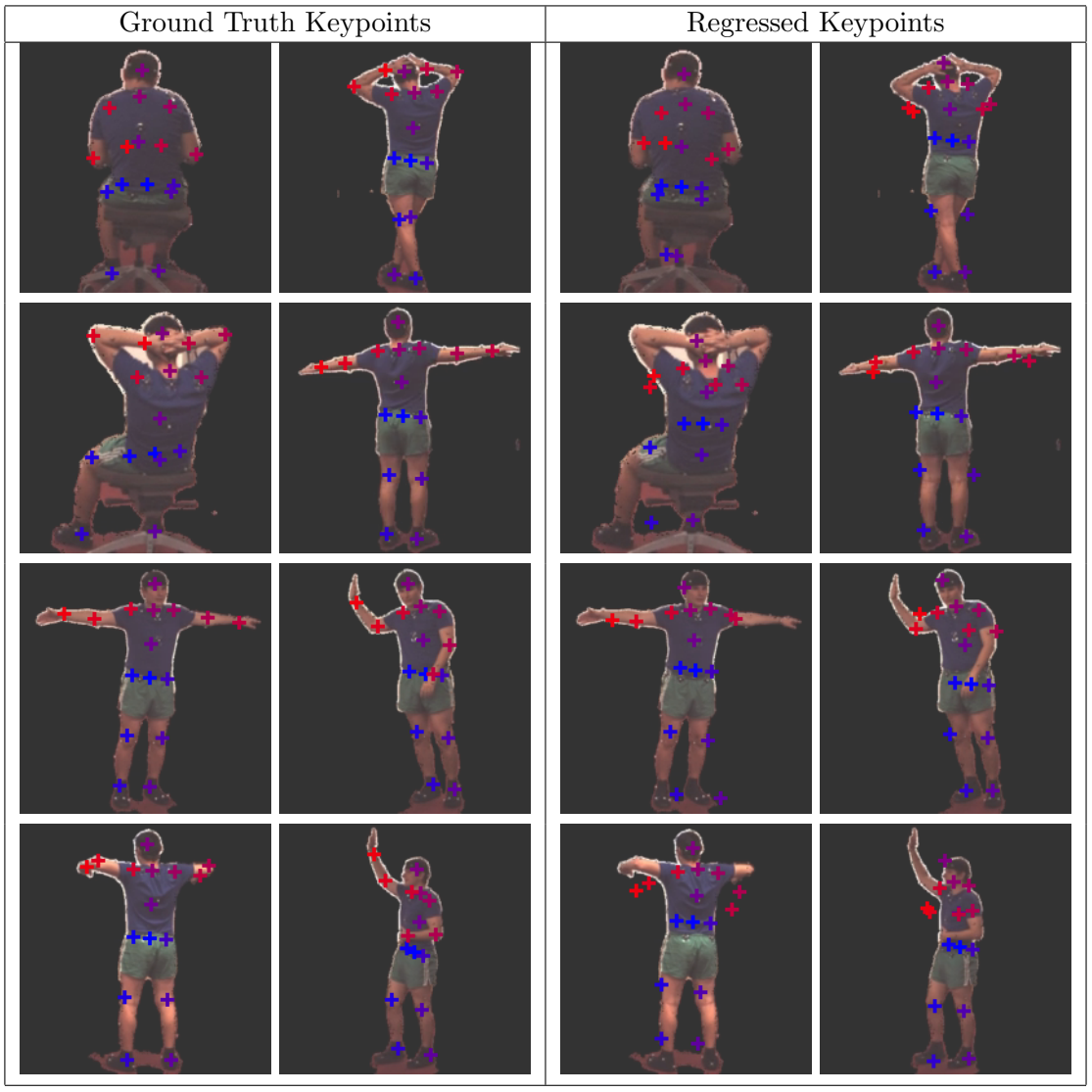


Table 4.18: Examples of regressed keypoints found on the Human3.6m dataset compared to their ground truth counter parts. Colours are a scale between blue for keypoint at index 0 to red for keypoint at index k, and are consistent between columns. Further examples available in Appendix B.1.5

We train on sequences of 32 samples of keypoint shapes, sampled with frame striding of 20 frames, and starting at a random point through each video. The action recognition labels included in the dataset are from 17 different classes, giving the accuracy of randomly guessing at 5.88%. Our results give a mean accuracy of 56.2% showing our keypoints have clearly captured a good amount of information from the images, not being far behind the 62.1% mean accuracy found using ground truth data. It is worth noting that our regressed keypoints do not leverage any sequential information, keypoints are taken from each video frame in isolation making these results even more impressive.

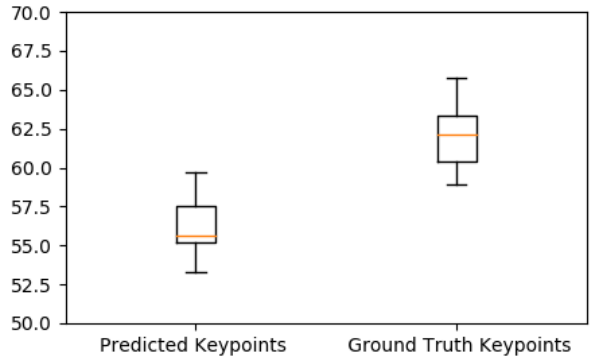


Figure 4.7: Box plot showing action recognition accuracy when we train an LSTM based classifier on the keypoints taken from our keypoint detector and from the ground truth points from the Human3.6m dataset. Data is showing distribution of results from training 10 LSTM based classifiers.

Table 4.19: Comparisons of task performance for each dataset being tested

	Distractor Accuracy (%)	Reconstruction Error	Middle Frame Accuracy (%)
MNIST	99.75	0.0291	99.65
FashionMNIST	99.97	0.0340	99.80
Shoes	89.82	0.2183	99.86
Human3.6m	98.21	0.0138	99.54

4.6.5 Further Quantitative Results

While we have seen in the previous section the locations where our networks have placed keypoints on images, we would also like to know how well our networks can solve the tasks that we have given them using those keypoints. Table 4.19 shows the task performance for each of the networks and we can see that on the whole, the tasks have been solved well. Notable outliers are the performance of the shoes dataset on the distractor and reconstruction tasks, which we believe is due to the large variance in the textural data but low variance in the shape data in this dataset, making both of these tasks difficult to solve.

4.6.6 Keypoint Confidence

By looking at the outputs from the keypoint detection network before the soft arg-max algorithm is used to extract numerical keypoints, we can learn more about how the network is locating points. In Table 4.21, we can see heatmaps for some of the found keypoints on the Human3.6m dataset. The second column shows a point that roughly tracks the hip of the target, either left or right depending on the orientation of the person in the image, and does so with a relatively strong confidence as seen by the small range of bright pixels in the heatmap. An example of a less confident point is in the third column, where our heatmap covers the majority of the person in the image. We

Table 4.20: Means and standard deviations of eigenvalue ratios taken from covariance matrices of Gaussians fit to heatmaps from the Human3.6m dataset. Columns align with those in Table 4.21.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Mean	0.7947	0.6136	0.2110	0.4382	0.3647	0.5373	0.3878	0.4302
SD	0.0817	0.0637	0.1214	0.0913	0.1686	0.1753	0.1797	0.2205

see another error case in the final column, where we normally find a confident point on the left hand, however in the second and fifth row, we find the back and the knee with a wider heatmap range.

Table 4.22 shows keypoint confidence for all 8 keypoints learnt on the shoes dataset. Compared to the results in Table 4.21, we can see that the network appears to be uncertain of the location for the majority of the shoe landmarks. The reason for this may be because of the high variance found in the textural information of this dataset, while the shape information has a relatively low variance. As we use an Hourglass network that leverages a combination of local and global information, the network is free to learn a global structure rather than searching for local features. Some local features have been successfully detected, but only those that are common for any kind of shoe such as the heel and the toe.

For a deeper analysis of the shapes of the features heatmaps that our network has learnt, we have applied a Gaussian fitting technique to learn the mean covariance matrix for each keypoint over the batch. Table 4.20 shows the mean ratio between greatest and smallest eigenvalues taken from an eigen decomposition of the covariance matrices of Gaussians fit to our outputted heatmaps from the Human3.6m dataset. A small ratio here means that the heatmaps are stretched and resemble long features whereas high ratios resemble relatively circular heatmaps. The values at indices 3 and 5 resemble those keypoints that find the legs on our dataset which explains the low ratio here.

Table 4.21: Heatmaps showing activation before the soft arg-max function found on the Human3.6m dataset. Colours represent the same colour scale as the keypoints shown in Table 4.15.

Table 4.22: Heatmaps showing activation before the soft arg-max function found on the Shoes dataset. Colours represent the same colour scale as the keypoints shown in Table 4.14.

	MNIST	FashionMNIST
NetA	98.4%	82.3%
NetB	93.1%	84.5%

Table 4.23: Classification Accuracy using Keypoints Detected from networks trained on each dataset. NetA has been trained using MNIST and NetB has been trained using FashionMNIST.

4.6.7 Verifying Generalisability

One of the aims of using a multi-task approach is to train a network to find generalised keypoints. How can we evaluate for generalisability in this self-supervised keypoint detection context? The ability for our keypoints to be used to solve arbitrary tasks gives us some idea but to push the limits of evaluating generalisability, we can attempt to use testing data that lies out of the training data distribution. Taking a pair of keypoint detector networks, one trained to locate 9 keypoints on MNIST characters and the other to find 9 keypoints on FashionMNIST, and swapping the inputs to each network, we can see how each react to data outside of the expected distribution. As seen in Table 4.23, accuracy for classifying each input using only their keypoints drop after switching datasets drops but remains high, especially on the network trained on MNIST. Table 4.24 shows the keypoints found overlaid onto the images and we can clearly see a reasonable structure captured in both of the cases after swapping.

We also see successful generalisation in complex data. We have shown our results earlier of capturing structures of shoes, and we can use this network to identify structures of

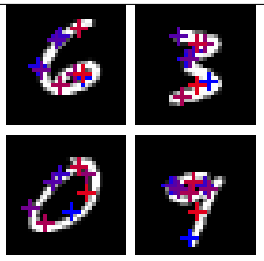
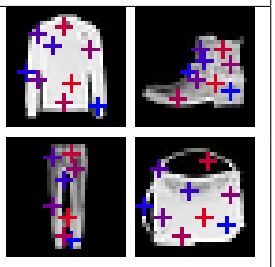
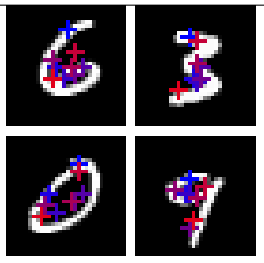
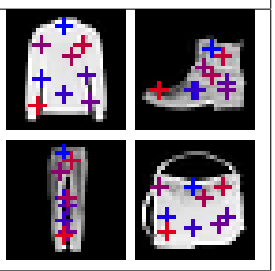
	MNIST	FashionMNIST
NetA		
NetB		

Table 4.24: Comparisons of keypoints found on the MNIST and FashionMNIST datasets, where NetA has been trained using MNIST and NetB has been trained using Fashion-MNIST. Colours are a scale between blue for keypoint at index 0 to red for keypoint at index k, and are consistent between examples from the same network.

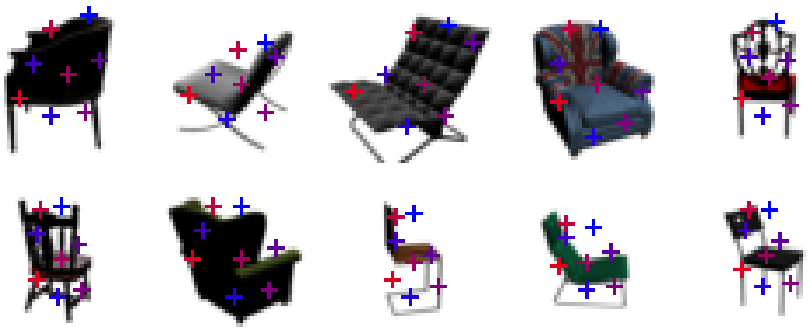


Table 4.25: Examples of keypoints found on the Chairs dataset when the network was only trained on Shoes. Colours are a scale between blue for keypoint at index 0 to red for keypoint at index k, and are consistent between examples from the same network.

Table 4.26: Comparisons of task performance for each task when tested on Shoes and Chairs using a model trained on Shoes

	Distractor Accuracy (%)	Reconstruction Error	Middle Frame (%)
Shoes	89.82	0.218	99.86
Chairs	56.12	0.226	98.15

chairs [1] as seen in Table 4.25. We are helped here as both of these datasets have single objects in the foreground and a plain white background, however there is solid evidence of generalisation here. As classification of this dataset would be unreasonably difficult using just 8 derived keypoints, we have instead evaluated our generalisability success via downstream task performance. Table 4.26 shows the downstream task performance for each task when this model is tested on the Shoes and Chairs dataset. As could be expected, there is a drop in performance for all tasks, with the distractor task showing the greatest drop.

4.7 Discussion

As seen in the qualitative results from the experiments above, we are able to distil shape from a wide variety of image subjects with no prior knowledge. We find that our desirable properties emerge naturally as the network learns a general keypoint structure that can be used to solve all of our selected tasks. While these properties could have been obtained through creating losses that directly optimise these requirements, as shown by Zhang et al. [118], we have observed better generalisation by indirectly learning these properties. If a neural network is able to locate keypoints that contain a strong structure found in the data with absolutely no priors of what structure to look for, then we have distilled this information purely from image data. The power of applying multi-task learning to this problem is that the network must learn this strong generalised structure in order to be able to solve the set of varied tasks.

Similar approaches [100, 118] use a large amount of hyper-parameter tweaking, including selecting different neural networks based on which dataset is used, whereas we aim to generalise keypoint regression in a truly domain-agnostic approach. The result is being able to test on a different dataset to the one used in training and still detect a strong set of keypoints that capture structure seen in the image successfully.

Standley et al. [96] study which tasks should be learnt together in a multi-task environment, but in the context of supervised tasks and with a time-accuracy trade-off. In the context of self-supervised learning, we face a different scenario where we would like to select tasks that help us learn a better representation and not focus solely on the single task performances.

We theorise that the success of our technique corresponds to the choice of tasks that are selected to train with. A good set of tasks should be:

1. Varied, the optimal solutions for each individual task must be contrasting enough that we cannot overfit to all tasks at once.
2. Difficult to optimise, easy tasks will either focus the optimisation in the downstream tasks to obtain good results using poorly chosen keypoints, but if we choose hard tasks, we require well placed keypoints to obtain good training performance. At the same time, overly difficult tasks will not be able to be solved using any keypoints, giving little to no training signal to the keypoint detector network.
3. Balanced in how the tasks are oriented towards our objective. While we do not want them to be completely unrelated as there will be nothing to learn, we also do not want them to be so related such that we overfit and lose generalisability.

4.7.1 Common Pitfalls in Self-Supervised Keypoint Detection

Training a network to discover keypoints that can successfully capture the structure of an object is by no means trivial. Keypoint detection is essentially a mapping from an image to a list of 2D coordinates that denote the pixel or sub-pixel location where a desired feature is. Traditionally keypoints have been described as having two desirable properties; robustness, how invariant keypoints are to changes, and repeatability, how successful we are at finding the same feature between examples. Robustness can be tested by applying a transform to an image and observing if the keypoints to follow that transform but repeatability is more difficult due to the lack of labelled data. An additional desirable property is for the keypoints to disperse to capture areas over the entire image space. We observe that in some cases, the keypoints found are either erroneous or have undesirable features and some of the reasons for this are discussed below.

4.7.1.1 Local and Global Information

It is common to find deep learning based techniques for pose estimation and keypoint regression to use an hourglass network, as it allows for leveraging both local and global shape information to decide where a keypoint should be. This tends to give us good results, especially in a supervised environment [85], however in the self-supervised case, we can see some undesirable behaviour. The balance between the reliance on local and global information can force our trained network to base keypoint detection almost entirely on *where* it is in the image and not on *what* textural information is at that point. We find a lack of robustness in difficult examples where the global structure differs from the expected range of poses learnt from the given data. If an image is rotated 90 degrees, then we would expect the keypoints to move with the image, but in almost all cases, we see a massive failure state. A successful keypoint detector should be able to deal with this case as the textural information is still available in the image.

4.7.1.2 The Responsibility Problem

The second failure state we see is relating to the responsibility problem [115]. How does the network decide which keypoint should capture information from each area of the image? We see more subtle failure states than before that are attributed to this problem, for example in Zhang et al. [118]'s results on the Human3.6m dataset. When the person in the image has raised a hand above their head, the keypoint that normally tracks the head is now responsible for tracking the hand. This gives us another clue about what we are actually learning in our hourglass network, and the local information about the head is less dominant than the global shape information marking the top of the structure. Perhaps taking inspiration from Zhang et al. [115], the solution in a self-supervised keypoint detection environment is in learning a *set* of keypoints instead of a *list*. With no prior knowledge, searching for ordered keypoints where each keypoint locates the same region of an object is unrealistic when we have datasets of real life objects that are free to move and rotate in a three dimensional world. If we learn keypoints that are deliberately unordered and order them if required for downstream tasks, we may expect to see a more accurate representation of shape, as a common structure can be learnt and then keypoints matched to features later.

4.7.1.3 Bilateral Symmetry

Bilateral symmetry creates a persistent issue when we have data of the object rotated, as is common in human pose estimation. We would like to keep consistency where we will have a keypoint that consistently tracks the left hand but we seen in our results and others [100, 118], this is not the case. The image information to work out the orientation of the subject is present, but our network is not correctly leveraging it and

without supervision, struggles to know that we could be looking at the back of the object.

4.7.1.4 Collapse During Training

One common issue with learning keypoints with no supervision is the degenerate solution where all keypoints collapse to $(0, 0)$ if trained without a prior that forces them to spread. In previous approaches, this is fixed with a specific separation constraint that minimises overlap of keypoints. While this initially fixes the collapse, this sort of prior can lead to some undesirable properties such as being unable to find two features that are within the range of separation set by the constraint. Ideally, we would like a softer method, that aims to capture the most amount of information instead of specifically spreading keypoints. This naturally fixes the collapse without adding any unforeseen consequences such as our keypoint detection failing to find two points in the same area. Our method aims to naturally solve this problem due to the multi-task nature of the optimisation, having multiple training signals aids in the training method by discouraging getting stuck in a local minima in our loss landscape.

4.7.1.5 Occlusion

Occlusion is a persistent challenge over a wide range of computer vision problems, with keypoint detection being no exception. Locating a keypoint that cannot be seen is naturally difficult, but some supervised approaches use shape models to predict hidden keypoints by using nearby visible landmarks [20]. When using self-supervision, we no longer have the prior knowledge of a strict structure but we can leverage information from other images in the dataset where occlusion does not occur. If we learn a solid representation, with the use of a carefully selected prior, then we may be able to take inspiration from Cootes et al. [20]'s work and infer where keypoints should be based on where we can find the neighbouring points.

4.8 Conclusion

In this chapter we have built keypoint detectors that are trained entirely through self-supervised methods without any hard-coded constraints to avoid undesirable behaviour as seen in previous work and have shown the effectiveness of applying multi-task learning to this area. Our results show increased generalisation over using a single task based approach, to the extent that performance does not drastically drop if we train on one dataset and test on a completely disjoint dataset. We believe this technique has value in landmark detection applications where little to no ground truth data is available and

for general spatial information extraction in a self-supervised setting. Finally, we have addressed some of the common pitfalls in this area, and given some suggestions of how these can be solved.

We have established a solid multi-task framework for keypoint detection but it is clearly adaptable by selecting different tasks. By using specific tasks, we may find that we have new emergent properties which will be desirable in different applications. One example of this, which we will explore in Chapter 6, is in 3D pose estimation. We will attempt to add this property by adding another task that takes the learnt 2D points and predicts the z coordinates to add depth to the learnt spatial representations. Another potential task could be checking for keypoint consistency between video frames to track an object as it moves through the scene. Leveraging this temporal information may help us locate better keypoints, such as those that lie on points of articulation in pose estimation or pushing points closer to the edges of a shape to capture a stronger structure. There is more work to be done on the analysis of multi-task learning in this application. We know that our chosen tasks give us desirable properties, but it is not known what properties each of these tasks gives us, and how the combination of the tasks changes the outputted keypoints. How we optimise the combination of tasks is another area of future work that may help us improve our results, as discussed in Section 4.4.2.2, loss balancing is key to finding a solution that satisfies all tasks.

While this chapter has focused on an abstraction of keypoint detection, allowing for generalised keypoints to be located for any structure, the purposes of keypoint detection in this thesis are more specialised with the aim of locating articulation models. To find keypoints that capture articulated structures in images, we will be investigating how to apply a carefully selected prior. Our focus is still on finding a generalised solution, so our prior cannot be too strong as to work for only one kind of articulation model, but also must not be too weak, as to not capture enough information. Chapter 5 will explore how priors can be applied to generalised spatial representation estimation, and additionally, its applications in keypoint lifting, as previously discussed in Chapter 3.

Chapter 5

Using Bone Rigidity as a Generic Prior

The previous chapters have shown the applicability of self-supervised learning to both keypoint detection and pose lifting, without the requirement of any strong priors. But solving an ill-posed problem such as self-supervised estimation of a 3D articulation model from an image, comes with difficulties without the use of some prior knowledge.

Typically, approaches such as these use strong priors, as demonstrated by Jakab et al. [50]. These can achieve impressive results for human pose estimation, but do not generalise to allow for generic articulation models.

This chapter aims to answer RQ4, with the primary aim of identifying a suitable prior required to solve the generalised articulated 3D pose estimation problem, with the aim of using only the minimum required prior information, to prevent an approach that is overfitted to a subset of articulation models.

5.1 Introduction

Careful selection of a prior is essential to the success of our generalised articulation models estimation approach. Too strong a prior will limit us to only a subset of the articulated models we are interested in, but too weak will not contain enough information to provide good results.

This chapter will start with an analysis of existing priors before creation of a new generic prior. We then show how this simple prior can be used within a differentiable operation to identify the connectivity of keypoints within a dataset, initially for two-dimensional points before extending this to 3D. Using this differentiable algorithm, we create an approach for pose lifting in which we look for keypoint connectivity and apply

our prior to estimate depth. As this is only half of the 3D pose estimation pipeline, we also demonstrate how this prior can be used for the keypoint detection stage of the pipeline as an approach for placing points on articulation and limb end points. We finish this section with a discussion of our prior and implementations, and highlight potential problems with using this prior.

5.2 Designing a Generic Articulation Prior

In the context of self-supervised learning, a prior is a restriction based on some knowledge of the data, before the data has been observed. We require a prior to train our network using self-supervised deep-learning, due to our requirements to use a variety of datasets that may not be labelled, but balancing the strength of such a prior is paramount to the success of our approach. We must be careful to select a prior that is generalised to such an extent that it can be applied to any articulated model, while providing enough information to solve a naturally ill-posed problem.

Figure 5.1 is a visual demonstration of this trade-off. If a prior makes stronger assumptions about the world, the performance of the approach increases. But this is at the cost of the applicability of our approach, thus we may struggle to translate into a new domain or onto a new unseen dataset.

But this is not unique to self-supervised articulation models, this theory applies to any self-supervised learning where a prior is required to find a satisfactory solution or to avoid an ill-posed problem. Even in supervised learning, some problems require additional knowledge about the world to supplement labelled data. Strong priors tend to be desirable due to the increase in performance, but it is apparent that this can often be essential to the generalisability and applicability of an approach. We must also

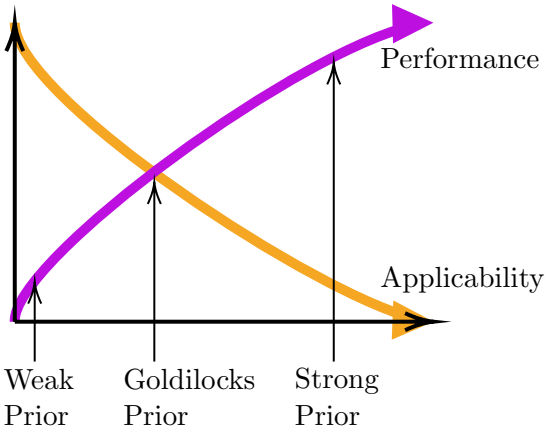


Figure 5.1: A simple illustration of how increasing the strength of a prior both increases performance but reduces applicability of the approach. We aim for the ideal trade-off in the middle, which represents the Goldilocks Prior.

be careful to not select too weak a prior, as we need enough information to make the problem solvable.

5.2.1 Requirements of our Prior

Our prior must make assumptions that are common for all articulated models that may be subjects of images. The requirements for our prior are as follows:

- Generic to all articulation models
- Reliant only on information found within unlabelled monocular images
- Contains knowledge of the 3D world to allow for location of three-dimensional models

This small set of requirements will guide us as we design a prior that will assist in finding generalised articulation models without making unnecessarily strong assumptions.

5.2.2 Comparative Priors

Previously, we used very weak prior knowledge of 3D articulation models, where the only assumption made is that every keypoint has a corresponding z co-ordinate, as it exists within a three-dimensional world. We have seen that this prior can give us a good 3D model when used as the basis of an adversarial pose lifting approach, but can sometimes be prone to the inverse pose problem and is both unreliable and slow to train.

One option for providing the necessary knowledge is to add a second viewpoint of the same scene. Kocabas et al. [59] apply this theory to 3D pose estimation by predicting two views of the same subject using a supervised 2D keypoint detector, before combining them using Epipolar geometry to create an estimated 3D pose for use with their 3D pose estimation network. While they gain promising results, this style of approach requires multiple view data for training, which is an assumption about the data that we wish to avoid to maintain generalisability.

A weakly-supervised approach by Chen et al. [14] introduces the injection of a robust 3D prior to locate 3D human poses from images. They use a pre-trained 2D pose estimator to obtain 2D skeleton maps which they regard to be a tree-structured kinematic graph representing a person, and group together multiple viewpoints into a shared representation in the latent space. Both multi-view geometry and pre-trained supervised models are elements that we are aiming to avoid when implementing our generalised system.

