Multi Stage Convolutional Neural Networks for 3D Object Structure Estimation From a Single Image

Name: Andrea Casino

Supervisor: Dr. Lourdes Agapito

20 August 2017

This report is submitted as part requirement for the MSc Degree in 'Computer Graphics, Vision & Imaging', at University College London. It is substantially the result of my own work except where explicitly indicated in the text.

Abstract

The goal of this project was to design a system can identify the 3D structure of an object from a single RGB image. For the purpose of this evaluation, images of chairs were used. The system was designed to estimate the object structure by identifying a series of fundamental keypoints within it, and lifting them into their respective 3D coordinates. The system consisted of two key sections: a multi stage convolutional neural network for keypoint extraction and a fully connected neural network to find the depth for the estimated 2D keypoint coordinates. The first section was designed as an extension of the keypoint estimator found in Tome et al.[1] as well as in Wei et al.[2], which is used for human keypoint identification. The fully connected network was an implementation of the one found in Srivastava et al.[3].

During the evaluation of the system, the trained keypoint extractor was able to estimate the coordinates of the keypoints on the object. Unfortunately, the system was not able to differentiate between different keypoints on the same level of the chair (upper back, seat and legs). However, the project proved that there can be great improvements to an output if a multi stage achitecture is used in convolutional neural networks instead of a single stage architecture. In addition, the depth estimator was successfully implemented with positive results. Finally, though it was not able to extract specific keypoints, the combined system was able to output a 3D model from an image input. This was done under the circumstance where the image portrays the chair facing the camera. Under these requirements, the system is able to exploit the known relationships between the object and its frame to successfully estimate which keypoint belonged to which appendage. This was done with successful results.

Acknowledgements

I would like to give special thanks to Professor Lourdes Agapito and Denis Tome, both of whom provided incredibly beneficial help throughout the course of this project. I would also like to thank Martin Runz for allowing me to use his machine for training. Finally I would like to thank my Family, (Mamma, Papà, Alessandro, Claudia, Becky and Bruno) who's constant support is always appreciated.

Contents

1 Introduction

The process of classifying images has improved greatly in the past years, largely due to the rise in popularity of convolutional neural networks. These allow the achievement of lower error rates when identifying subjects in many challenging data sets [4]. A logical extension of this field of research is the ability to classify an object in an image, and identify its structure in three dimensions. This has been done in the context of human pose estimation [5][2][1] as well as that of object structure estimation [6][7][8].

Capturing data for structure estimation of an object can be accomplished in many ways, including the use of depth cameras and infra-red sensors[9]. Although these techniques allow for more data to be captured, leading to more precise results, a large amount of research is being devoted to the use of a single rgb camera for these tasks. One of the reasons for this is to allow widespread utilization of the resulting technology, as rgb cameras are already predominantly used within the field of photography. Using a single rgb camera as a research focus gives pre-existing hardware the ability to use the software produced, as well as decreasing the need for any new hardware. In addition, due to the increase in smart phone adoption, rgb cameras are commonplace in the developed world, leading to a higher amount of possible users. Finally, a single camera set up leads to less physical weight, which is important in the field of robotics, head mounted hardware and space travel.

The purpose of this research project was to design a system that, using a single rgb image, can identify the 3D structure of an object within said image. The 3D structure is represented in the form of a skeleton, which is found by identifying the keypoints of an object and lifting them to their respective coordinates in 3D space (Figure 1) . The use of a skeleton was chosen over the use of any other 3D representation because it is able to "preserve the structural properties that we are interested in"[6], such as the object's dimensions and style, without having to store or process redundant information.

Figure 1: Example output of the desired system

The designed system consists of two key sections. The first is a multi stage convolutional neural network, used to find the keypoints of an object. The second is a fully connected neural network to use the identified keypoints and lift them into 3D. The architecture of the keypoint identification system was based on the pre-existing CNN used in both "Lifting from the Deep: Convolutional 3D Pose Estimation from a Single Image"[1] and in "Convolutional Pose Machines"[2] which is used for the estimation of human keypoints. This was modified to include the keypoint identification of furniture which is found in the keypoint-5 training data by MIT [6]. The data set includes images of chairs, beds and sofas along with the coordinates of each of their respective key points.