Another prior used in multiple approaches considers the angles between pairs of points [61, 90], and encourages keypoints to be estimated such that they obey the range of

valid angles. This is simple but effective, however will not be applicable to our scenario, as we do not know the structure of the articulation model beforehand, so setting ranges of valid angles would not be possible.

5.2.3 Bone Rigidity Prior

Taking inspiration from properties common to any articulated object, we have designed a bone rigidity constraint as our prior assumption to base our approach. If we have keypoints that capture either articulation points or the endpoints of limbs, then our assumption is that we should have fixed distances between all pairs of joints that are connected by a rigid bone. We will also make the assumption that our joints, if viewed as a graph, are connected as a tree and thus contain no cycles.

How can we exploit our bone rigidity prior to create a method for locating generalised articulation models? The remainder of this chapter will firstly design an algorithm based on this prior which identifies connectivity of keypoints, before demonstrating an implementation of both self-supervised 2D to 3D pose lifting, as shown in Section 5.4, and 2D keypoint detection in Section 5.5. We will show how this prior is suitable for proving sufficient information to solve both of these tasks, while not overfitting to a single category of articulation model.

5.3 Determining Joint Connectivity

Knowing which keypoints are connected gives valuable information for downstream processing and for creating a loss function to train a self-supervised network. While the joint connectivity information may be available in a labelled dataset, given our priority to learn in a self-supervised fashion, we will choose not to use this and aim to derive this information instead. Inferring connectivity helps with the generalisability of our approach as a new dataset of articulated subjects will function identically.

Another approach for estimating the connectivity of our points is to learn a connectivity matrix via a neural network. However, if we are learning to place keypoint on an image and learning to estimate connectivity at the same time, we may find our problem does not have enough constraints, and thus we may find degenerate solutions.

However, using deep learning to locate the connection matrix is not required as by using our bone rigidity prior, we can easily determine connectivity. This is because any two joints connected by a rigid limb will have a fixed distance between them. For this to hold, we must make the assumption that we have managed to estimate keypoints that lie on joints. This will be covered in more detail in 5.5.

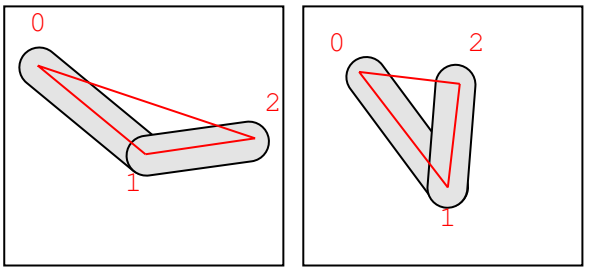


Figure 5.2: An illustration of an elbow-like joint built of two rigid limbs with articulation connecting them. As demonstrated by the red lines, the distance between (0,1) remains constant as does the distance between (1,2), but the distance between (0,2) does not remain constant when the pose of the articulated joint changes.

5.3.1 Limb Variance Minimum Spanning Tree

Our novel approach is to derive keypoint connectivity in a non-learnt method, which has numerous advantageous, such as a lower reliance on locating this pattern from data. This method uses our bone rigidity prior to estimate which keypoints are connected by calculating a Minimum Spanning Tree (MST) using Prim's algorithm [33] over a pairwise distance variance matrix. This idea is based on the fact that two points of articulation in a articulated model containing rigid bones will be located in three-dimensional space at an equal distance from each other when viewed over a batch of different examples. This simple idea is demonstrated in two-dimensions in Figure 5.2. Thus if we take a batch of data points, the variance across the pairwise distances will be smallest between points that are connected by a rigid limb. When we combine this knowledge with the assumption that the articulation models we are interested in are structured as a tree, it is clear that using a minimum spanning tree over this data will produce a matrix that represents the pairs of joints connected by limbs.

We calculate this by taking our batch of keypoints $[B, K, 3]$, and computing a pairwise distance matrix $[B, K, K]$ which gives the distance between every pair of keypoints. Taking the variance of this matrix over the batch dimension gives us a $[K, K]$ matrix of pairwise distance variance. With this matrix, we can now calculate the MST and if we make the assumption that our three dimensional keypoints lie on joints of a rigid articulated structure, this tree will have identified our connectivity matrix.

5.3.1.1 Implementation of a Differentiable Minimum Spanning Tree

Implementing a minimum spanning tree that is also differentiable comes with difficulty. The standard MST algorithm relies on an arg-min operator to locate the best index at each step, which is typically non-differentiable. The relaxation of the arg-min operator, used previously for converting heatmaps into keypoints in Section 4.5.2,[1] can be used to

[1] We previously used the soft-arg-max operator, but the negation of inputs is sufficient to re-use the algorithm.

estimate the arg-min function while continuing a gradient through our algorithm. We use a very low (0.000001) temperature parameter with the soft-arg-min operator to tend towards an integer value. But an unfortunate side-effect of the soft variant of the arg-min function is that it returns real values, which is useful in keypoint estimation for location of sub-pixel keypoints, but not when referencing an index of a matrix as required by the MST algorithm. To circumvent this, we have used a non-differentiable rounding function with the addition of the gradient pass-through trick [2] to round the soft-arg-min output to the nearest index. This results in a good approximation of the minimum spanning tree but can still contain errors in the rare edge case that two values in the limb variance matrix are identical. We use a final workaround here that simply adds a small amount of random noise to the data in the matrix before we run the algorithm to circumvent this scenario.

We implement our algorithm in Python using PyTorch, taking inspiration from Prim's algorithm [33] for finding a minimum spanning tree, but adapted to allow for gradients to flow through it.

```
# A method for finding the minimum spanning tree from a matrix of weights
#   m of type Tensor and shape [K * K] where K is number of keypoints.
    Represents our weight matrix for a graph.
def find_mst(m: torch.Tensor):
    n_kp = m.size()[0]
    # As we are actually re-using the soft-arg-max operator, we must
    first normalise and negate our input matrix
    m = torch.triu(1 - (m / m.max()), diagonal=1)
    # Select an initial node as Primm's algorithm states, and as we do
    not allow for self loops, remove index (0, 0)
    rows = torch.ones((n_kp))
    rows[0] = 0.0
    cols = torch.zeros((n_kp))
    cols[0] = 1.0
    # Initialise our Minimum Spanning Tree here
    mst = torch.zeros((n_kp, n_kp))
    # A tree has n-1 edges for graph with n nodes
    for i in range(n_kp-1):
        # Creating a mask from the currently selected nodes
        mask = cols.view(n_kp, 1) * rows.view(1, n_kp)
        # Apply that mask to the data to hide unselected edges
        masked_m =  mask * m
        # Using the soft arg max algorithm with a very low temperature
    parameter, approximate the index containing the maximum value
        smax = ((soft_arg_max(masked_m) + 1) / 2) * (n_kp - 1)
        # The next line implements the gradient passthrough trick to work
     around the not differentiable rounding operator
        inds = ((torch.round(smax) - smax).detach() + smax)
        # Select this index in our Minimum Spanning Tree
        inds = inds.long().squeeze(0)
        mst[inds[1], inds[0]] = 1.0
        # Disable the selected row to prevent loops
```

```
28          rows[inds[0]] = 0.0
29          # Enable the selected column to search for neighbours of the
    newly selected node
30          cols[inds[0]] = 1.0
31      return mst
```

Listing 5.1: Differentiable Minimum Spanning Tree Implementation using PyTorch

5.3.2 Demonstration in Two-Dimensions

To understand how we can use this prior to locate a suitable three dimensional articulation model, we can see clearly how this works in two dimensions and the extension is then conceptually trivial.

We have created a simple dataset of keypoints taken from the articulation points of a 2D stick figure to demonstrate this algorithm, as shown in Figure 5.3. Using this, we would like to determine a connection matrix to see which points are connected to learn our articulation model. The first step here is to calculate a pairwise distance matrix between every pair of keypoints. From this symmetric matrix we can see how far each point is from every other point, but this is not enough on its own to calculate connections. Next we need to calculate our variance matrix by taking a batch of these pairwise distance matrices, and calculating variance over these values. Assuming (1) that there is enough natural variance in the keypoints and (2) that we have a large enough sample size, then we will see a variance matrix that tends to 0 in positions where two joints are connected by a rigid bone. To determine connectivity now is trivial, and we calculate this using a minimum spanning tree (MST) over the values in the upper triangle of this variance matrix. The resultant tree from the MST algorithm now represents our connection matrix.

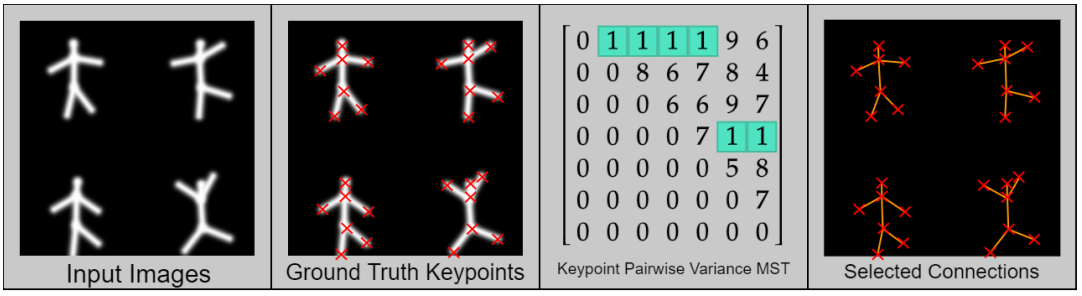


Figure 5.3: A diagram demonstrating the steps used in the LVMST approach for locating a connection matrix from a simple 2D dataset.

5.3.3 Extending to Three-Dimensions

Intuitively this extends into three dimensions as once we have derived the pairwise distances, the rest of this approach remains identical in 3D. The power of this method is now that we can take two-dimensional keypoints and predict the third dimension (the z-axis co-ordinate) to fulfil our rigid three-dimensional bone requirement in order to learn three dimensional articulation models from two-dimensional inputs without the need for ground-truth 3D keypoints. The next section will use this principle and apply it to the pose lifting problem as initially discussed in Chapter 3.

5.4 Pose Lifting with Rigid Bones Prior

With this prior, we can revisit the problem of lifting 2D poses into the third dimension as previously discussed in Chapter 3. We have briefly discussed in Section 5.3.3, how we can determine a connection matrix from 3D keypoints, but what if we only have 2D keypoints that represent a 3D model and we want to estimate the corresponding z co-ordinates?

As a simpler solution to depth inference, which does not require adversarial learning, we look to apply our rigid bones prior. When paired with a simple self-consistency loss, we are able to locate consistent 3D poses from a 2D dataset. This is in contrast to the weak 3D prior used in the original adversarial approach in Section 3.4, which is able to lift 2D poses into the third dimension, but as discussed previously, can be problematic.

5.4.1 Minimising Limb Length Variance to Estimate Depth

Given 2D keypoints, how can we use the rigid bones prior to estimate the keypoint depth? We know that lengths of limbs in three dimensions should be constant over a batch, so if we know two keypoints are connected by a rigid limb and the distance between them appears to be less than the limb length as observed in other examples, then it must either be going backwards into the image space plane or forwards out of it. We can develop a loss function to give a training signal to our depth estimation network using this knowledge, which simply minimises the variance of the three dimensional pairwise distances for pairs of connected keypoints as located by the LVMST connection matrix. By doing so, we should see that all pairwise distances tend towards being constant in three dimensions, as our network learns to estimate z co-ordinates that minimise the distance to the true keypoint depth.

However, this loss function alone does not allow us to estimate depth consistently yet, the estimated z co-ordinates can be either positive or negative relative to its connected

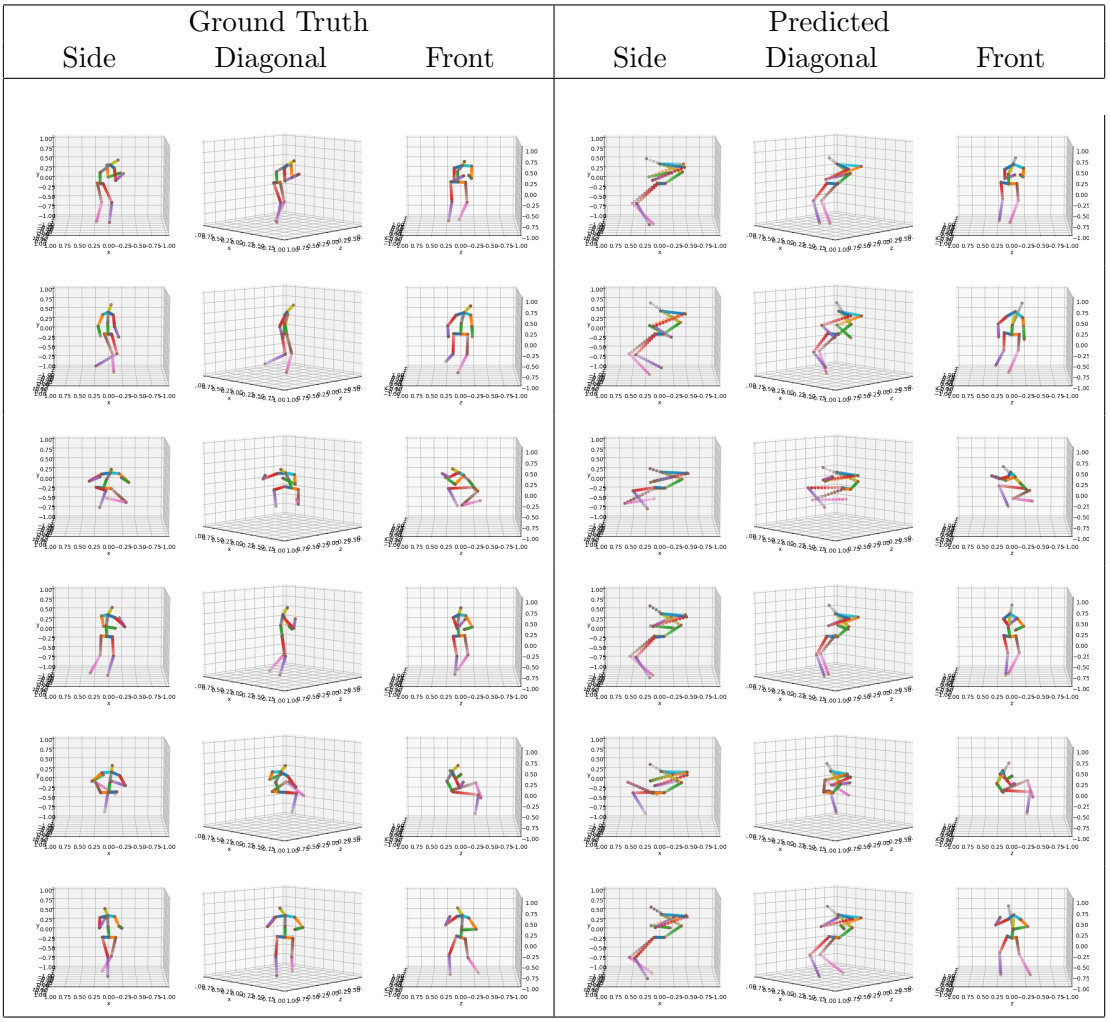


Table 5.1: Demonstration of using our rigid bones prior for keypoint lifting, without the use of a self consistency loss.

joint and still achieve a valid solution. We can see the results of using only this LVMST based loss function when training in Table 5.1.

5.4.2 Self Consistency in the X and Z Dimensions

To fix our initial issues found using the LVMST with our variance loss, we have introduced a self-consistency check as an additional loss function. Depth estimation network takes K_{xy} and generates K_z, fundamentally training the network to infer a side-on view of a shape. If we then pass K_{zy} into the network, and we have learnt a consistent depth then we can expect our output to approximate $-K_x$. We form this into a simple mean squared error loss function between net(K_{zy}) and $-K_x$.

We can visualise that our network is learning to rotate a structure by 90 degrees in either direction, and depending on the direction that it has learnt to rotate, our 3D structures are either learnt correctly or are learnt as the inverted pose, depending on initial network

Layer Type	Input Size	Output Size	Activation
Linear	$2k$	256	ReLU
Linear	256	128	ReLU
Linear	128	64	ReLU
Linear	64	k	-

Table 5.2: A table of the parameters of each layer of the pose lifting network, specifying their layer type, input size, output size, kernel and activation. For the experiments in this chapter, we use $k = 16$, using the simplified keypoints from Human3.6m, and the output of the network is the predicted z co-ordinates.

parameters. In either case, we achieve consistency over the batch of outputs, avoiding the occasional inverted pose problem seen when learning to infer depth using adversarial learning.

5.4.3 Implementation

We use a simple linear neural network, with input size of $2K$, giving an output of size k, which we provide our x, y data and get an estimate of z' as an output. Using $K_{xyz'}$, we can now predict a connection matrix using the LVMST algorithm and formulate our variance loss to minimise sum of the dot product of our batchwise keypoint distance variances and the connection matrix. We then apply the self-consistency loss as described previously and use a sum of the two loss functions as our final loss.

The network architecture used for our pose lifting network in these experiments is a simple sequential MLP with parameters as defined by Table 5.2. To train the network, we train for 250 epochs, using a batch size of 256 and a learning rate of 1e-5.

5.4.4 Experiments

As this self-consistency check is much simpler than the adversarial technique for inferring 3D as discussed in 3.4, our proof of concept experiments show that this method can be used to quickly and reliably estimate 3D poses from 2D inputs, without the difficulty of balancing a generator and discriminator.

We use the same experimental set up as in Section 3.4, using the ground truth 2D keypoint data from the Human3.6m dataset[48] as inputs and estimating the corresponding 3D points.

We show our quantitative results in Table 5.3. We show a large improvement in mm error when compared to our previous approach and comparative accuracy with respect to similar self-supervised and semi-supervised approaches.

Approach	Mean accuracy (mm)
Martinez et al. [79] (supervised)	45.5
Drover et al. [27] (weakly supervised)	64.6
Kudo et al. [61] (self-supervised)	130.9
Chen et al. [12] (self-supervised)	51.0
Ours (Chapter 1 - self-supervised)	155.5
Ours (bone rigidity prior - self-supervised	108.8

Table 5.3: Mean distance between predicted and ground truth poses in the human3.6m dataset.

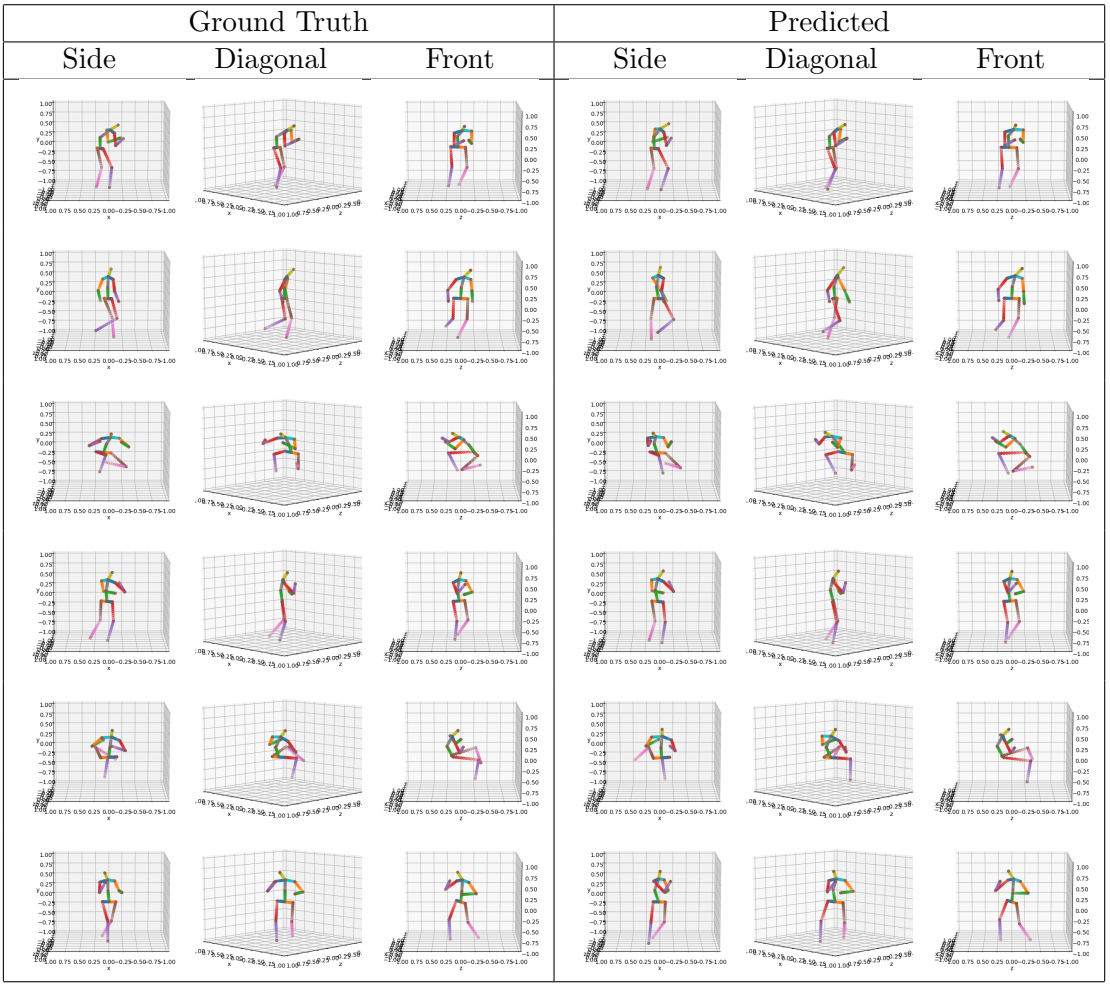


Table 5.4: Demonstration of using our rigid bones prior for keypoint lifting, using our self consistency loss to fix the issues demonstrated in 5.1. More examples can be seen in Appendix C.

Qualitatively, we see that our poses are reasonable and represent the correct pose in most cases, as can be seen in Table 5.4. But we do have some errors, notably the leg points sometimes being at angles that are not feasible.

5.4.5 Discussion

As we have previously discussed, our initial adversarial based pose lifting approach covered in Section 3.4 could lift poses into 3D, but not without some issues. If instead, we impose our rigid bone prior, we have enough information to determine consistent poses, avoiding the occasional issue of inverted poses, while achieving more accurate results. In addition, this approach locates which keypoints are connected, using that information to predict the depth of each point, and this richer representation could help us in the future to predict through occlusion of provide keypoint uncertainty information.

Another major advantage we see is the ease of training and adjusting hyper-parameters in the self-consistency method. Adversarial learning comes with challenges due to the requirement for balancing the generator and the discriminator networks, but being able to leverage the depth information using self-consistency omits this requirement. This model also required far fewer parameters, 50640 compared to the 224962 used in the keypoint lifting network of the adversarial approach. Additionally, we would expect this approach to be more robust to any other changes that would usually inhibit the results found using an adversarial method.

To answer RQ4, it is difficult to say from this experiment if this is definitively the minimum prior required to solve this problem, but as this prior makes no assumptions that prevents our approach from working on pose datasets for any entity built of rigid bones, we are satisfied that we have chosen an appropriate prior for representing generalised articulation models.

We must also address some potential concerns using this approach. In this case, with 2D keypoints as an input, the z co-ordinates are estimated to match the maximum apparent length of each limb, so we must make the assumption that the dataset contains examples where the limb is perpendicular to the direction of the camera. Without this, the estimated depths will only be selected to match the maximum visible lengths of limbs which could possibly underestimate limb lengths. But as we expect our datasets to be large and diverse enough, this ought to never be a significant issue.

But this section has been based around the assumption that keypoints can be located on points of articulation and ends of limbs as an approach for pose lifting. The next section will look at how we can learn such points by using our rigid bones prior.

5.5 Encouraging Keypoints to Locate Joints

In the previous section we have demonstrated the power of this prior for self supervised pose lifting. But this prior is not limited to applications within the depth estimation step of our articulation model pipeline, is also applicable to the keypoint detection stage.

As previously mentioned, for our bone rigidity constraint to hold, our keypoints must capture only joints between rigid limbs in order to exploit their fixed length as a prior. In our previous self-supervised multi-task learning approach, as described in Chapter 4, there were no objectives to encourage this behaviour. As we have generalised keypoint detection to look for keypoints in any structure, there is no reason why it should locate limbs over another arbitrary point that encodes similar information. To locate the ends of limbs, a new objective function is required. We will take inspiration from our previous reconstruction task and place a greater constraint on the reconstructions produced.

Readdressing RQ3, we are now switching the context of the term keypoint from being a generic spatial representation as was used in Chapter 4, to representing landmarks on an articulated object. The ideal landmarks are those that locate points of articulation, and endpoints of limbs, such as hands and feet. However, as we are still restricted to self-supervised learning, the ordering of these keypoints will be arbitrary, as there is no implicit ordering contained within the data that informs which keypoint should represent each joint. The only criteria is internal consistency, so poses from our model contain joints that align, which can be difficult, especially with the natural symmetry found in many articulates objects. This will be discussed later in Section 5.6.3.

5.5.1 Differentiable Sketching

Previously, we have been using a standard convolutional neural network for our reconstructor, but the outputs from this network are unconstrained. For some applications this is advantageous as it allows for fine tuned learning to create highly detailed outputs, as is the case for a GAN[35]. But for this application we do not require high detailed reconstructions, only a good captured representation. For this reason, we believe replacing the reconstruction network with a constrained image renderer will aid in this. A standard convolutional reconstructor network can learn to create an image from a set of keypoints with some flexibility, whereas a differentiable renderer has no way of inferring data that does not exist in the keypoint structure. This means that reducing the capability of our reconstruction network forces our keypoint detection network to locate stronger keypoints to reduce our reconstruction loss.

Our main motivation for reconstructing using a renderer is to encourage keypoints to be placed on the endpoints of limbs. For this reason, the solution we use is a simple sketching renderer that sketches straight lines between pairs of points. Differentiable Sketching [80] allows us to put such a substantial bottleneck on the ability of our reconstruction based losses. As the renderer works by drawing lines between keypoints, for a pixel to be drawn on the output image, it must be between a pair of keypoints and the connection matrix signify that the pair of points are connected. Intuitively, this means that to have good reconstructions, we must have keypoints selected at the end of a line and in the case of an articulated structure, this means it must be a joint or a limb endpoint.

For a wider range of applications, Li et al. [66] have developed an approach that uses a vector graphics based renderer, while maintaining differentiability. For use as a downstream renderer that uses keypoints as an input, this would allow for greater accuracy in recreating input images than a basic sketch.

Differentiable sketching builds upon our designed prior, following simple rules of keypoints that are connected only by straight lines. It allows for sketching images based on a set of keypoints and a connection matrix, which can be derived as demonstrated in Section 5.3. Combining and mapping these two elements back into image space, using differentiable operations, gives us the ideal framework for creating a self-supervised loss function that allows for training our network to locate ideal keypoints for articulation models.

5.5.2 Implementation

For a proof of concept, we have implemented a simple keypoint regressor network to derive keypoints from images and uses differentiable sketching to train the regressor in a self-supervised fashion. We use a simple linear encoder-decoder style network consisting of two linear layers to create an encoded space, and two layers to decode the latent vectors into a list of keypoints.

With these output keypoints, we are able to sketch the image for use with the reconstruction loss. Differentiable sketching is done via a rasterisation step, which creates a raster for every pair of keypoints. This raster calculates the pixel intensities by their distance away from a line drawn between the pair of points. We then create a composite image using these rasters alongside the LVMST derived connection matrix, which selects which rasters to use. These are then composed into the final image by overlapping rasters and capping the pixel intensities at 1.

Training using a LVMST derived connection matrix presents some interesting decisions. We have already established a multi-task keypoint detection approach in Chapter 4, but adding this as an extra task may lead to degenerate solutions where limb variance is minimised to create keypoints stacked on a single location. We instead train without this task for the first portion of the training time, and then add this task once we have started to locate some basic structure from the images.

The network architecture used for the 2D stickman experiments is defined in Table 5.5.2 We trained this model with batch size of 96 and for 30 epochs, without the use of the variance minimisation loss for the first 5 epochs. We balance our losses using the exponential function approach proposed by Liang and Zhang [68] with a manual multiplier of 10 to the reconstruction loss prior to this function.

Layer Type	**Input Size**	**Output Size**	**Activation**
Linear	w^2	512	ReLU
Linear	512	256	ReLU
Linear	256	512	ReLU
Linear	512	$2k$	TanH

Table 5.5: A table of the parameters of each layer of the keypoint detector adapted to locate ends of joints on the 2D stickman data, specifying their layer type, input size, output size, kernel and activation. We set $k = 7$, representing the keypoints used to build the simple toy stickman dataset and $w = 64$, representing the width and height of our input images.

Layer Type	**Input Size**	**Output Size**	**Kernel**	**Activation**
Data viewed as b vectors of size $h * w$				
Linear	$h * w$	$h * w$	-	-
Data viewed as b 3D tensors of size $1, h, w$				
Conv2D	1	16	3x3	LeakyReLU($\alpha = 0.1$)
Conv2D	16	k	3x3	-

Table 5.6: A table of the parameters of each layer of the keypoint detector adapted to locate ends of joints on the Human3.6m stickmen data, specifying their layer type, input size, output size, kernel and activation. We set $k = 16$, representing the keypoints used to build the simple toy stickman dataset and $h, w = 72$, representing the width and height of our input images. Output is passed through soft-arg-max to receive predicted keypoints of size $k, 2$.

Our keypoint detection network architecture for the human3.6m stickmen experiment is as defined in Table 5.6. We trained this model with batch size of 96 and for only 1 epoch, as the dataset contains 1.877e6 images. We balance our losses as in the 2D stickmen example.

5.5.3 Experiments

To initially test this approach for locating superior keypoints for downstream processing with regards to articulation models, we have created some simplified datasets as a proof of concept.

5.5.3.1 Creating a Simple Sketched Dataset

As a proof of concept for applying differentiable sketching to the problem of generalised articulation, we will demonstrate how this approach finds keypoints on simple datasets with articulation. We have created an artificial articulation model dataset in two-dimensions that resembles a stickman with limbs of rigid length, sampled at random angles. The root point at the mid shoulder area is sampled at random and the mid-hip point is set to a fixed distance directly below and head point is placed directly

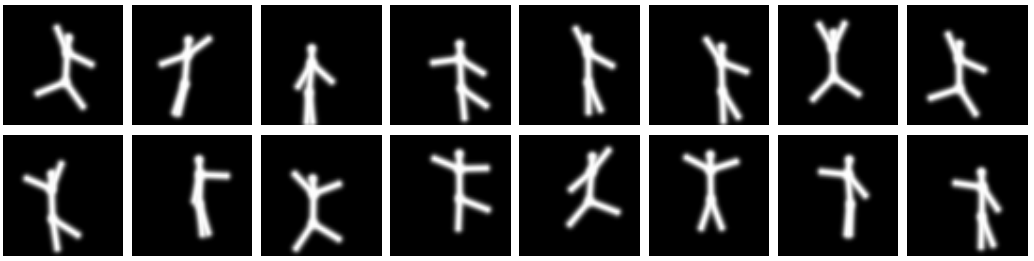


Table 5.7: Examples of images generated in our artificial two-dimensional articulation dataset.

above. Arms and legs are randomly sampled within a limited range of acceptable angles. Examples of images from this dataset are shown in Table 5.7.

5.5.3.2 Proof of Concept: Artificial Two-Dimensional Stick Figures

In the simplest case, we are looking to learn keypoints on an image and learn the underlying articulation model. As a proof of concept we have used our artificial dataset as shown in Table 5.7, that limits the articulation points to rotate only in two dimensions. Our network must learn where to place keypoints on the image to create a connection matrix via the LVMST method.

As we have created this dataset artificially, we have ground truth labels, and we can use these labels to quantitatively measure the distance between our predicted keypoints and the ground truth points. But as the ordering of our keypoints is unrestricted, before measuring distances we must use the Hungarian algorithm [62], to align the indices that contain keypoints in the ground truth and the predicted cases. This is applied to the predicted points and done on a batch-wise basis to enforce keypoint consistency, doing on a per-example basis would artificially inflate our result if any of our estimated keypoints were inconsistent. A numerical comparison of these keypoints is found in Table 5.8, showing the mean L2 distance between predicted and ground truth keypoints. Additionally, a qualitative analysis is shown in Table 5.9, to visually see where keypoints are placed in either case.

We note, both qualitatively and qualitatively that the LVMST approach with differentiable sketching is superior when our goal is locating keypoints that sit on articulation points or the ends of limbs, even if the multi-task approach is equally consistent. We believe that this is due to a greater restriction on the downstream task used for training, which cannot achieve low reconstruction losses unless a keypoint is placed on limb ends and articulation points. Our generalised keypoint detector has no such restriction as this would not maintain generalisability, as was a key aim of this approach.

	Multi-Task Keypoint Detector	LVMST Keypoint Detector
Centre Shoulder	0.2048	0.0003
Centre Hip	0.1993	0.0003
Head	0.2102	0.0006
Left Arm	0.2741	0.0005
Right Arm	0.2229	0.0024
Left Leg	0.7425	0.0023
Right Leg	0.4418	0.0024
Mean	0.2651	0.0006

Table 5.8: Comparison of mean Mean Squared Errors between Ground Truth and predicted landmarks, on a per landmark basis, from the artificial 2D stickman dataset.

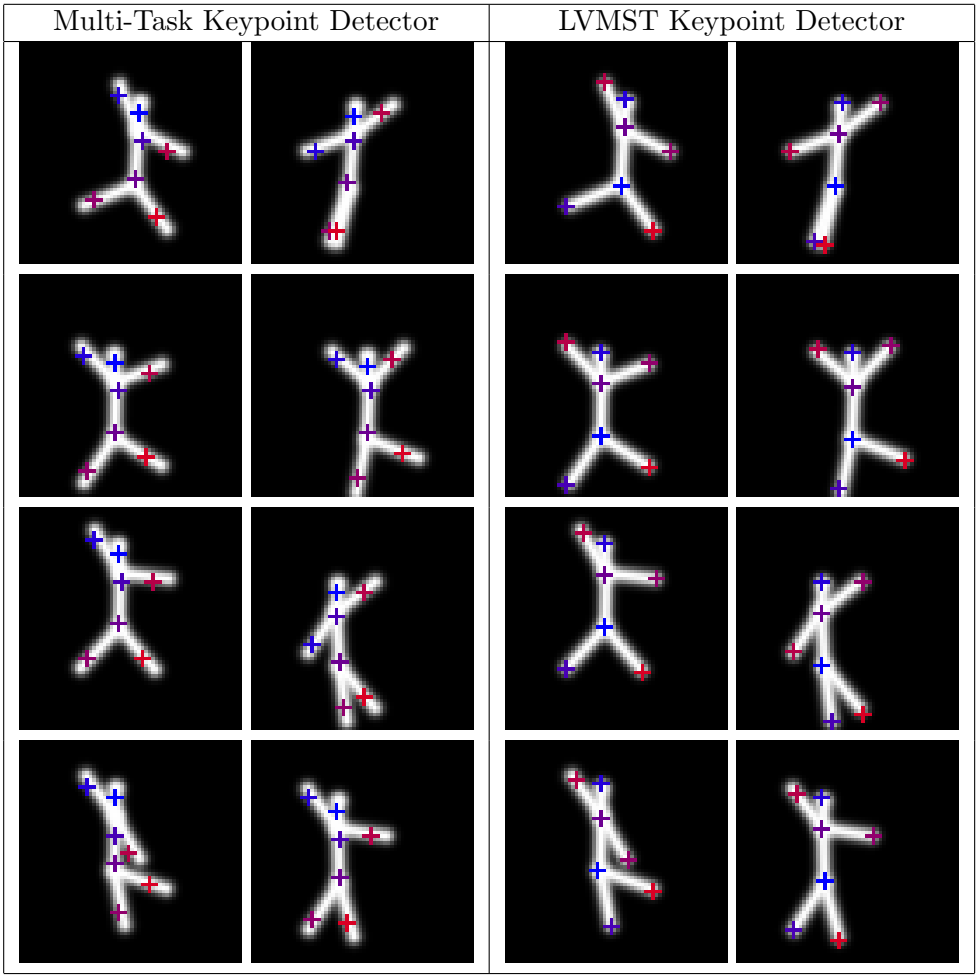


Table 5.9: Comparison of placement of keypoints between Multi-Task generalised keypoint detector and LVMST with Differentiable Sketching on the artificial 2D stickman dataset.

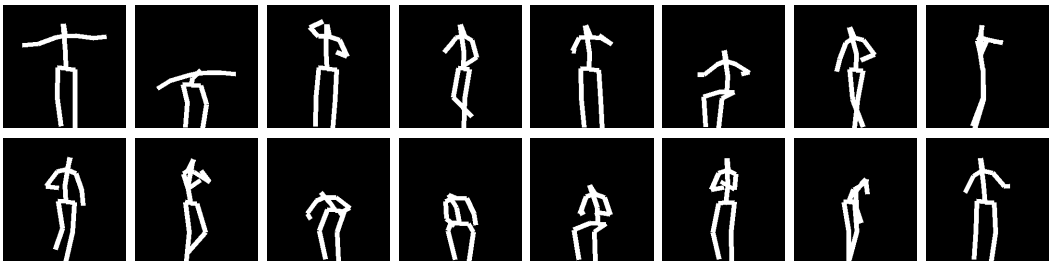


Table 5.10: Examples of images generated in our artificial three-dimensional articulation dataset, using ground truth points from the Human3.6m dataset.

This result will be advantageous when we look to apply this keypoint detector to harder examples where the articulation and end points are not as obvious, and will allow for better estimation of depth, as discussed previously in Section 5.4.

5.5.3.3 Toy Dataset: Human3.6m Stick Figures

We have shown the previous section how we can solve this task when dealing with two-dimensional articulated objects. The next step is to extend the previous solution into the third dimension in order to infer the depth of a structure. Can we locate keypoints on shapes that are not just two-dimensional and will thus likely contain self-occlusion?

We create a dataset in a similar format to the 2d stickman dataset as shown in Table 5.7, but with realistic poses taken from the 3d keypoints contained within the Human3.6m dataset[48]. Examples of images contained within this dataset are shown in Table 5.10. It is also worth noting that when we use these images for an input to our model, we apply a Gaussian blur onto them to help create a smooth gradient through our network.

We demonstrate the results of this experiment in Table 5.11. We see that we are now able to place keypoints near the ends of limbs as desired. However, finding consistent keypoints in this more complex case seems to be more of a challenge. This may be attributed to a couple of reasons. Firstly, the scale of the features that we are looking for is much smaller, especially in the case of the hip and shoulder keypoints where only a few pixels separate those points in the best case scenario. In a worst case scenario, these points may be overlapping in the image-space and thus our model must allow for this.

Secondly, self-occlusion makes finding consistent keypoints in highly dynamic areas such as the arms, especially as we are using self-supervised learning and the model is learning entirely from the image data.

Because of this lack of consistency, it is difficult to evaluate these results quantitatively, as distances from ground truth points are difficult to calculate when the Hungarian algorithm will produce highly variable alignments for each example. While one approach

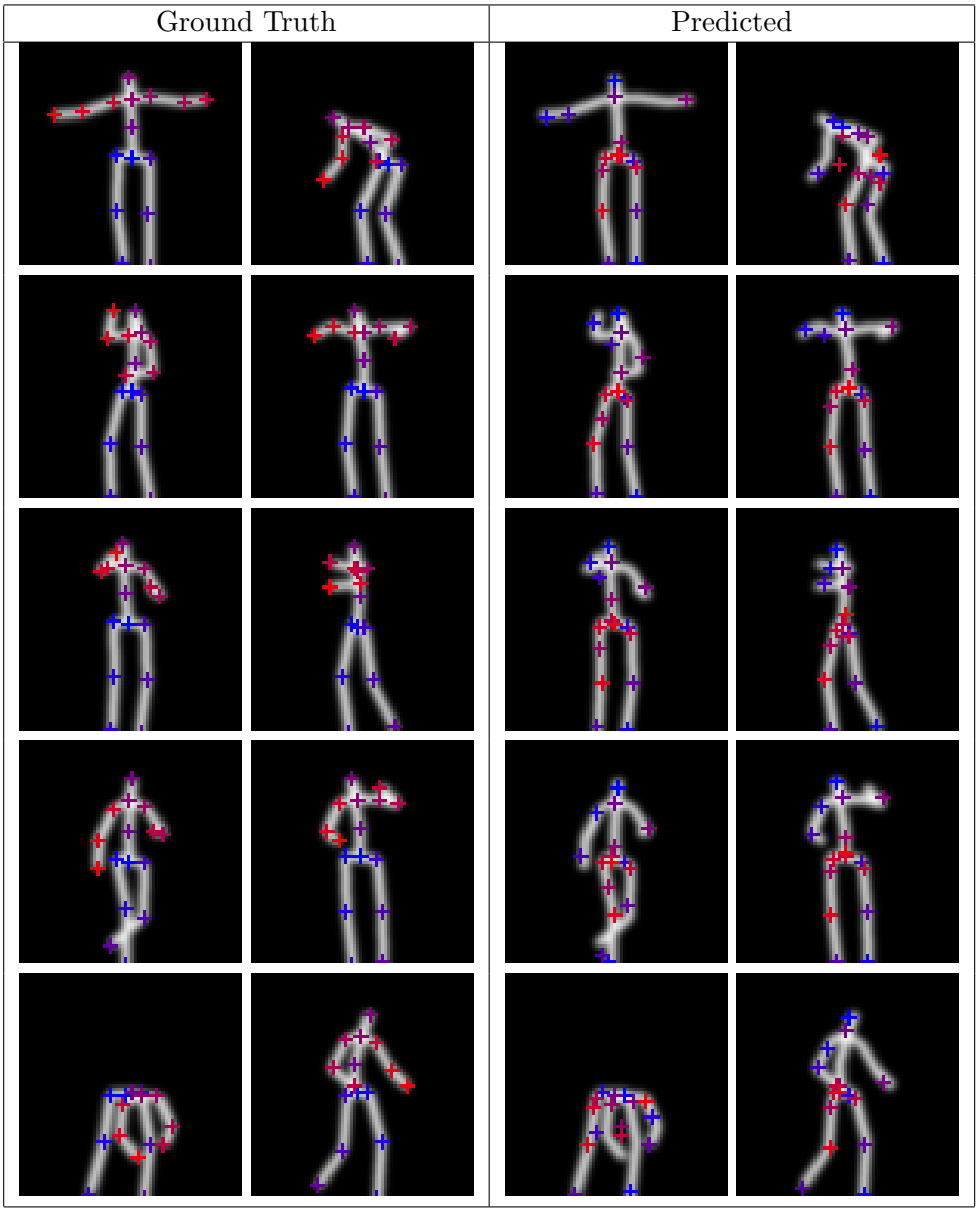


Table 5.11: Comparison of placement of keypoints between in the ground truth Human3.6m stickmen dataset and the differentiable sketching approach at placing keypoints.

could be to assume the nearest point is the best match and allowing for duplicate assignments, we run into issues around the arm areas in this case, as the distribution of keypoints are sparse in the upper half of the pose.

5.5.4 Discussion

In this section we have shown how we can use our rigid bones prior along with differentiable sketching to encourage our keypoint detector model to learn to select points

near areas of articulation and limb end points. This is desirable for our needs in determining articulation models, but we have only demonstrated success on toy datasets. This simplifies the task as it is much easier to sketch the output to match the format of the input images, meaning a simple mean squared error between the sketch and input images provides a sufficient training gradient. The difficulty will now be in translating this approach onto real world images.

One way could be by taking inspiration from Mihai and Hare [81], where sketching is used to communicate information from images between a pair of neural networks. Instead of a mean squared error loss based on the input images, we could make use of a referential game, where a differentiable renderer is used to communicate articulation between two networks. By restricting the data to examples that contain the same subject in different poses, our network should find an optimal solution by sketching an articulated object, by drawing a stick figure that resembles the object in order for another network to solve the referential game.

This approach may require an extra loss function to prevent the network from using the sketch like a hashing function, where the sketching network will output unique shapes for each image that do not follow the structure of the articulation model but still provide sufficient information to uniquely identify each pose. A structure-wise loss may be required, such as a Chamfer distance loss between a foreground/background segmentation and the sketch used in communication. Another solution would be to solve a similar game based on simple affine transforms of the same input image. This would prevent the network from communicating appearance information over structural information as the only change in the image will be structural between transforms.

5.6 Discussion

This chapter has introduced a simple yet powerful prior and we have subsequently demonstrated its usefulness in both stages of the self-supervised 3D pose estimation pipeline. But as the problem we are aiming to solve is very complex, this prior is not flawless in solving every problem in this space. This section will discuss issues with the prior and implementations that leverage the prior, and discuss other possible priors that could also be used to solve this difficult problem.

5.6.1 Problems with Limb Variance Minimum Spanning Tree

LVMST is a simple and elegant way of deriving the connectivity of an articulated model but it is not without its limitations. Using the absolute distances quickly breaks down when the scale of our subject can vary due to distance between camera and subject. Normalisation of limb lengths is the obvious solution, but with a number of unknown

parameters, of both the camera and subject, there is no obvious way of normalising the data without more information. We also acknowledge that at inference time, we require a batch to derive a connectivity, which means if this were a real-time application, we would require either a rolling batch of video frames, thus a short time delay before we can compute an articulated model while frames are batched, or a way of circumventing the batch requirement. One such way would be to train a simple network to imitate the output of the LVMST algorithm given a set of keypoints, assuming there is enough information in those points to make the inference.

Another issue we see with this algorithm is during the training stage. If the LVMST algorithm selects the wrong connection matrix, then that will be enforced later in training, as our loss that minimises variance makes the chosen connection matrix more likely to appear at the next training step.

A necessary improvement for the future is a training scheme that allows for greater flexibility for discovery of new connection matrices. One option could be train using only the other losses first, and introduce our minimising variance loss at a later stage. This would scope the dataset for suitable keypoints for solving the other tasks, with the hope that they capture some structure in the process. After a portion of training time, we would then introduce the loss that minimises variance and attempts to move the estimated keypoints onto suitable articulation locations.

5.6.2 Robustness to Errors in Training Data

We have demonstrated the ability to learn a connected articulation model when given high quality data, but in a scenario where high quality data is not available, modifications still need to be made. One such approach to add robustness when confronted with imperfect training data, is to provide each batch with a pre-processing step once keypoints have been detected from each image. This pre-processing could include outlier detection to remove any shapes that sit far outside of the expected range, and only deriving an articulation model from the remaining data. As we also require robustness in our system, which in deep learning is often provided with difficult training examples, in this case extreme poses, so careful tweaking of an outlier detection algorithm would also be required to maintain the robustness derived from difficult dataset examples.

5.6.3 Consistent Keypoints in Symmetric Models

Symmetry is a recurring problem in this space, as a lot of articulated objects that we are interested in are naturally symmetrical and determining if we are looking at the front or the back of our subject is not trivial. We observe symmetry issues in keypoint detection where our network will place keypoints in the same regions of the image, and

not flip left with right when we are looking at the back instead of the front. We briefly discussed this issue in Section 4.7.1.3. We also see issues in keypoint lifting, where depending on the network initialisation, we derive either all inverted poses or all correct poses. However, this is easily resolved by rerunning the network training with new initialisation parameters, or if we were happy to lose some generalisation, then a joint angle prior similar to that used by Kudo et al. [61] would resolve this issue.

5.6.4 Other Possible Priors

In this chapter we selected bone rigidity as our prior to assist the self-supervised model in learning articulation, but this is not the only prior that we considered. Careful selection of a self-supervised prior is required as too strong a prior would remove the generalisation advantages of using self-supervised learning, or may require information that we cannot take for granted in all articulated datasets. We must also think about edge cases where our prior knowledge may break down and lead to erroneous results.

5.6.4.1 Centre of Mass Estimation

A potentially simple and powerful prior would be to estimate a centre of gravity to make sure that our pose obeys an estimated physics model. This would assist in the depth inferring step as our z co-ordinates would have to be balanced when combined with our x and z co-ordinates, eliminating a range of poses that would no longer be possible. One issue with this is knowing the estimated mass of the real world parts that our keypoints represent. Estimating the weights of each articulated component purely from the image data with enough accuracy would be difficult in this approach, but number of surrounding pixels in the foreground object may provide a rough heuristic for mass. Some more fringe cases also exist that will skew the success of such a prior such as when the subject is sitting or leaning. Other cases such as being mid-fall or in low gravity may occur but certainly outside of the expected range of poses that we would expect to see.

5.6.4.2 Limiting Joint Angles

Intuitively we can see that joints in an articulated model can only bend so far before they have hit a limit and cannot bend any further without damage. This is a prior that could help remove erroneous examples, and provide additional information for estimating the connection matrix by assuming consistent range of angles between pairs of connected points. While Kudo et al. [61] and Raaj [90] both use a similar prior to add consistency to their approaches, both rely on knowing a real world model of joint angles. A proposed tweak to this prior, which makes the assumption that there is a range of valid angles

for each joint, should not remove generalisability and could perform outlier detection as a self-checking tool. By learning the distribution of each articulation point purely by observation in the data, it would be possible to apply expected these expected values to add consistency to the keypoint prediction. However, the implementation might require careful balancing to ensure it gives the model enough freedom to learn the correct distribution of points from the data before it restricts estimation of keypoints to these observed distributions.

5.6.4.3 Symmetry

It is common for articulated models, especially those in nature, to contain symmetry which is something we can potentially leverage to assist in pose estimation. Wu et al. [109] do this by assuming the image contains a 3D deformable object that is also symmetrical, and take an auto-encoder and renderer based approach to estimate the depth of human faces, cat faces and cars. A symmetrical approach would help in our articulated model case too, and error checking via left and right side correspondence, such as checking that limb lengths were consistent across the symmetrical plane, would help to add robustness in cases that estimated keypoints were erroneous. However this could lack the generalisability that we are looking for in our model, as some articulated models, such as robotic arms, do not contain symmetry that would allow for this approach to work.

5.7 Conclusion

This Chapter has introduced a prior that can be used to assist in finding self-supervised articulation models. Determining the correct prior is aiming to answer RQ4, and finding the balance between a very applicable prior and a very powerful prior is essential for locating generalised articulation models. We believe that this prior contains enough information to assist in locating articulation models without making too strong assumptions to inhibit the generalisability of this approach.

Chapter 6

Towards Self-Supervised Learning of 3D Articulation

This chapter aims to combine the work done in Chapters 3 and 4 into one pipeline that is able to convert a 2D image of an articulated object into a list of 3D keypoints. As discussed previously, this problem is ill-posed. Additional prior knowledge is required to solve this, thus we look to implement the bone rigidity prior from Chapter 5, to aid in determining the underlying articulation model. This chapter also aims to answer RQ1, as we pull together the knowledge learnt from all of the previous chapters into one pipeline that attempts to go from an input of an image to an output of a list of 3D keypoints.

6.1 Introduction

Self-supervised 3D pose estimation is a difficult task, thus it is unlikely that training a network without the use sufficient prior information will yield good results, and we demonstrate this in Sections 6.3 and 6.4. The use of a prior will firstly aim to improve our keypoint detector to place our keypoints on articulation points as demonstrated previously in Section 5.5. Articulation points contain more information about our structure than arbitrary points as found via a generic keypoint detector as previously discussed in Chapter 4, and thus contain more information for solving downstream tasks, especially when considering depth estimation. We have demonstrated previously, in Section 5.4, how we can exploit bone rigidity to infer depth in the articulated model when given ground truth 2D keypoints, but can we estimate keypoints to a close enough extent to achieve the same depth inference?