The purpose of the keypoint localization system used in Tome et al.[1] and in Wei et al.[2] is to find the defining keypoints of a human from a single RGB image. These are identified by their shape, color, angle and location within the image. Similarly, the system being designed must derive the defining keypoints of a piece of furniture using the same characteristics. Therefore, the existing model was adapted to include the identification of furniture keypoints while keeping the

same multi stage network structure. The system estimates keypoints through the generation of heatmaps. A heatmap is a single channel image that for every pixel cointains a value corresponding to the probability that a given keypoint is situated in it. It can then be assumed that the exact position of the keypoint resides at the coordinates with the highest value.

The second network can then use the 2D coordinates to estimate the object's 3D structure by structure by lifting 2D keypoints into 3D. The network used for this purpose was designed in "'A Simple, Fast and Highly-Accurate Algorithm to Recover 3D Shape from 2D Landmarks on a Single Image"[3]. This provides a fast and accurate way to regress the depth of keypoints for a given object. To train the network, sets of image keypoints are used along with their respective 3D coordinates. These were gathered using syntetic furniture models[10] that were projected onto a plane, while using data from the models themselves as the output. This simulates a photograph of the 3D model, which can then be used to regress its depth.

The main difficulties of the project resided in the keypoint localisation network. This included the fine tuning of the network parameters, input data and ground truth data. Seeing as neural networks are very sensitive to slight changes in their system, it was found that constants such as the input image size and the ground truth heatmap value range had a substantial inpact on the results. It was also confirmed that the multi stage nature of the keypoint localization neural network greatly improved the results when being compared to a single stage system.

In the following sections, related work is discussed to give context to the project. The architecture and implementation of the system are also discussed, results it provided are evaluated.

2 Related Work

2.1 Convolutional Neural Networks

A neural network is a type of supervised machine learning system, initially designed to emulate a human brain's synaptic connections [11]. A network system consists of many nodes called neurons, which apply a summation function and an activation function to a set of weighted inputs to produce an output(figure 2). The activation function depends on the given purpose of the network, and can range from a sigmoid function (equation 1) to a rectifier linear unit function (equation 2) and more. Sets of neurons are arranged in layers in which the weighted outputs of one layer will be the inputs of the next. Once trained, a piece of data is fed into the first layer of neurons, called the "input layer". The data is processed through the network using "hidden layers", and a desired result is obtained from the last layer of neurons, called the "output layer".

$$\sigma(x) = \frac{1}{(1 + e^x)} \tag{1}$$

$$f(x) = max(0, x) \tag{2}$$

To create a neural network, its framework must first be designed. This includes selecting the activation functions within each neuron, how many layers of neurons to use and constructing a cost function to evaluate the final result. The purpose of the cost function is to maximize cost when the least desired errors are made. In addition, a regularization function is usually applied to the cost to diminish the possibility of it over fitting the test model by normalizing the impact of very high costs. Lastly, the amount of layers must also be carefully evaluated seeing as this changes the degrees of freedom of the system, which may lead to over fitting or under fitting. Over fitting refers to a situation in which the neural network is tailored too closely to the training data,

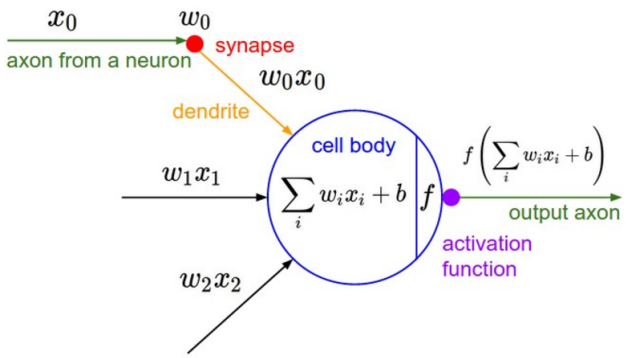

Figure 2: Visualisation of a neuron with three inputs [11]

and does not allow for more generalized inputs. For this reason, when testing, a separate set of data must always be used.