Extracting 3D information from a 2D image in a self-supervised way traditionally requires strong prior knowledge to know how to extract 3D information from a 2D representation of the world, commonly done using multiple viewpoints [67, 106, 4] to infer depth from information found in the input.

The following sections will investigate what happens when we try to learn 3D points with no prior, using a naïve method that combines the work of Chapters 3 and 4. We show that there is more information required than simply encouraging keypoints to find articulation as demonstrated previously in Section 5.5, with a simple approach combining differentiable sketching with adversarial keypoint lifting. We then explore how we can introduce our carefully designed bone rigidity prior in an attempt to determine 3D articulation. We finish this chapter by discussing the obstacles that still need to be overcome to complete a self-supervised 3D articulation model approach using a bone rigidity prior to achieve reliable results.

6.2 Related Work

A common approach to self-supervised 3D keypoint estimation relies on multi-view geometry. Using multiple views of unlabelled images, Wandt et al. [106] train a self-supervised monocular 3D pose estimator using multi-view consistency constraints to disentangle the 3D pose and the camera parameters. Their novel contribution in this space is being able to work without calibrated cameras, and thus allows for non-static cameras that are pointed at the same person. They also make use of an off-the-shelf 2D pose estimator for the initial stage of the 3D pose estimation pipeline, which is something that we are aiming to avoid, attempting to learn a 2D keypoint detector using purely self-supervised techniques.

Another approach by Li et al. [67] looks into applying geometric knowledge to solve the self-supervised 3D pose estimation problem. To reduce errors from self-occlusion, they use confidence values for each 2D keypoint and integrate losses from different viewpoints as an effective way of creating a 3D structure when some part of the structure cannot be seen from all angles. In a similar fashion to Wandt et al. [106], they also use a state-of-the-art 2D pose estimator at the first stage of their pipeline, breaking the strict self-supervised restriction that we have imposed for this thesis.

In a similar multi-view environment to the two approaches discussed above, Bouazizi et al. [4] use self-supervised learning to estimate 3D human pose with a high degree of accuracy without the use of a pre-trained 2D pose estimator. Their algorithm triangulates the 2D pose estimates from each viewpoint, while additionally implementing a re-projection loss and imposing geometric constraints, looking for a consistent 3D model across every viewpoint. At time of publication, their approach was state of the art for purely self-supervised approaches to 3D pose estimation on the Human3.6m dataset.

In a 3D hand-pose specific implementation, Wan et al. [105] aim to train a pose estimator through a fitting function using a dataset of hand depth maps. Their approach uses a differentiable renderer to approximate the surface of hand found in the depth map, and by leveraging a simple reconstruction loss to train in a self-supervised fashion. Their approach makes use of a heavy prior in the form of a hand model that is fit to each depth map, which is a hand crafted model built of 41 spheres that estimates the shape of a human hand. Naturally this lacks the generalisability to any articulated object that we desire in this thesis.

Unlike the majority of these approaches, our aim is to remove the requirement for multiple views of the same object, any semi-supervised learning, or using a pre-trained keypoint detector. Our approach should only use a monocular dataset and our generic articulation prior as previously introduced in Chapter 5.

6.3 Naïve Combination of Previous Approaches

In this section, we will demonstrate how learning self-supervised 3D articulation models from images is ill-posed when we do not use a strong enough prior. This approach for an end-to-end pipeline is to leverage the multi-task framework, introduced in Chapter 4, to learn 2D keypoints, before introducing an additional keypoint lifting task, that is trained via our adversarial loss from the first section, to infer the depth of each point. We have previously demonstrated that structure can be derived from images with a multi-task trained keypoint detector, and that adversarial learning can lift a two-dimensional pose into 3D, but can both these steps be trained simultaneously?

6.3.1 Motivation

The idea behind this approach is simple as both sections are established in Chapters 3 and 4, so logically training both sections end-to-end would give a simple solution to this difficult problem. While it was unrealistic to assume that this approach would be successful, experimenting in the simplest case and observing where the failures occur gives valuable insight into the nature of the problem at hand.

6.3.2 Implementation

A quick implementation of this, adapted from the implementation discussed in Section 4.5, quickly shows the issues with trying to solve the problem in this way.

Figure 6.1 outlines this approach, similar to the layout for capturing spatial representations in Chapter 4, but with an additional pose lifting task, which feeds into the balanced

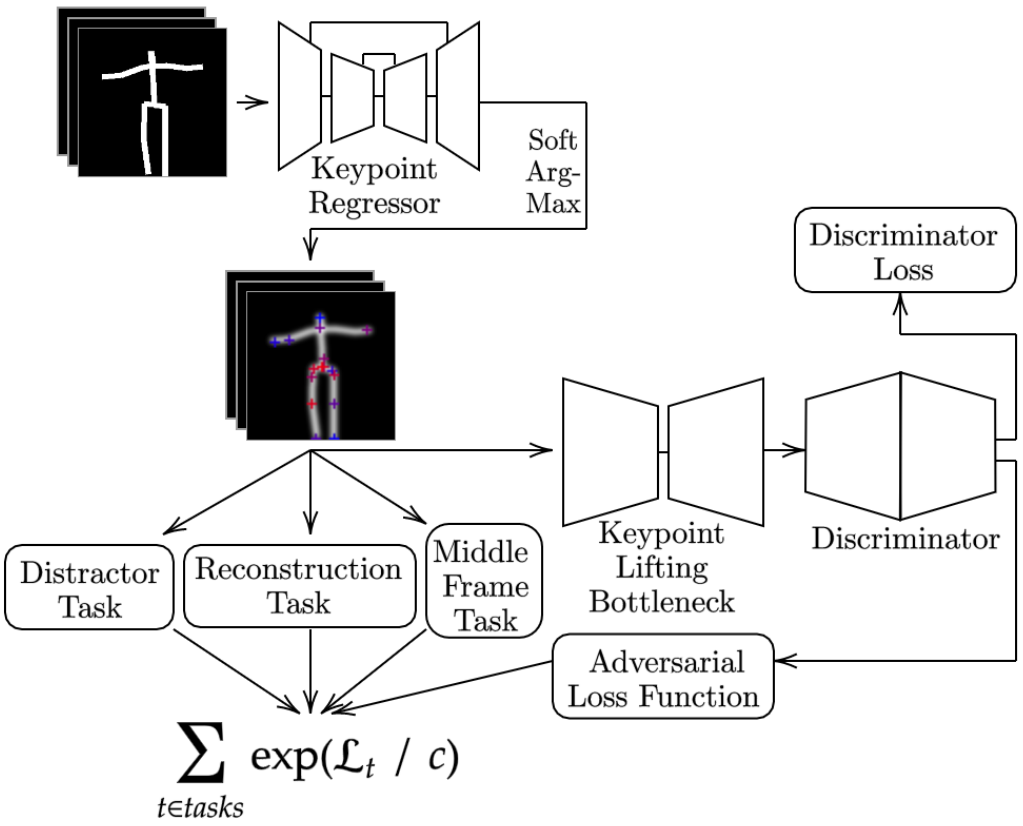


Figure 6.1: Diagram outlining this naïve approach to a full 3D articulation model estimation pipeline. The losses from the three keypoint estimation training tasks are summed with the adversarial loss function (typically known as the generator loss function), with a separate loss to train the discriminator network.

multi-task learning loss balancing function [68]. The discriminator is also trained in the same way as in Chapter 3.

6.3.3 Experiments

In Table 6.1, we can clearly see that our 2D keypoints capture the two-dimensional structure, but the learnt 3D structures are mostly degenerate solutions or ambiguous shapes with no discernible structure. The experiments using this technique show us quickly that this approach will not work. We hypothesise that the reasoning for the inability to learn 3D information is that our 2D keypoints found from self-supervised learning do not contain enough of the fine nuanced details required to infer the 3D structure.

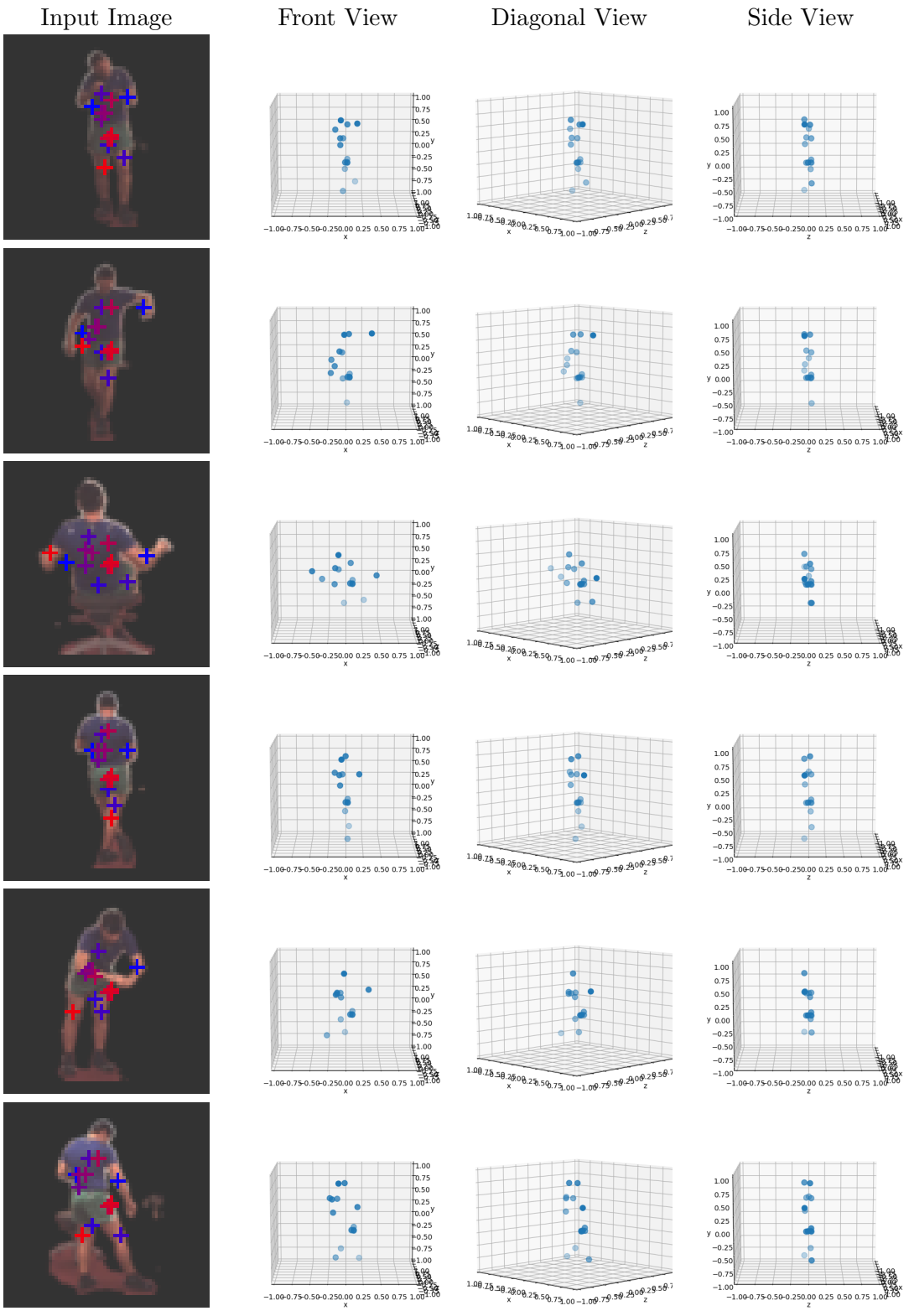


Table 6.1: Results when attempting this naïve method on the Human3.6m dataset. While some basic structure has been captured, it is clear to see that keypoints have not located articulation points or ends of limbs, and keypoints fail to adequately cover the entire subject in each image.

6.3.4 Discussion and Analysis

A key part of this failure is that the keypoints found from this self-supervised multi-task approach do not always sit on points of articulation as they are not required to by any of our objective functions in order to minimise our overall losses. Without locating articulation points, we have little in the way of information required for inferring depth in the image with self-supervised techniques. The inability to learn 3D articulation models without the use of a prior demonstrates the requirement for stronger prior knowledge. We have two solutions to overcome the spatial bottleneck problem, we either create better keypoints that capture articulation with more efficiency, or we pass through image information along with the keypoints to give context to our keypoint, such that depth can be learnt. When considering the power and simplicity of the bone rigidity prior, as introduced in Chapter 5, we naturally would like to infer stronger keypoints. Referring back to RQ3, we notice a semantic gap between the keypoints learnt by our generalised keypoint detector and the keypoints (sometimes called landmarks in this context) that were previously used to infer depth in Chapter 3. This gap is caused by the amount of information that each keypoint posses, even though both are numerically identical, as a list of X, Y co-ordinates. Switching our context of keypoints from arbitrary points capturing spatial information to points that capture articulation, allows us not only to correctly implement our bone rigidity prior, but to specialise into a system that considers the spatial relationships between keypoints in three-dimensions.

6.4 Full Pipeline with Adversarial Learning and Differentiable Sketching

As we have established in Section 5.5 and seen again within Section 6.3, locating keypoints that sit on articulation points or ends of joints is difficult. As before, we are helping our self-supervised keypoint detector to locate the correct points of articulation through a differentiable sketching technique to replace our previous standard reconstruction task. If keypoints are closer to the true articulation points, then our depth estimation using adversarial learning should be able to correctly estimate the z-coordinates, as shown when using ground truth 2D keypoints in Chapter 3.

6.4.1 Motivation

As introduced in Section 5.5, we can use a differentiable sketching based reconstruction loss to place keypoints in desirable locations such as articulation points and ends of limbs within the two-dimensional space. We aim to concurrently learn to estimate articulation points as 2D keypoints, and lift those points into the third dimension using the adversarial keypoint lifting approach described in Section 3.4. While we introduced

the idea of using the Limb Variance Minimum Spanning Tree in Section 5.5, for simplicity in this experiment, we estimate the connection matrix using the keypoint detector network, learning both keypoints and connections through the same latent space vector.

6.4.2 Implementation

Our implementation for the experiments in this section is a fusion of the adversarial keypoint lifting found in Section 3.4 and the keypoint detection approach as discussed in Section 5.5. The output of the latter is fed into the former to estimate 3D keypoints using images as the original input. By training a pipeline to perform these both simultaneously, we hope to be able to locate good 2D keypoints while also estimating their depths.

Our implementation consists of two sections, each with their own optimiser. The first contains a 3D keypoint detector using an auto-encoder to encode each image into a latent space, before predicting the 3D keypoints and a connection matrix which determines if two keypoints should be connected. The second network used in this approach is a discriminator that determines if the viewpoint of the keypoints is as originally found within the data or a side-on viewpoint using the estimated depth.

We then use these the estimated keypoints to create a reconstruction loss via the differentiable rendering approach, by projecting the 3D keypoints into a 2D shape at a fixed rotation $\theta = 0$. For our adversarial loss function, we randomly sample values for θ between $-\pi$ and π, and project our 3D data using that value as a rotation parameter, around a fixed y-axis. The first network is optimised to maximise the error rate of this prediction while the discriminator is optimised to minimise the error rate. The final loss functions we use are an L2 reconstruction loss between input images and rendered images, adversarial loss for our keypoint detector and lifter, and an adversarial loss for our discriminator.

6.4.3 Experiments

Using the toy dataset first established in Section 5.5.3.3, we experiment to see if we are able to not only predict articulation points, but also simultaneously estimate the corresponding z co-ordinates. Our results, shown in Table 6.2, demonstrate that encouraging keypoints towards articulation gives a good model for reconstructing the input images using differentiable rendering. However, when it comes to the depth inference, this approach is not enough to learn sensible 3D articulated structures as shown in Table 6.3.

As can be seen in these results, while 2D keypoints are somewhat placed in good locations and leads to decent reconstructions, the lifted points do not represent a reasonable structure. It can also be seen that the placement of keypoints not required for the reconstruction are placed around the edges with no connections predicted between them.

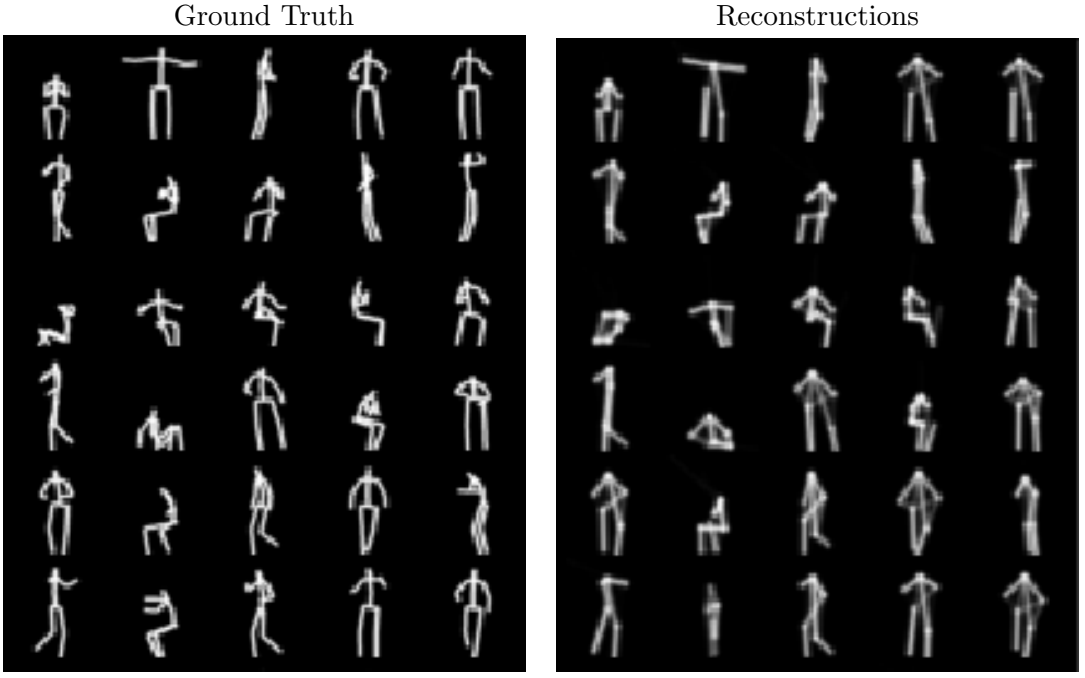


Table 6.2: Demonstration of input images and sketches that aim to reproduce the input using regressed keypoints, connection matrix and the differentiable sketching module.

Presumably, this is learnt by the encoder in part to minimise the generator's loss for the adversarial learning section.

6.4.4 Discussion

As we have seen, this approach succeeds in pushing keypoints onto articulation and end points, and achieving decent results in terms of two-dimensional reconstructions, but this is not sufficient to convincingly lift our estimated 2D keypoints into the third dimension. While selecting articulation and end points as keypoints should work, there is still further information required to piece together fully self-supervised keypoint detection and lifting.

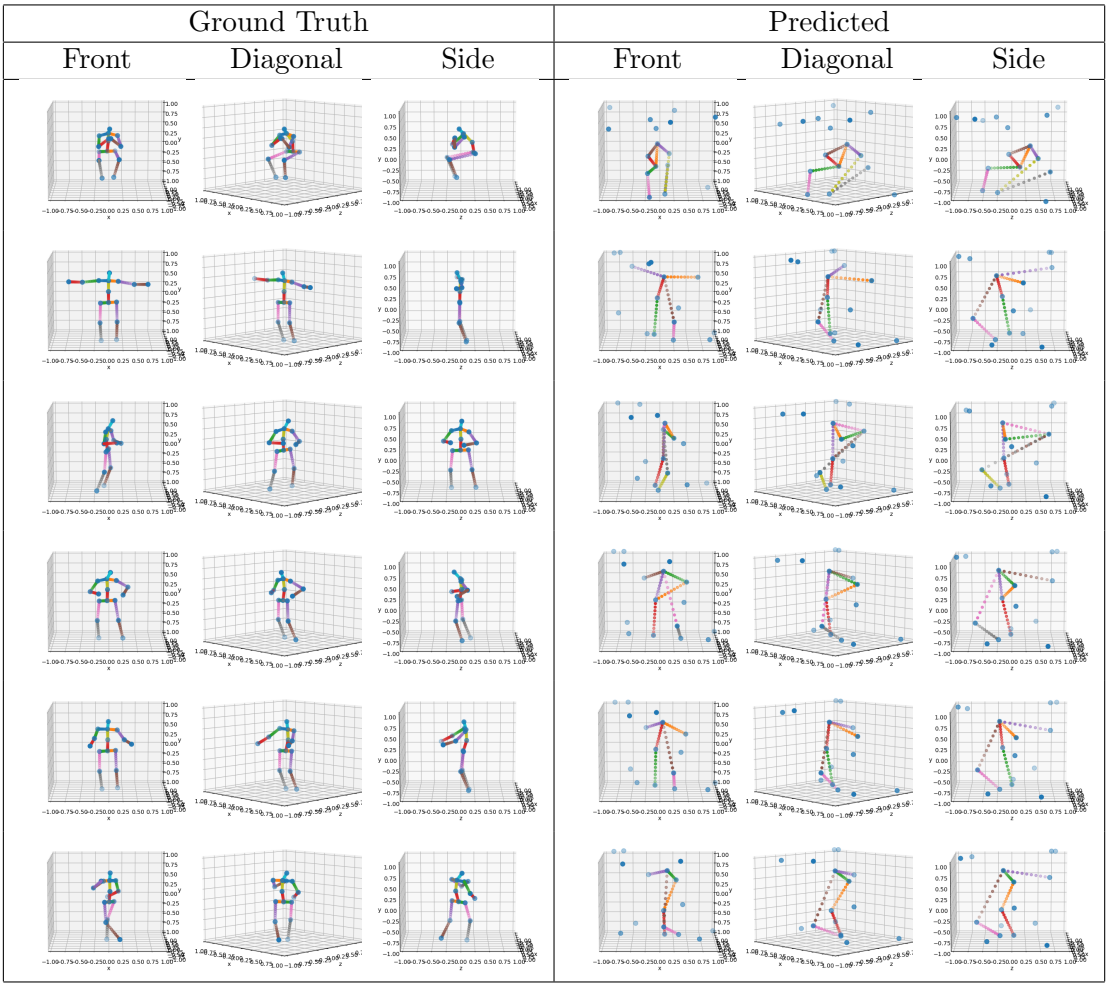


Table 6.3: Demonstration of 3D keypoints found when concurrently learning to place keypoints and lift them into three dimensions using differentiable sketching. More examples can be seen in Appendix D.

6.5 Full Pipeline with Bone Rigidity Prior

In the previous section we have shown that an image-to-3D approach with minimal prior knowledge is too ill-posed to learn a solution to the generalised articulated pose estimation problem. As demonstrated in Chapter 5, a bone rigidity prior can be used to both locate the articulated model and infer the depth of each two-dimensional keypoint to create a 3D articulation model. We also show how it can help to locate better 2d keypoints, aiming to locate points of articulation when used in conjunction with a constrained differentiable sketching module in place of a reconstruction network.

Our goal is to leverage the bone rigidity prior to simultaneously infer the depth of each keypoint, and to assist in location of two-dimensional keypoints, especially in cases with occlusion.

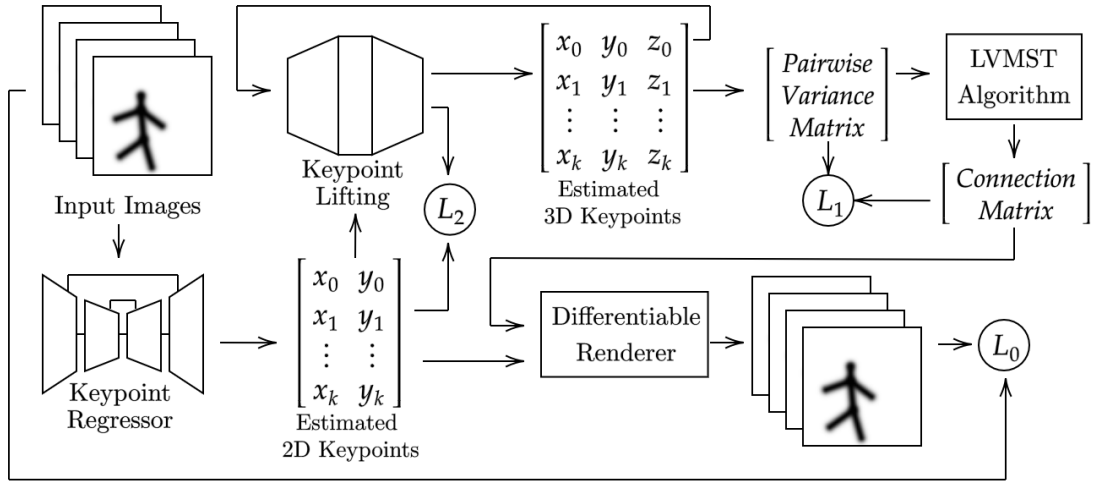


Figure 6.2: A diagram outlining the full approach using the bone rigidity prior. 2D keypoints are regressed from input images, before being lifted into 3D. The lifted keypoints are used to predict the connection matrix via the LVMST algorithm, which is then used in combination with the 2D keypoints to create a rendered sketch. The three losses, L_0, L_1, L_2 are summed using the Balanced Multi-task Learning framework to train the learnable parameters in the system.

6.5.1 Motivation

Putting all of these elements together into one pipeline is now the final step required for our generalised articulation model detection approach. This pipeline is best shown in Figure 6.2.

The following sections will break down the approach into the stages of the pipeline and describe the aim of each stage.

6.5.1.1 Keypoint Detection

Our pipeline starts with a keypoint regressor network which takes a batch of images of shape B, C, X, Y, and returns a batch of keypoint shapes $B, K, 2$, where each element represents K x, y co-ordinates. For basic datasets, a simple MLP suffices, but for complex datasets a convolutional network followed by a soft-arg-max operator is used.

6.5.1.2 Depth Estimation

Depth estimation is a simple step that uses a standard MLP as a depth inference network. Our estimated 2D keypoints from the keypoint regressor are fed through the depth inference network and output corresponding Z co-ordinates for each X, Y pair, and we append each Z co-ordinate to the inputs to get the estimated 3D keypoints.

6.5.1.3 Connection Matrix Estimation

Using the previously discussed LVMST algorithm, we can derive a connection matrix using this batch of estimated 3D keypoints. Our second loss function to give a training signal to our depth estimation network is then formulated by summing the result of an element-wise multiplication between our connection matrix and our pairwise distance variance matrix. Minimising this loss encourages our network to learn Z co-ordinates that maintain our bone rigidity consistency over the batch.

6.5.1.4 Differentiable Sketching

Our differentiable renderer then takes our 2D keypoints and connection matrix and renders a sketch where lines are drawn between the X,Y co-ordinates of every keypoint pair as determined by the derived connection matrix. We can now establish our reconstruction loss as a mean squared error between input images and these sketches.

6.5.1.5 Loss Functions

The loss functions required to minimise concurrently are as follows:

Reconstruction Loss (L_0)

Our reconstruction loss is simply formulated as the mean squared error between the input images and the reconstruction images generated by the differentiable renderer using the 2D keypoints and the connection matrix.

LVMST Variance Loss (L_1)

We would additionally like to impose a loss to enforce constant pairwise distances over the batch of the pairs of points connected by rigid bones. Our loss function to meet this aim is simply to minimise the sum of the dot product of our connection matrix and pairwise distance variance matrix.

Self-Consistency Loss (L_2)

Finally, we impose a self-consistency loss, as introduced in Section 5.4. The Z co-ordinates are then attached to the Y co-ordinates and Z,Y is then fed back through the depth inference network to estimate X co-ordinates for our self-consistency loss, which is calculated using mean squared error between the original value for X and the prediction of X.

We are also required to balance these losses using a monotonically increasing transformation function [68], and sum the transformed losses to get a final combined loss.

6.5.2 Implementation

The implementation for this approach is a combination of numerous elements from previous sections. The keypoint lifting implementation was previously discussed in Section 5.4 and re-used for this implementation. Likewise, our keypoint detection approach, where differentiable sketching was used to encourage keypoints to find articulation and end points, is unchanged from Section 5.5. During our experimentation, we have tried multiple neural network architectures for our keypoint regressor including a simple linear network when using images from our toy articulation dataset, and a convolutional network where images have greater complexity.

6.5.3 Experiments

As can be seen in Table 6.4, we are able to generate realistic two-dimensional sketches of the input images, which appear to capture the articulation of the subject with success, especially visible in the knee keypoints. The areas around the arms appear to be less successful but still manage to contain some convincing articulation. This is likely due to the increased complexity of the arm poses as seen in the dataset.

However, as can be seen in Table 6.5, we have not managed to capture the desired three dimensional structure. It is not intuitive to say exactly why this is failing, however it is likely to be due to flaws in the two-dimensional keypoints, and 2D keypoint errors will propagate and accentuate errors when points are lifted using the LVMST-based lifting.

6.5.4 Discussion

As we have seen, the complete pipeline when being trained end-to-end does not grant satisfactory results. One reason for this could be that it is difficult to simultaneously learn to place 2D keypoints when given an image of an articulated object, while also lifting those points into three dimensions. In an attempt to alleviate this problem, in the next section we will demonstrate the same pipeline but with the two sections being trained independently.

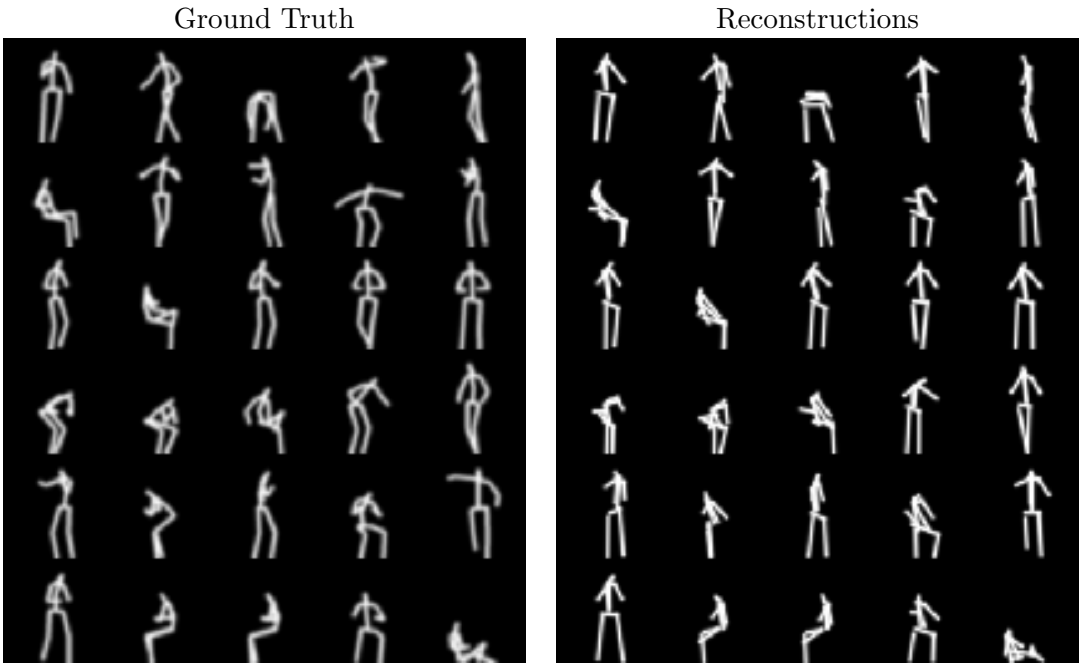


Table 6.4: Demonstration of input images and sketches that aim to reproduce the input using regressed keypoints, connection matrix and the differentiable sketching module.

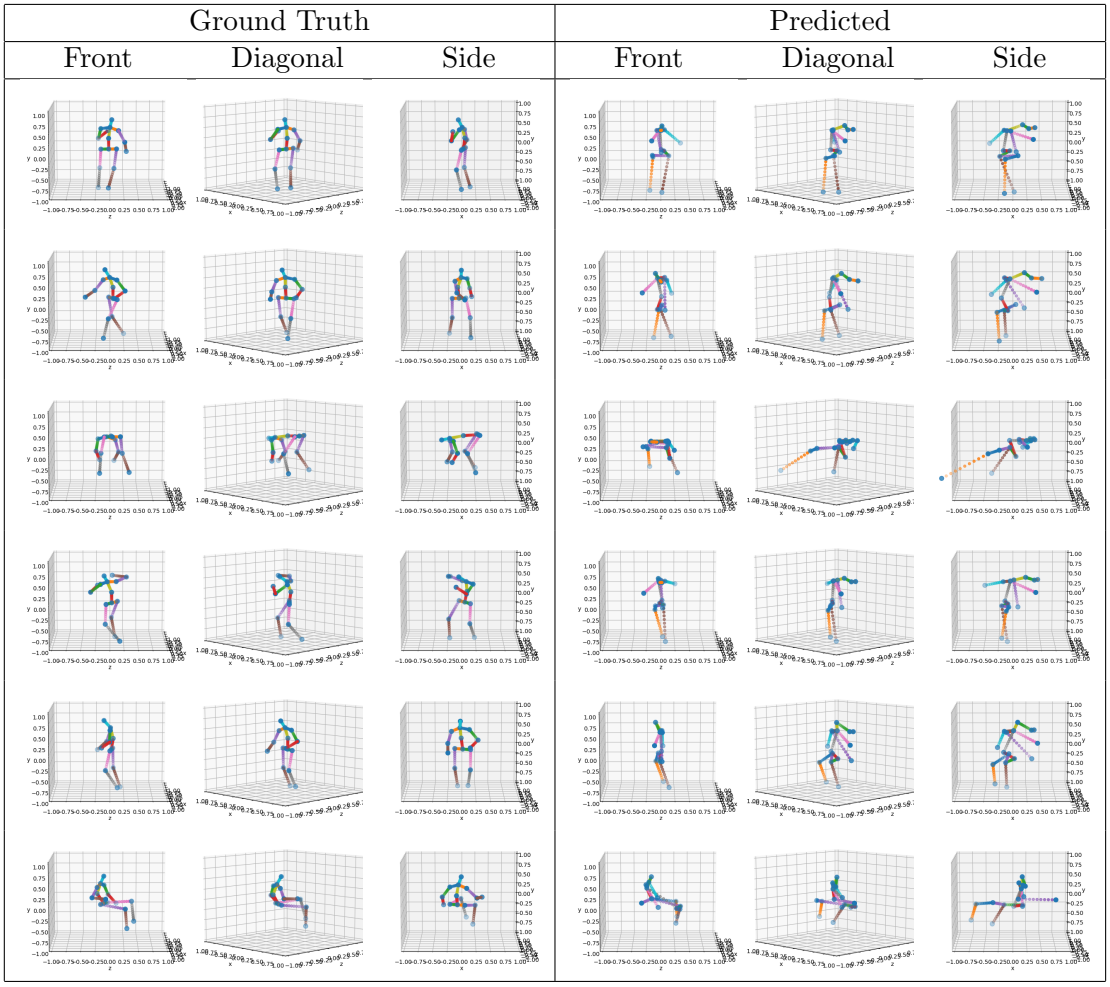


Table 6.5: Demonstration of 3D keypoints found when concurrently learning to place keypoints and lift them into three dimensions using our rigid bones prior. More examples can be seen in Appendix D.

6.6 Simplification via Pipeline Splitting

We have shown the failure case when attempting to learn the entire pipeline in an end-to-end fashion. In an attempt to derive success using these approaches, we have split the pipeline into its two components, keypoint detection and keypoint lifting, using the differentiable sketching based approach to find keypoints before lifting them using the LVMST approach.

6.6.1 Motivation

We have previously demonstrated success using the two individual approaches, but the combination of them into one system that is trained in an end-to-end fashion comes with difficulties. By splitting into two separate parts, keypoint detection and keypoint lifting, saving regressed keypoints from the former, before using those as the input to train the latter, we aim to avoid some of the problems that occur when training concurrently. If the predicted keypoints are an exact match to the ground truth points, then we can expect success, but any errors in the found keypoints will inhibit the results of the lifting stage.

6.6.2 Experiments

The first stage of this pipeline is shown in Table 6.6 as a recap, these results are no different to those initially discussed in Chapter 5. We then use these two-dimensional keypoints as the ground truth inputs for the keypoint lifting stage.

As can be seen in Table 6.7, this approach does not achieve satisfactory results for estimating three-dimensional structure correctly. However this approach does manage to derive some features in the structure, convincing articulation has been identified in the leg area.

The results of this further reveal the extent to which the limb variance minimum spanning tree approach to 2D to 3D keypoint lifting relies upon good quality 2D data that correctly captures the underlying three dimensional structure. However, this does not completely invalidate the usage of this algorithm in a self-supervised environment. We will discuss possible adaptations and future directions of the LVMST algorithm in more detail in the following section.

Ground Truth	Reconstructions
	

Table 6.6: Demonstration of input images and sketches that aim to reproduce the input using regressed keypoints, connection matrix and the differentiable sketching module.

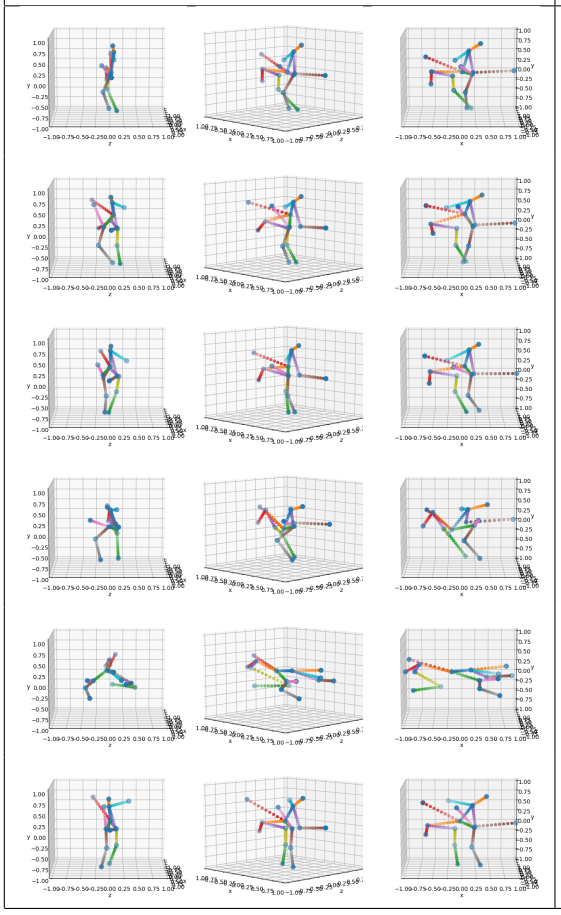
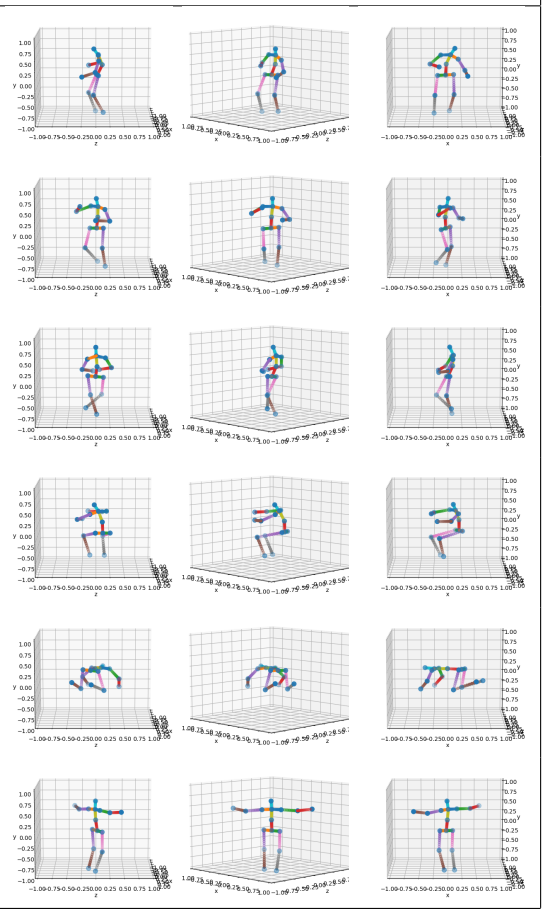
Predicted			Ground Truth		
Front	Diagonal	Side	Front	Diagonal	Side
					

Table 6.7: Demonstration of 3D keypoints found when using our rigid bones prior, but the full pipeline is split into keypoint detection and depth estimation. More examples can be seen in Appendix D.

6.7 Discussion

We have made some strong progress towards locating generalised articulation models, but there is still some way to go before we can be satisfied with these results. While we have shown we can both locate good keypoints from images and estimate depth from ground truth keypoints, doing both concurrently comes with difficulties.

The key take-away from our experiments concerning the keypoint lifting section of the pipeline is that using a bone rigidity prior for depth estimation is very sensitive to noise in the two-dimensional data. Small errors found at the keypoint detection stage will lead to large errors when we attempt to lift those points into three dimensions. The rationale for this is that bone rigidity based lifting is based purely on consistent limb lengths through-out a batch of data and exploiting that consistency to predict the depth of each point. Noise breaks that consistency and thus inhibits the ability to estimate depth.

Theoretically, this means that our approach could still work with further research. The requirement to obtain good results relies on a large increase in accuracy at the keypoint detection stage. While out of scope for this thesis, one approach could be to leverage a supervised or semi-supervised keypoint detection stage, using paired images and ground truth keypoints, to create an accurate and reliable keypoint detector. Integrating this into our pipeline with keypoint lifting and training in an end-to-end fashion could allow for a 3D keypoint detector which requires only 2D keypoints, and in a semi-supervised case, only a small number of labelled examples. We discuss semi-supervised learning relaxations further in Section 7.3.3.

6.7.1 Limitations

We note that our generalised self-supervised articulation estimation model has limitations, and we highlight these here.

6.7.1.1 Limb Variance Minimum Spanning Trees

Our LVMST method for finding connect-ability does have some limitations. We notice there are difficulties in balancing exploration and exploitation, we need to be able to explore for better points in the gradient descent optimisation while restricting our keypoints to a shape that captures some structure.

We also note that our differentiable implementation requires workarounds, and while these allow for the minimum spanning tree to be used within a deep learning system, there are edge cases where we may find errors. If the variance of two limbs is identical across the batch then errors can occur in the soft-arg-max function, selecting the index

that lies half-way between the pair of minimum values. This is hugely unlikely, and can be ignored in most cases, however if this were to be used in safety critical environments, then this is a problem that must be addressed.

We have also not implemented any outlier detection within our LVMST algorithm, which could potentially cause misclassified keypoints from the keypoint detection stage to break later stages of the pipeline. Outlier detection could be used to discard individual poses from the connection matrix calculation to ensure an error in the batch does not propagate to other poses. A sophisticated error correction approach could then be used to reverse engineer the keypoints in the erroneous poses, using the expected limb length to better estimate the keypoints. This could aid in examples where occlusion is present in an image.

6.7.1.2 Differentiable Sketching

Differentiable sketching gives us the power to restrain the capacity of the downstream network, enforcing stronger keypoints which are more likely to lie on joints and the ends of limbs. However, when dealing with real world images, we must make some adjustments to create a strong gradient between the sketched and input images.

Adding learnt colours to our sketched lines is an initial step, but in real life, it is unrealistic that this gives enough power to draw the articulated object in the image. A common example of a failure state is a fore-arm on a person wearing a T-shirt, where the top half will be coloured by the T-shirt and the lower half as the colour of their skin. Colouring each end of the line with a gradient between the two could be one option to allow for this to be learnt.

Learning the widths of each line is also essential for sketching real world images, as not all sections of an articulated object will be the same width. Adding a constraint for the maximum and minimum line widths would be essential for this adaption, as the network could learn to use the lines as large brush strokes to colour areas of the image, or setting the line width to zero if a keypoint is redundant for a reconstruction.

In all, if we would like the differentiable sketching approach to be widely applicable for learning articulation, we must carefully balance the capabilities such that we can draw a coloured articulated model, without allowing for too much control, so that the found keypoints are still required to capture a strong structure.

Another consideration is differentiable sketching in three dimensions, which we will discuss in the future work section, found in Section 7.4.

6.7.2 Challenges when using Real (In the Wild) Images

When applying this theory into practical applications, there are numerous uncontrolled variables that we may encounter, and could cause issues within our approaches.

The first is the camera perspective, which could be managed using a disentangled camera and simulating the projection matrix. Distance between camera and person currently breaks the pairwise distance assumption, and would require a more sophisticated data normalisation stage to solve. Occlusion, in both self-occlusion and object occlusion, will add uncertainty to any hidden keypoints. Multi-subject images could potentially be solved by appending an object detector to the start of the pipeline and running with the cropped frames containing individual subjects. And finally, extreme poses that lie out of training data distribution would require better generalisation or a better training dataset to create satisfactory robustness.

6.7.3 Occlusion in Self-Supervised Learning

Occlusion has long been a challenge in computer vision [32] but in our self-supervised learning case, it can lead to problems at both training and inference time. As all the data are able to work with is contained within the image itself and not labels or annotations, occlusion can remove that data from our training images leaving us with data points that are useless at best and damaging to training at worst. Some work by Reddy et al. [91] has begun to look into self-supervised 3D keypoint detection with occlusion, but do not extend their solution to dynamic objects such as those containing articulation. As discussed by Jalal and Singh [52], there are three different sorts of occlusion that must be considered; self-occlusion, where part of an object blocks itself, inter-object occlusion, where two objects in the image overlap, and background occlusion, where part of the background occludes the object.

To work around this, a simple idea is to select a dataset that contains no occlusion, but in our case, articulation naturally leads to occlusion, especially self-occlusion, and even with an easier dataset, we would result in a network with no robustness to occlusion.

6.8 Conclusions

Finding a good articulation model in 3D, using only self-supervised methods comes with many difficulties. Prior knowledge is essential when training with no ground truth keypoints, as shown in Section 6.5, but selecting the right prior is important. Too strong a prior will overfit our approach to a subset of articulation models and too weak will yield poor results. As we have shown with the bone rigidity prior in easy examples,

we manage to achieve good 2D results without overfitting, but applying this theory to complex datasets comes with difficulty.

Referring back to RQ1, finding a self-supervised articulation model comes with difficulty, but we have made strides of progress towards achieving this aim. The individual components that build up an articulation model can be achieved using self-supervised learning and by employing a suitable prior we can find a further improvement in our results.

Chapter 7

Conclusions

The original aims of this thesis were ambitious. Self-supervised deep learning is still relatively in its infancy which leads to difficulty in finding satisfactory solutions, but that is not to dismiss the original contributions towards the field.

In this chapter we will conclude the work covered in this thesis and look at the wider picture of how it contributes to the fields of self-supervised learning and articulation models. We will discuss issues that have remained unsolved within the time-frame of this thesis, work that lies outside of the scope and future directions this work could take.

7.1 Broader Impact

The merits of our research in this thesis are not solely to solve the problem of determining three-dimensional articulation models using self-supervised learning. In this section we will discuss how our approaches have a broader impact to other fields of research, and how this work can be translated into solutions for other domains.

7.1.1 Pose Lifting in Dynamic Shapes

While our approaches to inferring depth through lifting are specific to articulation models, there are elements that could be repurposed in other domains, for example the batch lifting consistency via the LVMST algorithm. The advantage of this approach over previous approaches, such as adversarial based pose lifting, is that we simultaneously learn the average lengths of the rigid bones of the articulation model captured by the data, while also applying those learnt lengths to the task of pose lifting. There is a caveat that the dynamic shapes must contain a structure of rigid bones for this approach to be successful, which may limit usage of this approach outside of articulation models. Additionally, there cannot be too much variance in the bone lengths of the data points

within the dataset, but small amounts of variance should not affect performance of the algorithm.

7.1.2 Potential Articulation Modelling Applications

Deriving a 3D articulated pose will realistically never be the final goal of an image processing pipeline. Other downstream tasks that may require articulation model keypoints include, but are not limited to:

- Action detection
- Tracking for medical diagnostics
- Augmented/virtual reality
- Security or surveillance

Generalising the detection of articulated poses will allow for these applications, which are typically restrained to only human poses, to be applied to any subject containing rigid boned articulation regardless of its skeletal structure, without needing a customised pose estimation approach to be devised.

7.1.3 Differentiable Minimum Spanning Tree (DMST)

Our differentiable implementation of the Minimum Spanning Tree algorithm, covered in detail in Section 5.3.1.1, has been shown to be useful for our purposes of learning articulation models. But locating minimum spanning trees is ubiquitous across many fields, and if any of these applications are being used in deep learning, our differentiable approach could solve the problem of passing gradients through a minimum spanning tree algorithm. Research interest in Graph Neural Networks (GNNs) has recently grown, owing to the power of deep learning and representing data in the form of graphs [21]. A minimum spanning tree has many applications within any graph, but is typically a non-differentiable operation due to the use of an arg-max function. As Loukas [75] states, a graph neural network cannot learn a minimum spanning tree, however we have created an implementation that allows for a relaxation of the MST algorithm to allow for gradients to pass through. While our implementation does not use a GNN, we see no reason why it cannot be extended for use within a GNN environment.

7.1.4 Spatially Constrained Representation Learning

Our approach to keypoint detection using multi-task learning, as covered in Section 4.4, gives an insight into learning spatially constrained representations from images. We

show that multi-task learning gives an increased level of generalisation when detecting keypoints on an image, locating stronger representations when the downstream task is unknown and generalised to the extend of being transferable between similar datasets.

7.2 Unsolved Issues

Within the scope of this thesis, we were not able to solve every issue brought up during our work or research every element in this field. This section will outline some of the issues we have faced along the way and describe our initial thoughts on how they could be approached in the future.

7.2.1 Improving Downstream Rendering

The differentiable sketching used in our implementations was a simple and naïve approach to creating sketches based on a set of keypoints and a connection matrix, but this limits the variety of datasets that can be used to derive an articulation model. If instead we used a more sophisticated renderer, similar to Li et al. [66], and use our derived keypoints to control more complex parameters instead of simply endpoints of sketched lines, then theoretically a larger range of datasets could be used given a sufficient approach to differentiable rendering. But we would have to carefully select the parameters that a network has the power to learn. As discussed in Section 5.5, the reason why better keypoints are located using differentiable sketching is due to a restriction on the reconstruction ability, requiring more information to be encoded in the points in order to get a low reconstruction loss. If we had too much flexibility then we may return to the problem found using a standard reconstructor network where keypoints do not capture the articulation of an object.

7.2.2 Self-Supervised Keypoint Metrics

What is the best way of evaluating keypoints found with a self-supervised method? We are yet to determine the best way of measuring the effectiveness of our keypoints in an ideal way. For landmarks in pose estimation, a distance metric from the estimated to the ground truth is traditional, and effectively measures the accuracy against the target. But as we are interested in general keypoints, that capture a spatially constrained representation, this is not always suitable due to lack of data availability. In our evaluation in Section 4.6, we attempted to use a few novel metrics, including performance on an unseen downstream task, but this it not without issues. The difficulty comes when we must select a downstream task that effectively measures our keypoints performance without bias. There is the risk of selecting a metric which we then overfit our keypoint detector to, so a range of varied unseen downstream tasks may be one possible solution.

7.2.3 Non-Rigid Limbs

Our rigid bones prior breaks down in the scenario that we wish to learn an articulated pose that does not have rigid limbs, for example an articulated machine that uses hydraulics that are able to extend or retract. This feature breaks our assumption of rigid pairwise distances when two joints are connected. How would we derive three-dimensional keypoints when the distance between two fixed joints can change between poses? There may be some image cues where the articulation mechanism changes appearance based on how extended the hydraulic section is. Additionally, with enough image examples we should be able to derive a maximum and minimum limb length based on the physical limitations of the hydraulic module. This could then be used to give a range of values for the depth of our articulation points, and expected values can be estimated from that range. The image data could also be used as a visual cue to work out if a limb is in its extended or retracted state.