Once designed, the neural network can be trained through learning. The weights between the neurons are first initiated. Following this, the system is then trained by giving it a data input with a known output. A solution is then calculated, and is compared to the ideal output with the use of the the loss function. Once a loss value is calculated, through the process of back propagation, the weights within each neuron can be adjusted to minimize the error. This is done with the use of an optimizer, such as the adam algorithm [12] or gradient descent, which takes the derivative of the loss function in respect to the weights and iterates over it to find the local minimum cost. Once this process is has been completed, the old weights are updated to reflect the minimized cost function. The learning step demands the choice of multiple parameters. These include the step size between iterations when looking for the minimum and the learning rate, which indicates how much each local minimum contributes to updating the weights. To finalize the network, this process must be iterated over all the test data for a number of epochs.

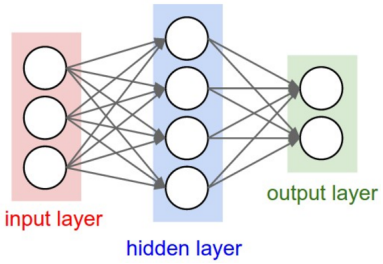

Figure 3: Basic visualization of a neural network [11]

Within the context of machine vision, convolutional neural networks are the most widely used type of neural networks. These are based on classic neural networks, but use specific properties within the layers which allow them to be optimized for image inputs. Most CNNs can be broken down into three sets of layers: convolution layers, pooling layers and ReLU layers[11]. Convolution layers use a scrolling filter and apply it to many bundles of pixels using convolution. This produces a "2-dimensional activation map that gives the responses of the filter at every spatial position"[11] . The size of this matrix depends on the size of the filter(n x n), and the filter's stride while scrolling. Pooling layers, on the other hand, take the convolution layer's output and re-size it into a smaller matrix of results. This is used to decrease the large amount of pixel data and to make over fitting less likely to occur. Finally, the ReLU layer is used to apply a rectifier linear unit function to its weighted input, which speeds up gradient descent by thresholding it at zero [13]. These are ideal for machine vision purposes as they allow to for the high dimensionality contained within images to be processed efficiently.

2.2 Keypoint Localisation

Convolutional neural networks are efficient systems for key point localization in images. An example of an efficient implementation of this is from the paper: “Efficient Object Localization Using Convolutional Networks” [5]. This research finds a state of the art approach for key point estimation from a single rgb image using convolutional neural networks. Firstly, the paper uses three levels of a gaussian pyramid as inputs to the network. This is done to analyze the images at three different frequencies which allows to identify details of various sizes. The images are then input into two identical neural networks with different input resolutions. The first layer of both networks is used for local contrast normalization, which highlights peaks within local areas of the image. This is done to make the following network layers more sensitive to high frequencies and less sensitive to uniform areas [13].

The result of the local contrast normalization layer is then input into a network which samples it in 7 different resolutions to identify the key point regardless of image size. This is done by passing the original resolution through a convolution layer, then through a ReLU layer, and finally through a pooling layer to be re-sized to the next resolution. Once all the resolutions have been processed, the outputs of all three neural networks (one for each member of the gaussian pyramid) then go through a pooling stage so their resolutions match. Before passing through a final convolution layer, a dropout layer is present. This is a regularization technique to avoid over fitting which randomly selects a few neurons and its associated synapses, and does not use them during some training stages. This has shown to improve neural networks by reducing over fitting and improving the performance of many types of neural networks [3]. Finally, a heat map for each key point is output, each denoting the probability that any of the given pixels is the relative key point being identified. Once trained, this system was able to identify key points on a human with a detection rate ranging from 64.8 percent to 96.1 percent.

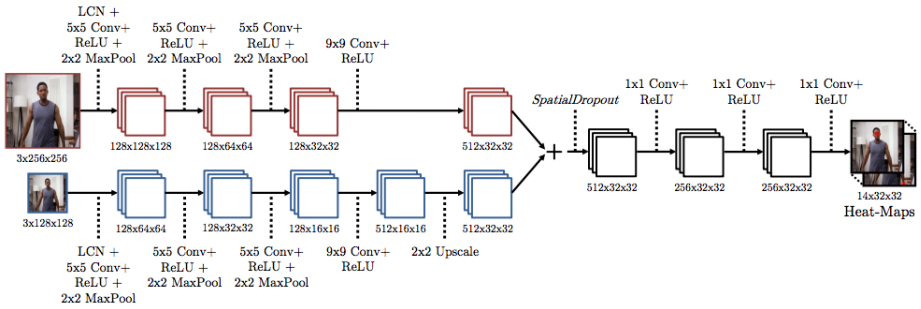

Figure 4: Diagram of neural network for keypoint localisation [5]

2.3 Multi Stage Networks for Keypoint Localisation

A multi stage network is an example of a convolutional neural network which is divided amongst stages. The first stage resembles a classic CNN that uses an image as its input and outputs a result. By contrast, in the second stage and every subsequent stage, the system consists of a neural network that takes the original image as an input, as well as the output from a previous stage. By going through the different stages, the result is refined, rendering the final output more accurate.