7.2.4 Vertebrae

While the discs that connect to create a vertebrae do obey our assumptions of rigidity, their scale compared to the image space shape will lead to some variance in distance between either ends of the vertebrae. For our experiments we have assumed that in most cases, this variance will be marginal but in some scenarios this may not be the case. One solution is to learn many more keypoints such that they approximate a one-to-one mapping between disc and keypoint, but in practice this would be difficult. Another solution would be to model flexibility, such that the limb length is fixed but does not have to be a straight line, so a backbone could be modelled with more realism while maintaining the fixed length.

7.2.5 Pipeline Section Interdependence

As discussed in Section 5.3.1, to determine a good 3D model using our LVMST approach, we need good keypoints that lie on the articulation points of our subject. But we rely on a good learnt 3D model to encourage our keypoints to be placed in the correct locations, especially when we have difficult examples such as those with occlusion and self-occlusion. This means that the performance of both elements of our pipeline require the other to be accurate to correctly train. Other possible solutions fix this sort ill-posed problem would use a strong prior, such as 3D poses taken from a separate dataset [51] or ground truth 2D points which then gives a simpler task of 2D to 3D lifting.

7.3 Work that Lies out of Scope

This thesis considered a range of topics, but with limited time, there were some logical extensions that we were not able to be included in the scope of this work.

7.3.1 Video Sequences

It may be the case that the cues we require to determine a three dimensional articulation model without using labelled data is found within sequential data. As humans we experience a world that moves, so it is possible that we identify articulation through movement rather than through examples of an object in different poses. Using video data still keeps this solution as a self-supervised approach and although this will prevent the usage of any non-video datasets, we still have a variety of video datasets available to use, for example Human3.6m[48], as well as an ever increasing amount of video data available from the internet. The use of self-supervised techniques through this thesis allows for transfer-ability to new datasets without costly and time consuming labelling being required for each new dataset.

7.3.2 Differentiable Rendering in Three Dimensions

As seen in Section 5.5, we can use differentiable sketching to turn a spatial representation back into image space for use with a reconstruction loss, but naïvely extending this into three dimensions quickly runs into memory issues. This is only the case as the simplest extension uses voxels instead of pixels, and other rendering methods with lower computational complexity would be preferable in this case. Liu et al. [71] implemented one such 3D differentiable renderer, which rasterises based on 3D meshes with camera parameters. But in our generalised, self-supervised environment, a differentiable renderer using a coloured 3D mesh would be difficult unless given a strong prior beforehand. Deriving a satisfactory complex 3D mesh using self-supervised learning may be an option, but will certainly be difficult to optimise.

7.3.3 Semi-Supervised Relaxation

Self-supervised learning is currently of academic interest and has applications in many areas where labelling data is difficult, costly or time-consuming. However, there are clear benefits of supervised and semi-supervised learning for applications in the real world, when practically creating a reliable deep learning based system.

In many cases, semi-supervised learning is a more feasible solution than either supervised or self-supervised as it requires much less labelled training data than supervised learning

approaches while achieving superior performance than purely self-supervised. We have identified some possible strategies for transferring these self-supervised techniques into semi-supervised approaches and discuss the implications of using a small amount of labelled data.

7.3.3.1 Semi-Supervised Multi-Task Approach

As we have established a multi-task framework, the addition of a supervised task to enrich the training is a feasible solution to creating a semi-supervised training approach. The implementation is simple, taking our existing self-supervised training losses and adding an additional supervised loss to the multi-task framework that aims to regress to a target taken from labels in the dataset. The loss could be one of two options, two-dimensional keypoints for our keypoint regressor to aim towards, or three-dimensional keypoints for our pose lifting network to predict.

As we are considering semi-supervised learning, we would not expect to have keypoints for every data point, but this should not matter as we can only include the loss in batches that contain corresponding labelled keypoints. If required, we can artificially inflate the supervised loss to put greater priority to its training signal over the self-supervised losses to compensate for the sparse signals generated from the supervised loss.

7.3.3.2 Post-Processed Semi-Supervised Approach

As described in Section 4.6, and used as an evaluation metric, one approach to adapting our approach to semi-supervised learning is to regress the self-supervised keypoints towards the ground truth using a smaller set of labelled points. We have demonstrated this for two-dimensional keypoints but extending to 3D would also be viable, and would be just as simple to implement. With a small amount of training data, we may be able to dramatically increase the 3D results, as long as the data we have is unbiased and covers a wide enough range of poses.

Another option in this area would be having a base self-supervised model that provides unspecialised keypoints on a wide range of articulated structures, and using a different semi-supervised regressor, perhaps a neural network with a greater level of complexity, for each dataset. This approach, becoming more popular with transformer models, could allow for a singular powerful model to be trained once, and quick training for the regressor to specialise to datasets at a later stage.

7.4 Future Work

This section aims to look at the future work that logically flows from the content of this thesis. It will consider improvements to the approaches taken, but also extensions that are currently a distant goal for self-supervised articulated pose estimation. Self-supervised learning is still relatively in its infancy, but has big potential to tap into the vast quantity of unlabelled data that exists. However, implementing successful approaches that use self-supervised learning comes with difficulties, and care must be taken to add enough prior knowledge and create loss functions that manage to successfully train a network.

7.4.1 Differentiable Sketching in 3D

Images that contain information in the depth dimension, representing three-dimensional poses, are inherently more complex than those that are two-dimensional. It would be advantageous to give our differentiable sketching component the option to exploit depth so that sketches can replicate occlusion present in images. As the memory requirements of sketching in 3D using voxels and flattening down into a 2D image are much too high, we propose a method for learning a permutation for our simple renderer to order the lines drawn, such that those at the back of the scene are identified to be drawn first. Then by using composite function that overlaps pixels with those at high indices in the permuted tensor. The permutation we use can either be learnt by the neural network, so that for a better sketch to be made, the lines must be ordered as they appear in the image, or we can derive it by the depth of our points located from our 3D estimation.

If we choose to learn our raster permutation, we require a differentiable way of reordering each line raster into the learnt permutation. Ideally our permutation matrix will contain only 0s and 1s, and each row and each column will sum to 1, giving a linear assignment from input to output permutation. However, as this is a learnt matrix, we must relax this constraint and approximate the assignment such that we maintain a gradient through the permutation re-ordering step. In this case, the Sinkhorn operator could help to convert our learnt permutation matrix into one that is approximately doubly stochastic. A side-effect of this is that rasters will no longer contain a singular line, but darkened pixels of multiple rasters, in the case that the permutation matrix contains values that are continuous between 0 and 1 exclusive. In our output renderers, this can give a soft estimation of occlusion, as pixels of overlapping lines are blended to create a sliding scale of occlusion.

Another avenue of research to consider could take inspiration from Wan et al. [105] and similar work, which uses differentiable rendering of a simplified shape in three-dimensions, and projecting into two-dimensions using the silhouette of the shape to create a reconstruction loss with the original input image. This approach may suffer

when it comes to self-occlusion, and their approach used a fixed articulation model that approximates a hand. In order to generalise this approach, it would be necessary to learn the 3D model to fit to the images from the data which could lead to difficulty in learning a suitable model.

7.4.2 Further Generalisation of Articulation Models

One common trend we see in unsupervised and self-supervised techniques is the requirement for heavy hyper-parameter tweaking [118] and our approach does not entirely avoid this. We must set the number of keypoints we are looking for prior to training, which ideally would not be a requirement. The difficulty here is simultaneously training a network to find keypoints while also selecting the optimal number of keypoints to match the articulated structures found in the dataset.

7.4.2.1 Dealing with Unknown Number of Articulation Points

Currently we restrict our implementations to a static number of keypoints that is manually chosen beforehand, meaning a small amount of hyper-parameter tweaking before training on a new dataset. One consideration in selecting this parameter before training a model is based not only on the type of articulated objects that we are interested in, but also the scale of the object within each image. While calculating this using the number of bones found within an animal could be a sensible predictor, many of those bones are at too small a scale to be located within the image.

Our initial thoughts are to select an upper bound of keypoints, above the realistic maximum number needed for any articulated structure and selecting a subset while training, aiming to both reduce data redundancy and maximising information captured in our spatial representation. From a brief survey of existing literature, this is an approach yet to be taken in the self-supervised 3D pose estimation space. This could be implemented using soft-arg-max, and then selecting the heatmaps that represent local maxima. Using these curated points, we can then build an articulated shape using this subset, assuming some consistency between structures in the batch. This may not be quite so simple in practice, our method for finding connectivity based on a LVMST scales poorly with keypoints ($\mathcal{O}(n^2)$) due to the calculation of pairwise distances between every pair of points. This is exacerbated further as gradients must be calculated throughout for the backpropagation of our neural network. Because of this, the upper bound that can be chosen for this hyper-parameter is limited.

Another option is to simply learn a large number of keypoints through-out the training stage, assuming there will be redundancy when capturing the desired structure, and then pruning at inference time. The pruning method should then remove redundancy

between keypoints, keeping only the most salient points that capture a desirable structure. Pruning could take the form of an optimisation problem, which keeps the minimum number of keypoints required, while maximising information.

The choice of these proposed approaches may vary based on scenario. If the downstream task is not known then pruning later may help preserve relevant data, with a pruning technique specific to the scenario. But if the downstream tasks used require fewer points, either for computational complexity or for a task specific reason, then the keypoint selection will be best during training.

7.4.2.2 Robustness to Variance in Articulated Structure

Currently, our LVMST algorithm requires a batch of objects with identical articulation structure and limb lengths, as a prerequisite to infer connections and depth. But in some cases we may desire a solution that could infer the articulated structure in a mixed dataset containing a high variance of structures, either varying bone structure or varying limb lengths. This would be ideal from a generalisation perspective as it would no longer require us to retrain our network for each type of articulated object. Training to this level of generalisability would be very difficult however, as convolutional layers would need to be able to adapt to a range of shapes and appearances. This would possibly be easier to solve with the video data as described in 7.3.1, possibly leveraging optical flow to locate consistent points where flow changes direction, signifying an articulation point. If we were dealing with video data instead of image data, then we could still use the LVMST algorithm, but taking each batch as a set of frames from a video, assuming a video contains only one articulated object. Our derived connection matrix will then have the flexibility to change on a per video basis, allowing for a mixed batch, as long as our keypoint detector could deal with this higher level of variance. This implementation will also rely on the variable number of keypoints as discussed in Section 7.4.2.1.

Another option for dealing with varying structures is to work from a bottom up approach, similar to traditional work around poselets [5], with a network that proposes an undefined number of body parts before using those to build up an articulation model via neighbouring end-points. An extra step is required to correlate batches of varying structures, with zero-ed out elements where no correlating limb was found within an image.

7.4.3 Building Robust Models

As with any computer vision system, we cannot rely on the image data being perfect examples of the object we are interested in. Whether occlusion is present and some elements are hidden, either by other objects or the object itself, or examples are visually

different to the expectation, building robustness is essential. As self-supervised learning is a data-centric approach, we must also be capable of developing robustness to imperfect training data.

7.4.3.1 Robustness via Artificial Noise

One approach for adding robustness to models is to add noise through data augmentation with image transforms. While some noise could aid in robustness of keypoint detection, image skewing and non-linear transforms such as thin plate splines would break the rigid limb length assumption. Noise can also be added to the 2D keypoints to add robustness within the 3D lifting stage, but would have to be minimal to prevent breaking the LVMST algorithm.

7.4.3.2 In the wild data

Another element of robustness to consider, from both a training and testing dataset viewpoint is "in the wild" data. When images are captured in a lab environment, many properties such as camera angle, lighting, occlusion and positioning within the image are carefully controlled to allow for a constrained dataset to work with. But in reality, an implemented system cannot reasonably make the assumption that any of these properties are carefully controlled. As discussed previously, adding robustness to a model will allow it to better deal with "in the wild" data, but we can also consider using those datasets to train our models to instil robustness in a data driven way. In addition, using self-supervised learning means that we do not require labels on these datasets, so images and videos can be quickly scraped from a web source and used to train a model.

7.5 Revisiting the Research Questions

In Section 1.4 we introduced our research questions for this thesis, and have referred back to these throughout. We will now conclude and explain how we have answered, or attempted to answer each research question.

RQ1. How can we learn a self-supervised articulation model?

We have demonstrated in this thesis the components required to take an image and learn a 3D articulation model. While a full self-supervised pipeline that learns to lift 3D poses from images was unsuccessful, the individual components were able to locate 2D keypoints and able to lift 2D keypoints to 3D. We also demonstrated how our LVMST

algorithm was able to infer the connectivity of 2D keypoints, while also lifting them into 3D.

RQ2. In the context of articulation models, what is a keypoint?

The semantic definition of a keypoint varies depending on the context. We have demonstrated keypoints as a generalised spatial constraint in Chapter 4, but also as landmarks that signify points of articulation and limb end-points in Chapters 3, 5 and 6. Particularly in Section 5.5, we demonstrated the differences between capturing generalised keypoints and specific landmark points. In the case of articulated pose models, landmark points of articulation and ends of limbs are the ideal keypoints to derive the articulation found within the object.

RQ3. How can we effectively represent spatial constraints?

We have demonstrated in Chapter 4 how keypoints are a suitable spatial constraint that manages to be generalised, interpretable and captures purely spatial information, compressing information into a small vector of keypoint coordinates. But when using keypoints, there are still design decisions that must be carefully considered. List and sets of keypoints are semantically different, as mentioned in Section 4.7.1, and in some cases a set of keypoints is the appropriate choice, even though they contain less information about keypoint correspondence than a list. We also need to consider if our keypoints are generalised points or are landmarks, in the latter case there exists a perfect keypoint for each point in each image, but this does not hold in the former case.

RQ4. What is the minimum prior information required to solve self-supervised articulation model estimation?

During this thesis, we have demonstrated the applicability of a simple limb rigidity prior to solve the constituent parts of the 3D pose estimation pipeline. But as we were not able to fully solve a self-supervised pipeline to locate articulation models from images, this prior may not be strong enough for this task. We believe that this approach has great promise, and possibly just more research to locate 3D keypoints in a self-supervised environment.

These research questions have guided our approach throughout this thesis, and we have to produced answers or partial answers for each question. Some have been successful but some questions are yet to have a satisfactory answer. As is the nature of research some of these answers have brought to light new questions such as “how do we determine the best keypoint representation for each dataset?”.

7.6 Final Concluding Remarks

This thesis has investigated the discovery of generalised articulation models from images using self-supervised learning. Through the research process, we have contributed original research to 2D-3D keypoint lifting, self-supervised multi-task keypoint estimation, and invention of a simple but effective generalised articulation prior. The final approach to estimating 3D models of articulation with no labelled data and no strong priors did not manage to provide satisfactory results, but that is owing to the difficulty of the problem over the lack of original contributions to the field. Future work in this field should take inspiration from our advancements in generalisation through both multi-task learning and designing of generic priors that elegantly solve difficult self-supervised problems.

Appendix A

A.1 Factorised Auto-encoder based keypoint lifting

A.1.1 Network Parameters

Pose residual encoder:

Layer Type	**Input Size**	**Output Size**	**Activation**
Linear	$2k$	$25k$	ReLU
Linear	$25k$	$50k$	ReLU
Linear	$50k$	$25k$	ReLU
Linear	$25k$	l	TanH

Pose residual decoder:

Layer Type	**Input Size**	**Output Size**	**Activation**
Linear	l	$50k$	ReLU
Linear	$50k$	$3k$	TanH

Camera Parameter encoder:

Layer Type	**Input Size**	**Output Size**	**Activation**
Linear	$2k$	$25k$	ReLU
Linear	$25k$	$30k$	ReLU
Linear	$30k$	$25k$	ReLU
Linear	$25k$	3	TanH

Mean 3D pose network (latent space of 1 approximates a very simple scaling parameter):

Layer Type	**Input Size**	**Output Size**	**Activation**
Linear	$2k$	$25k$	ReLU
Linear	$25k$	1	ReLU
Linear	1	$25k$	ReLU
Linear	$25k$	$3k$	TanH

Where k represents the number of keypoints and l_p represents the pose latent space size.

A.2 Adversarial 2D-to-3D Estimation

A.2.1 Hyper-Parameters

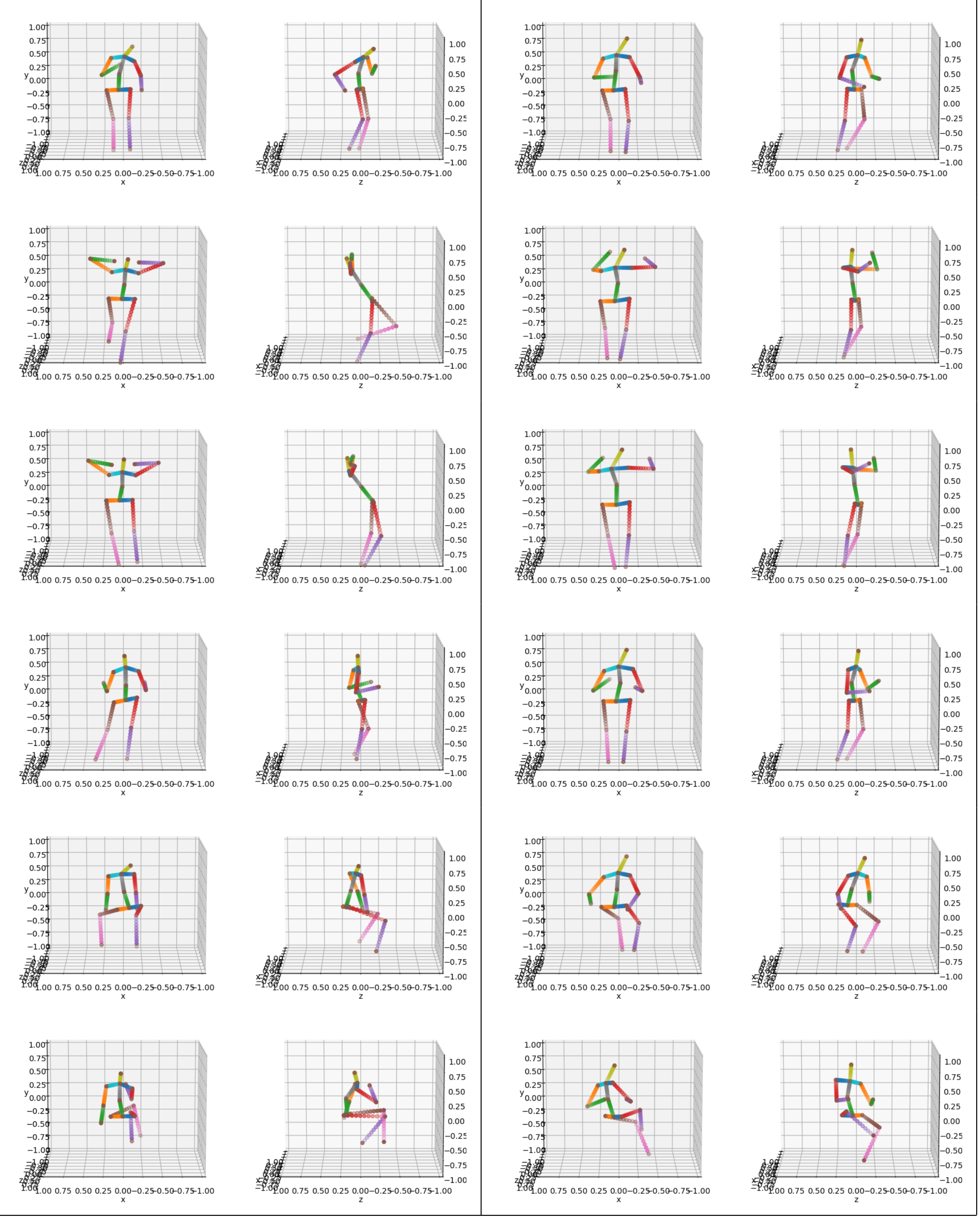
Parameter	Value
Number of Keypoints	16
Pose Latent Size	7
Batch Size	512
Keypoint Lifting Learning Rate	1e-5 (multiplied by 0.1 after every 100 epochs)
Discriminator Learning Rate	1e-6 (multiplied by 0.1 after every 100 epochs)
Epochs	1500

Table A.1: Hyper-parameters used to train keypoint lifting network using adversarial approach.

A.2.2 More examples using Adversarial Approach

Table A.2: Results showing 3D estimations from our adversarial pose lifting approach using 2D data from the Human3.6m dataset [48] as inputs.

3D Ground Truth		3D Estimation	
Front View	Side View	Front View	Side View



Appendix B

B.1 More Keypoint Examples

Below are more examples of keypoints on each of the testing datasets

B.1.1 MNIST

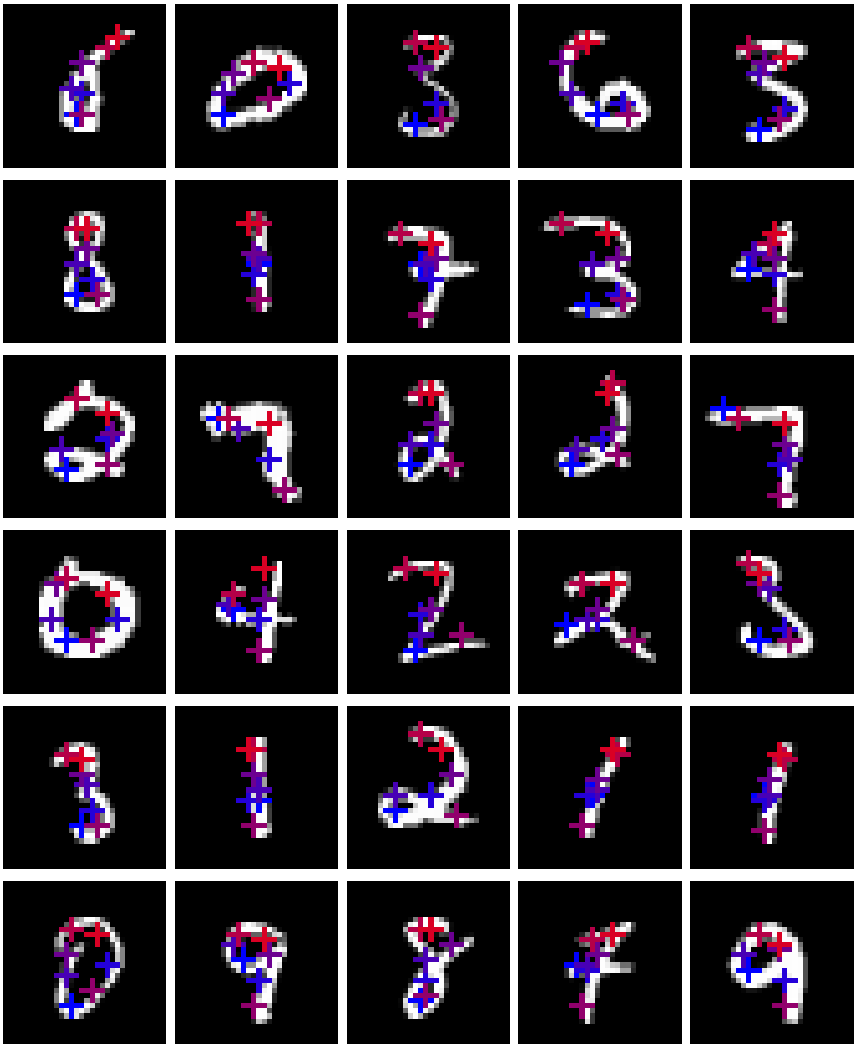


Table B.1: More examples of keypoints found on the MNIST dataset

B.1.2 Fashion MNIST

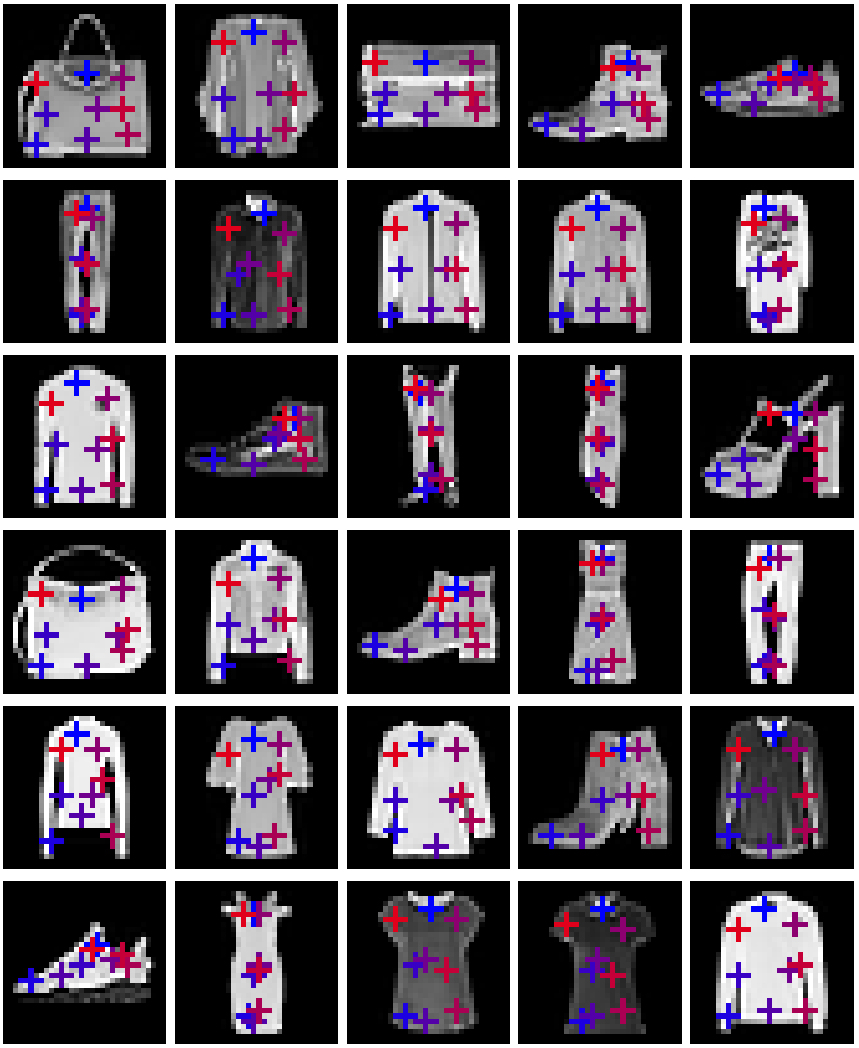


Table B.2: More examples of keypoints found on the Fashion MNIST dataset

B.1.3 Shoes

Table B.3: More examples of keypoints found on the Shoes dataset

B.1.4 Human3.6m

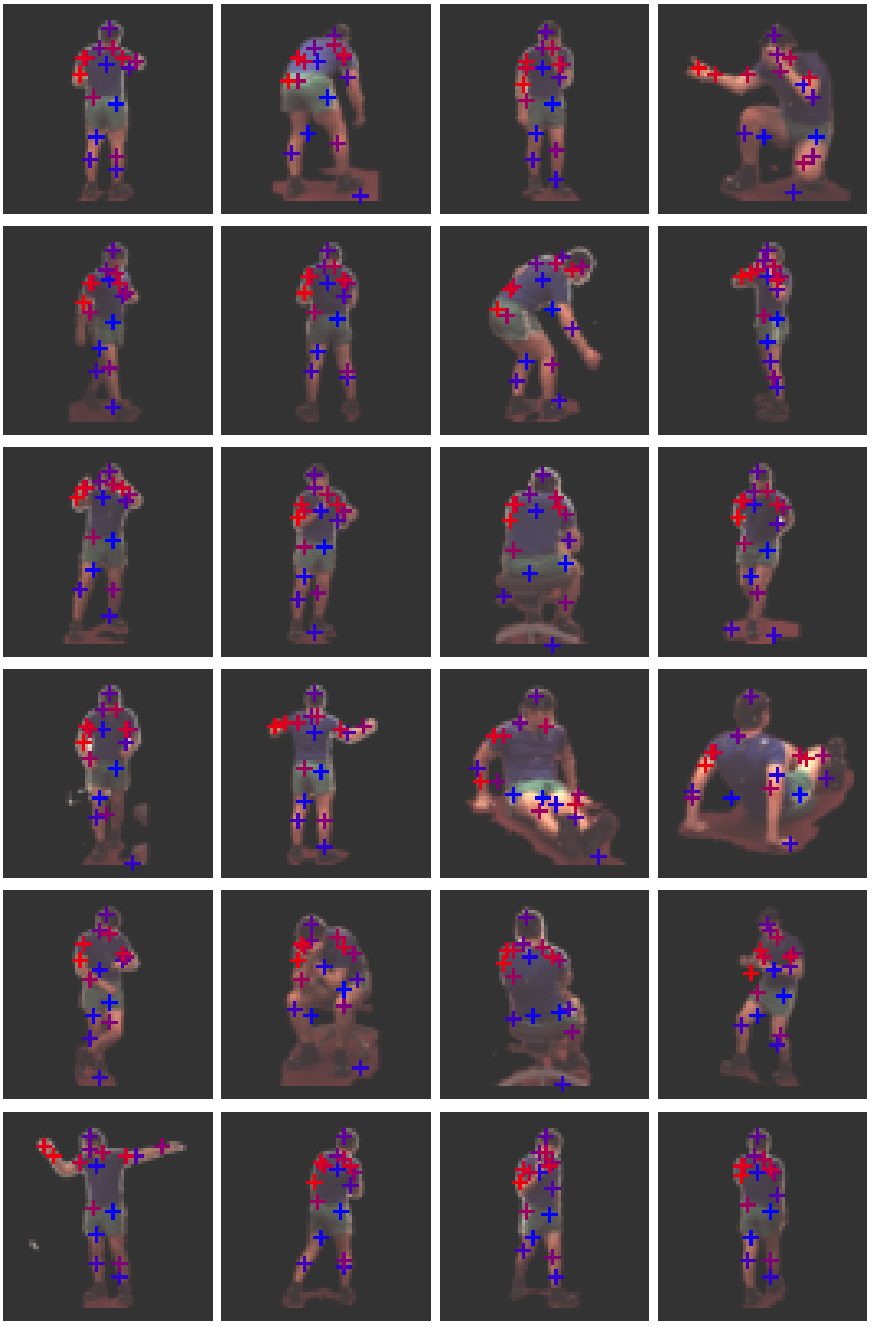


Table B.4: More examples of keypoints found on the Human3.6m dataset

B.1.5 Regressed Human3.6m

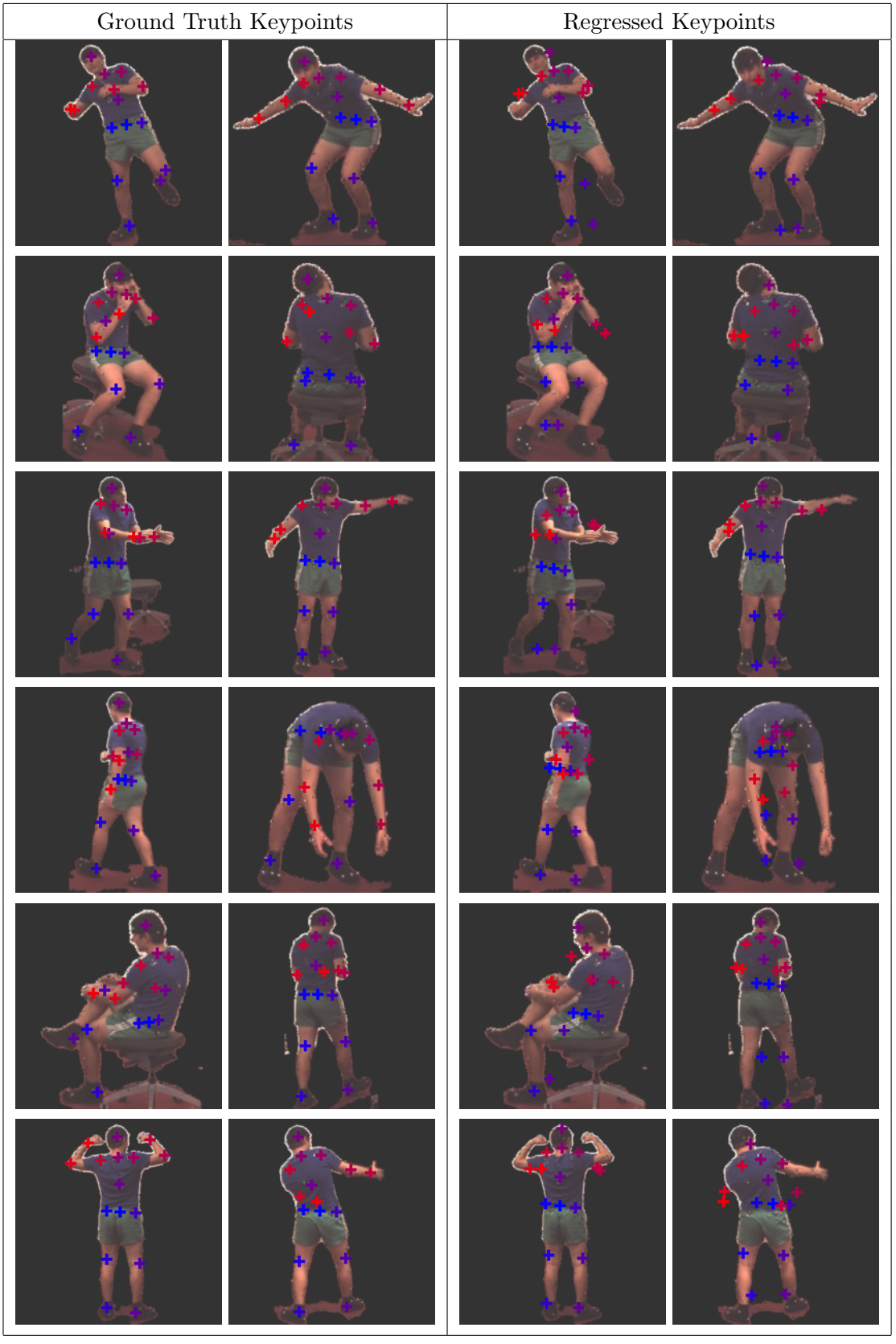


Table B.5: More examples of regressed keypoints found on the Human3.6m dataset compared to their ground truth counter parts. Colours are a scale between blue for keypoint at index 0 to red for keypoint at index k, and are consistent between columns.

Appendix C

C.1 Pose Lifting with Rigid Bones Prior

C.1.1 Further Experimental Results

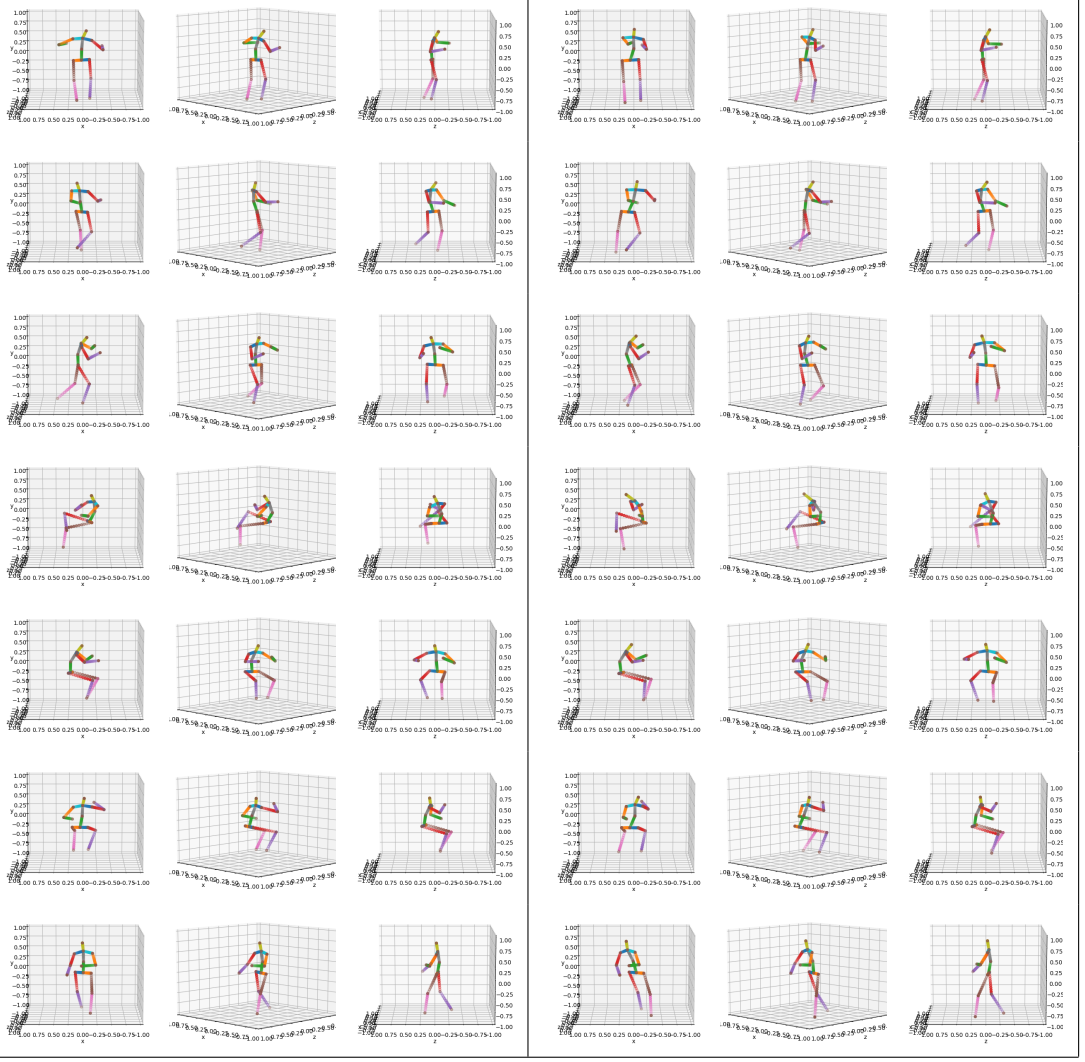
Ground Truth			Predicted		
Side	Diagonal	Front	Side	Diagonal	Front
					

Table C.1: More examples of pose lifting outputs when using self-consistency loss.

Ground Truth			Predicted		
Side	Diagonal	Front	Side	Diagonal	Front

Table C.2: More examples of pose lifting outputs when using self-consistency loss.

C.2 End-of-limb Keypoint Detection

C.2.1 2D Stickmen

This section includes further experiment results of the toy 2D stickmen keypoint detection where points are encouraged to find the ends of limbs via use of the LVMST algorithm and a differentiable sketching reconstruction task.

C.2.1.1 Further Experimental Results

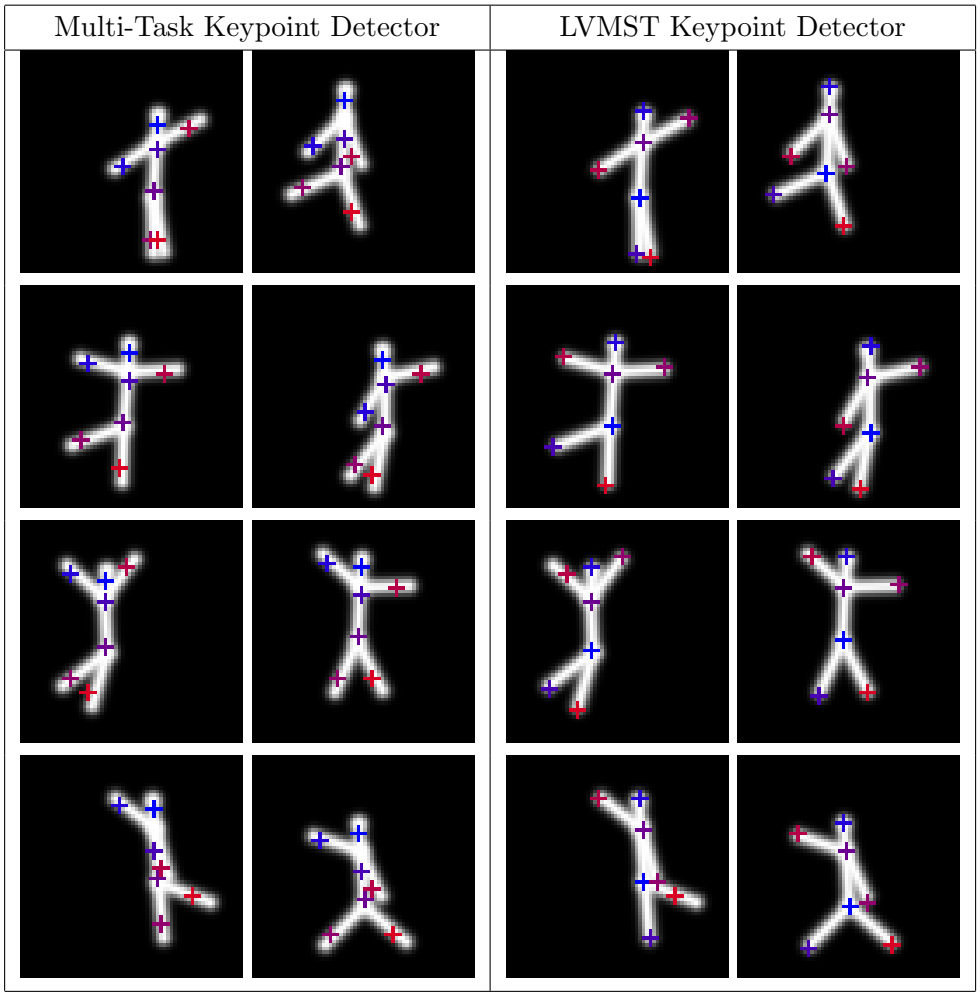


Table C.3: More examples of comparisons of placement of keypoints between Multi-Task generalised keypoint detector and LVMST with Differentiable Sketching on the artificial 2D stickman dataset.

C.2.2 Human3.6m Stickmen

This section includes further experiment results of the Human3.6m stickmen keypoint detection where points are encouraged to find the ends of limbs via use of the LVMST algorithm and a differentiable sketching reconstruction task.

C.2.2.1 Further Experiment Results

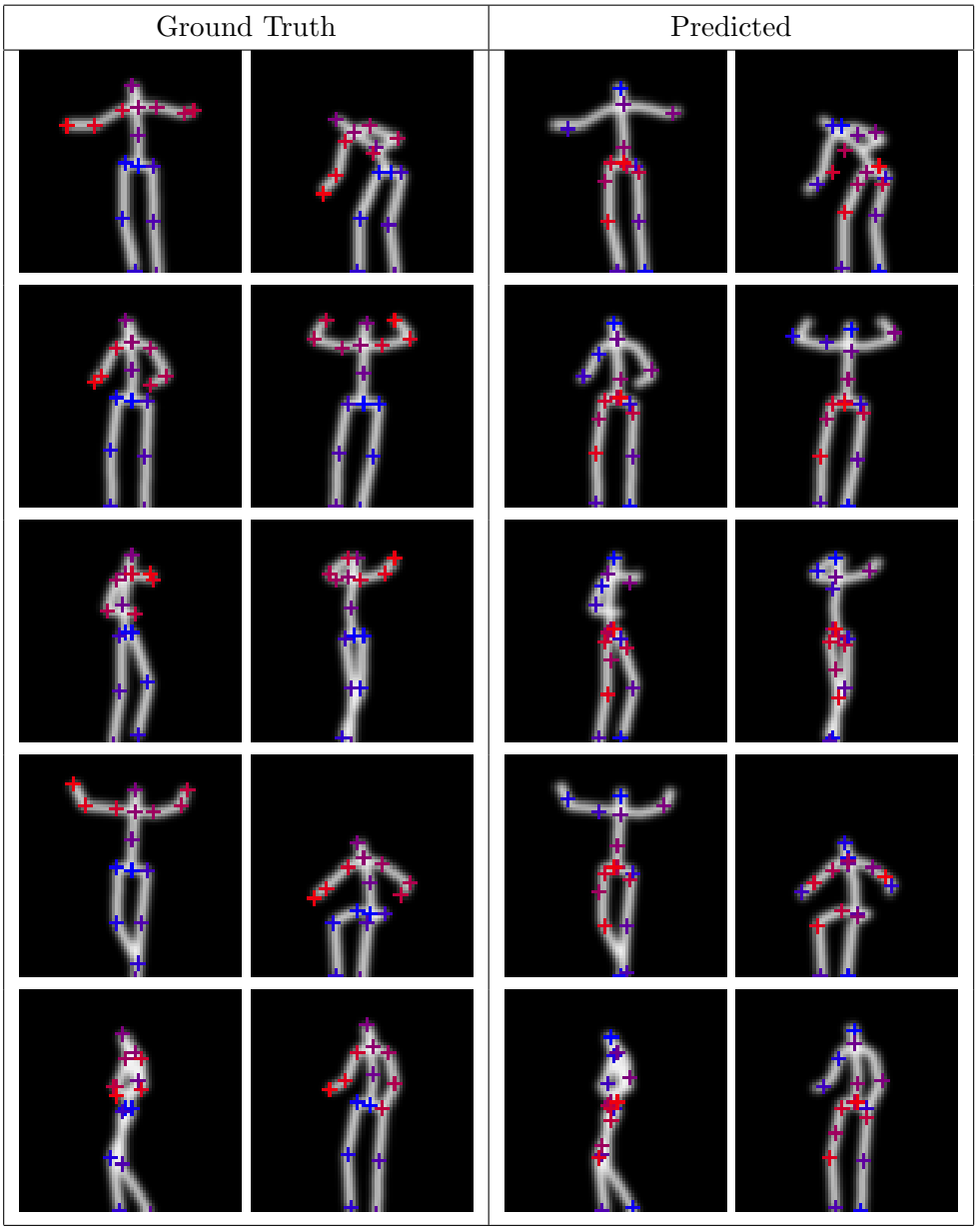


Table C.4: Additional comparison of placement of keypoints between in the ground truth Human3.6m stickmen dataset and the differentiable sketching approach at placing keypoints.

Appendix D

This appendix contains further details for the implementations of experiments in Chapter 6, along with further qualitative results.

D.1 Differentiable Sketching with Adversarial Lifting

D.1.1 Depth Estimation Network Architecture

The depth estimation network we use is a sequential network with layers defined below:

Layer Type	Input Size	Output Size	Activation
Linear	$2k$	128	ReLU
Linear	128	64	ReLU
Linear	64	32	ReLU
Linear	32	k	ReLU

Where k is set as the number of keypoints.

D.1.2 Hyper-Parameters

Here we outline the hyper-parameters used for this experiment. We learn 16 keypoints from 32x32 images in batches of 64. We optimise using two Adam optimisers, one for the generator and one for the discriminator, both using learning rate of 1e-3. The discriminator is updated once every 5 iterations, whereas the generation is updated every iteration. We use exponential function loss balancing (section 4.4.2.2) for our losses where the reconstruction loss has an additional multiplier of 7500.

D.1.3 Further Experiment Results

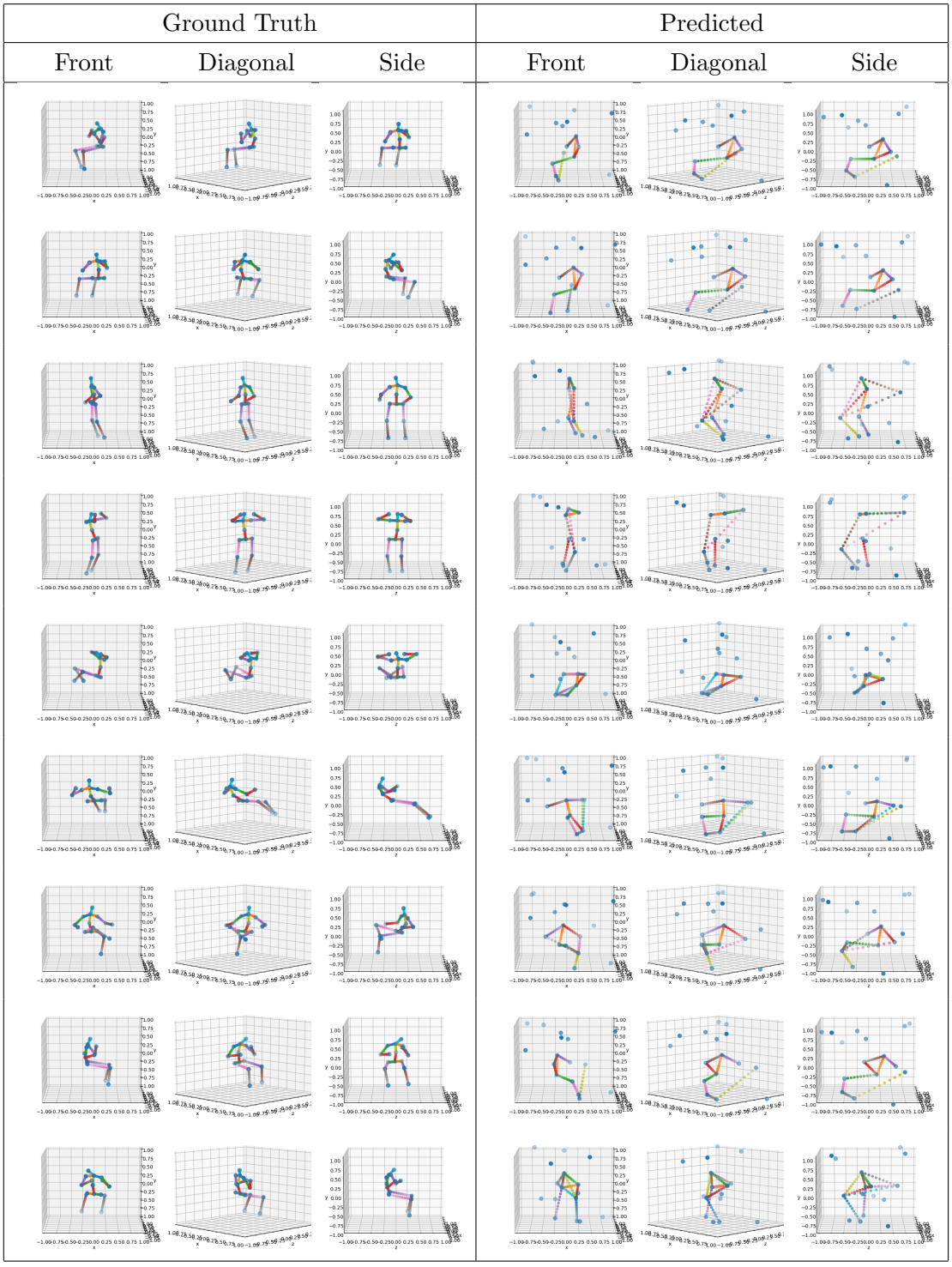


Figure D.1: Further demonstration of 3D keypoints found when concurrently learning to place keypoints using differentiable sketching and lift them into three dimensions using adversarial learning.

D.2 Full Pipeline with Rigid Bones Prior

D.2.1 Further Experiment Results

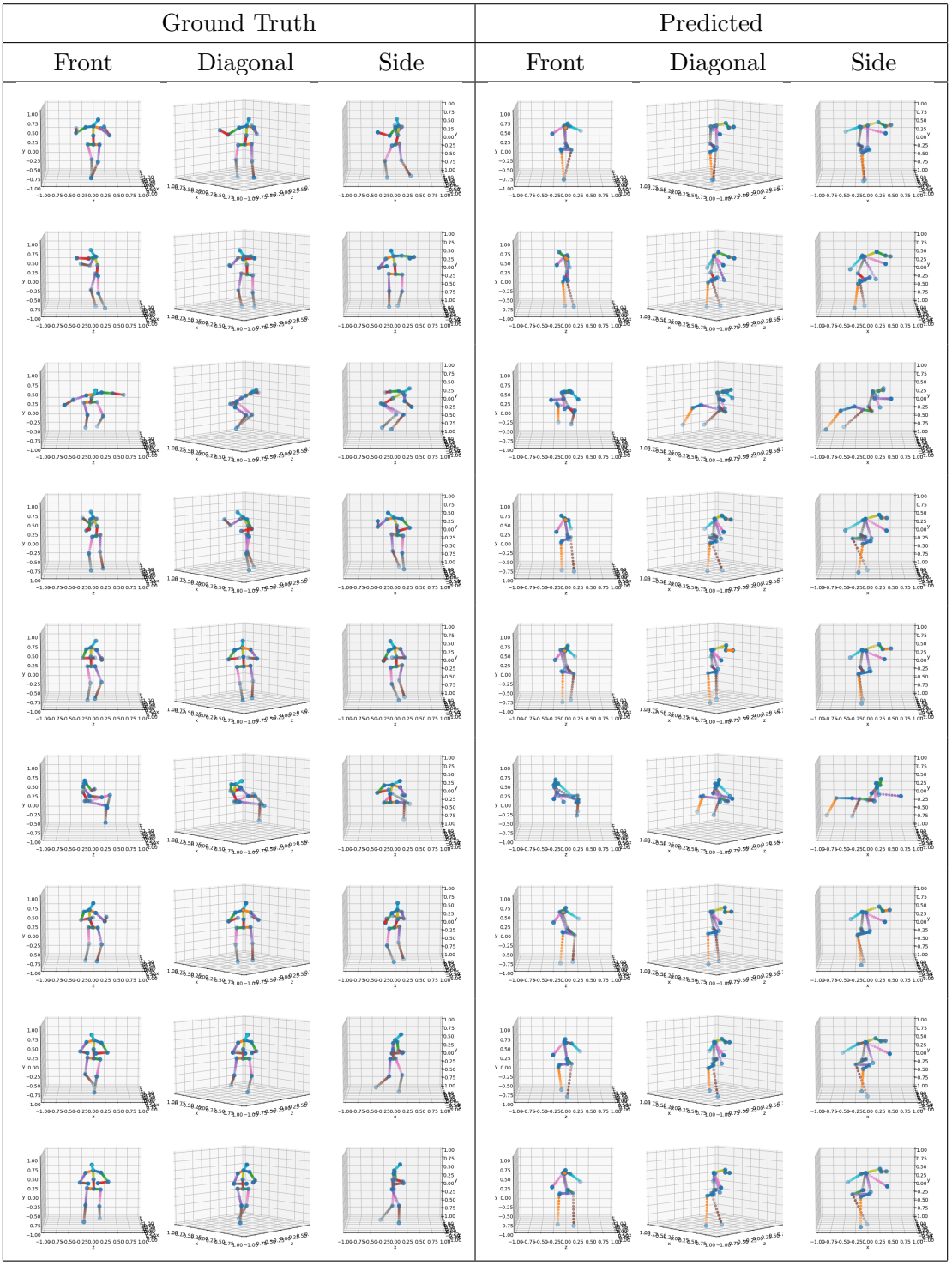


Table D.1: More examples of 3D keypoints found when concurrently learning to place keypoints and lift them into three dimensions using our rigid bones prior.

D.3 Split Pipeline with Rigid Bones Prior

D.3.1 Further Experiment Results

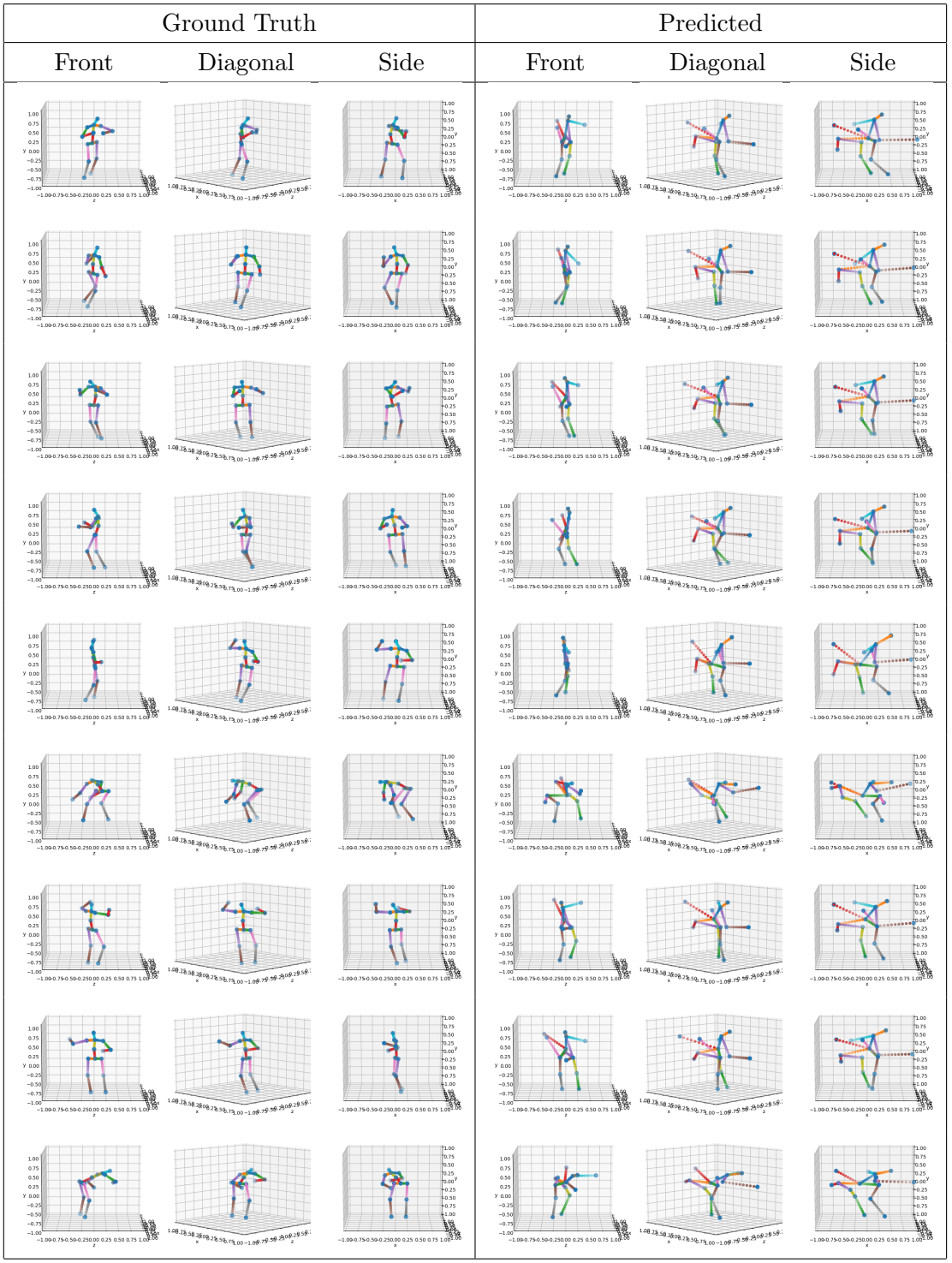


Table D.2: More examples of 3D keypoints found when concurrently learning to place keypoints and lift them into three dimensions using our rigid bones prior.

Bibliography

[1] Mathieu Aubry, Daniel Maturana, Alexei A. Efros, Bryan C. Russell, and Josef Sivic. Seeing 3D chairs: Exemplar part-based 2D-3D alignment using a large dataset of CAD models. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3762–3769, 2014. ISSN 10636919.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013. ISSN 01628828.

[3] David Bojanić, Kristijan Bartol, Tomislav Pribanić, Tomislav Petković, Yago Diez Donoso, and Joaquim Salvi Mas. On the comparison of classic and deep keypoint detector and descriptor methods. In *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 64–69, 2019.

[4] Arij Bouazizi, Julian Wiederer, Ulrich Kressel, and Vasileios Belagiannis. Self-Supervised 3D Human Pose Estimation with Multiple-View Geometry. *Proceedings - 2021 16th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2021*, 2021.

[5] Lubomir Bourdev and Jitendra Malik. Poselets: Body Part Detectors Trained Using 3D Human Pose Annotations. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1365–1372, 2009. ISSN 15505499.

[6] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2016. ISSN 10636919.

[7] Angela Caunce, David Cristinacce, Chris Taylor, and Tim Cootes. Locating facial features and pose estimation using a 3d shape model. In *International Symposium on Visual Computing*, pages 750–761. Springer, 2009.

[8] Angela Caunce, Chris Taylor, and Tim Cootes. Using detailed independent 3d sub-models to improve facial feature localisation and pose estimation. *International Journal on Artificial Intelligence Tools*, 22(06):1360017, 2013.

[9] Angela Caunce, Christopher Taylor, and Timothy F. Cootes. Improved 3d model search for facial feature location and pose estimation in 2d images. *10.5244/C.24.81*, pages 1–10, 09 2010.

[10] James Charles, Tomas Pfister, Mark Everingham, and Andrew Zisserman. Automatic and efficient human pose estimation for sign language videos. *International Journal of Computer Vision*, 110(1):70–90, 2014.

[11] Ching-Hang Chen and Deva Ramanan. 3d human pose estimation = 2d pose estimation + matching. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5759–5767, 2017.

[12] Ching Hang Chen, Ambrish Tyagi, Amit Agrawal, Dylan Drover, Rohith Mv, Stefan Stojanov, and James M. Rehg. Unsupervised 3D pose estimation with geometric self-supervision. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June, 2019. ISBN 9781728132938.

[13] Ching Hang Chen, Ambrish Tyagi, Amit Agrawal, Dylan Drover, Rohith Mv, Stefan Stojanov, and James M. Rehg. Unsupervised 3D pose estimation with geometric self-supervision. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June, 2019. ISBN 9781728132938.

[14] Xipeng Chen, Kwan-Yee Lin, Wentao Liu, Chen Qian, and Liang Lin. Weakly-Supervised Discovery of Geometry-Aware Representation for 3D Human Pose Estimation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[15] Yucheng Chen, Yingli Tian, and Mingyi He. Monocular human pose estimation: A survey of deep learning-based methods. *Computer Vision and Image Understanding*, 192(August 2019):102897, 2020. ISSN 1090235X.

[16] Yongju Cho, Dojin Kim, Saleh Saeed, Muhammad Umer Kakli, Soon Heung Jung, Jeongil Seo, and Unsang Park. Keypoint Detection Using Higher Order Laplacian of Gaussian. *IEEE Access*, 8:10416–10425, 2020. ISSN 21693536.

[17] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F Stewart, Jimeng Sun, and Sutter Health. RotNet unsupervised learning. *International Conference on Learning Representations*, 2018(2016):1–14, 2017.

[18] Peter Hviid Christiansen, Mikkel Fly Kragh, Yury Brodskiy, and Henrik Karstoft. Unsuperpoint: End-to-end unsupervised interest point detector and descriptor. *arXiv preprint arXiv:1907.04011*, 2019.

[19] Roberto Cipolla, Yarin Gal, and Alex Kendall. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018.

[20] Tim Cootes, Christopher J. Taylor, David H. Cooper, and Jim Graham. Active shape models-their training and application. *Comput. Vis. Image Underst.*, 61: 38–59, 1995.

[21] Ameya Daigavane, Balaraman Ravindran, and Gaurav Aggarwal. Understanding convolutions on graphs. *Distill*, 2021. https://distill.pub/2021/understanding-gnns.

[22] Qi Dang, Jianqin Yin, Bin Wang, and Wenqing Zheng. Deep learning based 2D human pose estimation: A survey. *Tsinghua Science and Technology*, 24(6):663–676, 2019. ISSN 18787606.

[23] Boyang Deng, JP Lewis, Timothy Jeruzalski, Gerard Pons-Moll, Geoffrey Hinton, Mohammad Norouzi, and Andrea Tagliasacchi. NASA: Neural Articulated Shape Approximation. *European Conference on Computer Vision*, 2019.

[24] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018.

[25] Carl Doersch and Andrew Zisserman. Multi-task Self-Supervised Visual Learning. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:2070–2079, 2017. ISSN 15505499.

[26] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[27] Dylan Drover, M. V. Rohith, Ching Hang Chen, Amit Agrawal, Ambrish Tyagi, and Cong Phuoc Huynh. Can 3D pose be learned from 2D projections alone? *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11132 LNCS:78–94, 2019. ISSN 16113349.

[28] Jean Duchon. Splines minimizing rotation-invariant semi-norms in sobolev spaces. In Walter Schempp and Karl Zeller, editors, *Constructive Theory of Functions of Several Variables*, pages 85–100, Berlin, Heidelberg, 1977. Springer Berlin Heidelberg. ISBN 978-3-540-37496-1.

[29] Zeyu Feng, Chang Xu, and Dacheng Tao. Self-supervised representation learning by rotation feature decoupling. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:10356–10366, 2019. ISSN 10636919.

[30] Martin A. Fischler and Robert A. Elschlager. The Representation and Matching of Pictorial Structures Representation. *IEEE Transactions on Computers*, C-22 (1):67–92, 1973. ISSN 00189340.

[31] Yoav Freund and Robert E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. ISSN 00220000.

[32] Pierre F. Gabriel, Jacques G. Verly, Justus H. Piater, and André Genon. The state of the art in multiple object tracking under occlusion in video sequences. In *In Advanced Concepts for Intelligent Vision Systems (ACIVS), 2003*, pages 166–173, 2003.

[33] Saul I. Gass and Michael C. Fu, editors. *Prim's Algorithm*, pages 1160–1160. Springer US, Boston, MA, 2013. ISBN 978-1-4419-1153-7.

[34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[35] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[36] Jacob M. Graving, Daniel Chae, Hemal Naik, Liang Li, Benjamin Koger, Blair R. Costelloe, and Iain D. Couzin. Deepposekit, a software toolkit for fast and robust animal pose estimation using deep learning. *eLife*, 8:1–42, 2019. ISSN 2050084X.

[37] Meng Hao Guo, Jun Xiong Cai, Zheng Ning Liu, Tai Jiang Mu, Ralph R. Martin, and Shi Min Hu. PCT: Point cloud transformer. *Computational Visual Media*, 7 (2):187–199, 2021. ISSN 20960662.

[38] Christopher G Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[39] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[40] Nils Hasler, Carsten Stoll, Martin Sunkel, Bodo Rosenhahn, and H-P Seidel. A statistical model of human pose and body shape. In *Computer graphics forum*, volume 28, pages 337–346. Wiley Online Library, 2009.

[41] Kaveh Hassani and Mike Haley. Unsupervised multi-task feature learning on point clouds. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-October:8159–8170, 2019. ISSN 15505499.

[42] Serhii Havrylov and Ivan Titov. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. *Advances in Neural Information Processing Systems*, 2017-December:2150–2160, 2017. ISSN 10495258.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[44] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty. *Advances in Neural Information Processing Systems*, 2019.

[45] Le Hui, Jia Yuan, Mingmei Cheng, Jin Xie, Xiaoya Zhang, and Jian Yang. Superpoint network for point cloud oversegmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5510–5519, October 2021.

[46] Tom Huysmans, Bart Haex, Tom De Wilde, Remy Van Audekercke, Jos Vander Sloten, and Georges Van der Perre. A 3d active shape model for the evaluation of the alignment of the spine during sleeping. *Gait & posture*, 24(1):54–61, 2006.

[47] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[48] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, 2014. ISSN 01628828.

[49] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, jul 2014.

[50] Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Learning Human Pose from Unaligned Data through Image Translation. Technical report, University of Oxford, 2019.

[51] Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Self-supervised Learning of Interpretable Keypoints from Unlabelled Videos. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 8787–8797, 2020.

[52] Anand Singh Jalal and Vrijendra Singh. The state-of-the-Art in visual object tracking. *Informatica (Slovenia)*, 36(3):227–248, 2012. ISSN 03505596.

[53] Simon Jenni and Paolo Favaro. Self-supervised Multi-view Synchronization Learning for 3D Pose Estimation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12626 LNCS:170–187, 2021. ISSN 16113349.

[54] Xiaopeng Ji, Qi Fang, Junting Dong, Qing Shuai, Wen Jiang, and Xiaowei Zhou. A survey on monocular 3D human pose estimation. *Virtual Reality and Intelligent Hardware*, 2(6):471–500, 2020. ISSN 26661209.

[55] Yutong Jiang, Xiaofei Liu, Dongyuan Wu, and Pengbiao Zhao. Residual Deep Monocular 3D Human Pose Estimation using CVAE synthetic data. *Journal of Physics: Conference Series*, 1873(1), 2021. ISSN 17426596.

[56] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):4037–4058, 2020.

[57] Sam Johnson and Mark Everingham. Clustered pose and nonlinear appearance models for human pose estimation. In *bmvc*, number 4 in 2, page 5. Aberystwyth, UK, 2010.

[58] Sam Johnson and Mark Everingham. Learning effective human pose estimation from inaccurate annotation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1465–1472, 2011. ISSN 10636919.

[59] Muhammed Kocabas, Salih Karagoz, and Emre Akbas. Self-supervised learning of 3D human pose using multi-view geometry. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:1077–1086, 2019. ISSN 10636919.

[60] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:1920–1929, 2019. ISSN 10636919.

[61] Yasunori Kudo, Keisuke Ogaki, Yusuke Matsui, and Yuri Odagiri. Unsupervised adversarial learning of 3d human pose from 2d joint locations. *arXiv preprint arXiv:1803.08244*, 2018.

[62] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[63] Tejas D Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. In *Advances in Neural Information Processing Systems*, volume 32, pages 10724–10734, 2019.

[64] Yann LeCun. The MNIST database of handwritten digits., 1998.

[65] Shichao Li, Lei Ke, Kevin Pratama, Yu Wing Tai, Chi Keung Tang, and Kwang Ting Cheng. Cascaded Deep Monocular 3D Human Pose Estimation with

Evolutionary Training Data. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 6172–6182, 2020. ISSN 10636919.

[66] Tzu Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics*, 39(6), 2020. ISSN 15577368.

[67] Yang Li, Kan Li, Shuai Jiang, Ziyue Zhang, Congzhentao Huang, and Richard Yi Da Xu. Geometry-driven self-supervised method for 3d human pose estimation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11442–11449, Apr. 2020.

[68] Sicong Liang and Yu Zhang. A simple general approach to balance task difficulty in multi-task learning. *arXiv preprint arXiv:2002.04792*, 2020.

[69] Yuanqing Lin, F Lv, S Zhu, M Yang, T Cour, K Yu, L Cao, Z Li, MH Tsai, X Zhou, et al. Imagenet classification: fast descriptor coding and large-scale svm training. *Large scale visual recognition challenge*, 2010.

[70] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the CoordConv solution. *Advances in Neural Information Processing Systems*, 2018-December(NeurIPS):9605–9616, 2018. ISSN 10495258.

[71] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:7707–7716, 2019. ISSN 15505499.

[72] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[73] Xiao Le Liu, Si Yang Yu, Nico A. Flierman, Sebastián Loyola, Maarten Kamermans, Tycho M. Hoogland, and Chris I. De Zeeuw. OptiFlex: Multi-Frame Animal Pose Estimation Combining Deep Learning With Optical Flow. *Frontiers in Cellular Neuroscience*, 15(May):1–14, 2021. ISSN 16625102.

[74] Zhao Liu, Jianke Zhu, Jiajun Bu, and Chun Chen. A survey of human pose estimation: The body parts parsing based methods. *Journal of Visual Communication and Image Representation*, 32:10–19, 2015. ISSN 10959076.

[75] Andreas Loukas. What graph neural networks cannot learn: depth vs width. *arXiv preprint arXiv:1907.03199*, 2019.

[76] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[77] Diogo C. Luvizon, Hedi Tabia, and David Picard. Human pose regression by combining indirect part detection and contextual information. *Computers and Graphics (Pergamon)*, 85(November 2017):15–22, 2019. ISSN 00978493.

[78] Petros Maragos. *PDEs for Morphological Scale Spaces and Eikonal Applications*. Elsevier Inc., second edition edition, 2005. ISBN 9780121197926.

[79] Julieta Martinez, Rayat Hossain, Javier Romero, and James J Little. A simple yet effective baseline for 3d human pose estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 2640–2649, 2017.

[80] Daniela Mihai and Jonathon Hare. Differentiable drawing and sketching. *arXiv preprint arXiv:2103.16194*, 2021.

[81] Daniela Mihai and Jonathon Hare. Learning to draw: Emergent communication through sketching. *Advances in Neural Information Processing Systems*, 34:7153–7166, 2021.

[82] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60:63–86, 10 2004.

[83] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9905 LNCS:527–544, 2016. ISSN 16113349.

[84] Hans P. Moravec. Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover, Tech. Report, Robotics Institute, Carnegie-Mellon University, pp.1-175. *Technical Report CMU-RI-TR-80-03*, pages 1–175, 1980.

[85] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. *Lecture Notes in Computer Science*, page 483–499, 2016. ISSN 1611-3349.

[86] Markus Oberweger, Mahdi Rad, and Vincent Lepetit. Making deep heatmaps robust to partial occlusions for 3D object pose estimation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11219 LNCS:125–141, 2018. ISSN 16113349.

[87] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[88] Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. Learning to Estimate 3D Human Pose and Shape from a Single Color Image. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018. ISBN 9781538664209.

[89] Talmo D. Pereira, Diego E. Aldarondo, Lindsay Willmore, Mikhail Kislin, Samuel S.H. Wang, Mala Murthy, and Joshua W. Shaevitz. Fast animal pose estimation using deep neural networks. *Nature Methods*, 16(1):117–125, 2019. ISSN 15487105.

[90] Yaadhav Raaj. Exploring pose priors for human pose estimation with joint angle representations. *arXiv preprint arXiv:1909.12761*, 2019.

[91] N Dinesh Reddy, Minh Vo, and Srinivasa G Narasimhan. Occlusion-net: 2d/3d occluded keypoint localization using graph networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7326–7335, 2019.

[92] Grégory Rogez, Carlos Orrite-Urunuela, and Jesús Martínez-del Rincón. A spatio-temporal 2d-models framework for human pose recovery in monocular sequences. *Pattern Recognition*, 41(9):2926–2944, 2008.

[93] Ben Sapp and Ben Taskar. Modec: Multimodal decomposable models for human pose estimation. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3681, 2013.

[94] Cordelia Schmid and Roger Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–535, 1997. ISSN 01628828.

[95] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

[96] Trevor Standley, Amir R. Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which Tasks Should Be Learned Together in Multi-task Learning? *International Conference on Machine Learning*, 2019.

[97] Richard Szeliski. Image alignment and stitching. In *Handbook of mathematical models in computer vision*, pages 273–292. Springer, 2006.

[98] James Thewlis, Samuel Albanie, Hakan Bilen, and Andrea Vedaldi. Unsupervised Learning of Landmarks by Descriptor Vector Exchange. *Proceedings of the IEEE International Conference on Computer Vision*, aug 2019.

[99] James Thewlis, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object frames by dense equivariant image labelling. *Advances in neural information processing systems*, 30, 2017.

[100] James Thewlis, Hakan Bilen, and Andrea Vedaldi. Unsupervised Learning of Object Landmarks by Factorized Spatial Embeddings. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:3229–3238, 2017. ISSN 15505499.

[101] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *International Conference on Learning Representations*, 2020.

[102] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[103] Yao-Hung Hubert Tsai, Yue Wu, Ruslan Salakhutdinov, and Louis-Philippe Morency. Self-supervised learning from a multi-view perspective. *arXiv preprint arXiv:2006.05576*, 2020.

[104] Timo Von Marcard, Roberto Henschel, Michael Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *European Conference on Computer Vision (ECCV)*, sep 2018.

[105] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. Self-supervised 3D hand pose estimation through training by fitting. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:10845–10854, 2019. ISSN 10636919.

[106] Bastian Wandt, Marco Rudolph, Petrissa Zell, Helge Rhodin, and Bodo Rosenhahn. Canonpose: Self-supervised monocular 3D human pose estimation in the wild. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 13289–13299, 2021. ISSN 10636919.

[107] Jinbao Wang, Shujie Tan, Xiantong Zhen, Shuo Xu, Feng Zheng, Zhenyu He, and Ling Shao. Deep 3D human pose estimation: A review. *Computer Vision and Image Understanding*, 210(August 2020):103225, 2021. ISSN 1090235X.

[108] Gerald Westheimer. Illusions in the spatial sense of the eye: Geometrical–optical illusions and the neural representation of space. *Vision Research*, 48(20):2128–2142, 2008. ISSN 0042-6989. Vision Research Reviews.

[109] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2020.

[110] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[111] Wei Yang, Wanli Ouyang, Xiaolong Wang, Jimmy Ren, Hongsheng Li, and Xiaogang Wang. 3D Human Pose Estimation in the Wild by Adversarial Learning. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 5255–5264, 2018. ISSN 10636919.

[112] Aron Yu and Kristen Grauman. Fine-grained visual comparisons with local learning. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 192–199, 2014. ISSN 10636919.

[113] Hang Yu, Yufei Xu, Jing Zhang, Wei Zhao, Ziyu Guan, and Dacheng Tao. Ap-10k: A benchmark for animal pose estimation in the wild. *arXiv preprint arXiv:2108.12617*, 2021.

[114] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*, 2020.

[115] Yan Zhang, Jonathon Hare, and Adam Prügel-Bennett. Deep set prediction networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[116] Yu Zhang and Qiang Yang. An overview of multi-task learning. *National Science Review*, 5(1):30–43, 2018.

[117] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[118] Yuting Zhang, Yijie Guo, Yixin Jin, Yijun Luo, Zhiyuan He, and Honglak Lee. Unsupervised discovery of object landmarks as structural representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2694–2703, 2018.

[119] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial Landmark Detection by Deep Multi-task Learning. *European Conference on Computer Vision*, 2014.

[120] Silvia Zuffi, Angjoo Kanazawa, David Jacobs, and Michael J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:5524–5532, 2017.