An example of a multi stage convolutional neural network being used to identify key points is from the paper "Convolutional Pose Machines" by Carnegie Mellon University [2]. This uses a single rgb image to output heat maps, identifying the location of given key point using a multi stage cnn. In the first stage, this system receives an input image and produces a series of

heat maps corresponding to the key points being identified. This is done by passing the image through five 9 x 9 convolution layers, with a pooling layer between each set of convolution layers. These are present so that the image can be processed at a variety of resolutions, allowing for the key points within the input image to not depend on size. In this stage, the convolution layers work by locally extracting features. This indicates that rather than simultaneously analyzing the entire image, local sections in the image are isolated. After this, the network consists of two 1 x 1 layers used to achieve a "fully convolutional architecture" [2]. This allows the network to apply global processing to the results once the image has already been processed locally [14] and, using the features extracted in the previous section, to identify the keypoints.

From the second stage onwards, every stage holds a network which uses the spacial context provided by the previous stage to refine the heat map results. This is done by firstly inputting the original image in a local convolutional neural network with four convolution layers and three pooling layers. This local output is then compared to the fully convolutional output of the previous stage. The system can then impose geometric constraints that apply to the object to refine the local heat maps using the context of the results from the previous stage. This refines the heat maps and increases their accuracies leading to "eliminating wrong and strengthening correct estimations on the belief maps"[2]. The result of this is then input into three 11 x 11 convolution layers. The large receptive field of these layers is used to detect long range correlations of key points within the image. The paper states that " the accuracy improves as the effective receptive field increases, and starts to saturate around 250 pixels" [2] allowing for higher result accuracy. In addition, the loss function implemented at the end of every stage is designed so that the difference between the produced heat map and the ideal heat map is minimised. Finally, the data can flow through multiple stages, each of which further refines the result. The research showed that, using the LSP human body pose dataset [15], the increase in accuracy reached a saturation level after four stages were implemented.

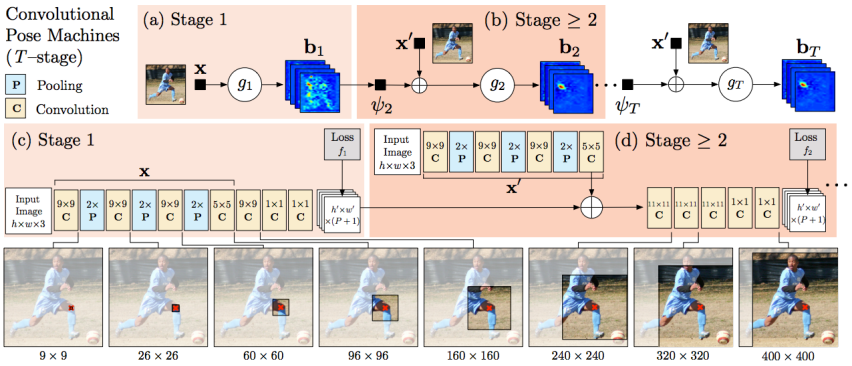

Figure 5: Diagram of multi stage neural network for key point estimation [2]

Another example of a multi stage network is the system this project is based on: “Lifting from the Deep: Convolutional 3D Pose Estimation from a Single Image” from D. Tome, C. Russell and L. Agapito [1]. Similarly to the previously discussed system, this one uses a single image to find heat maps corresponding to key points within it. In addition, this system uses the aid of 3D skeleton models to lift the key points into three dimensions, outputting a 3D pose estimation of the subject in the input image. To do this, firstly the input image is put through the first stage of a trained convolutional neural network. This is composed of a feature extractor used to highlight important features within the image, and a joint predictor used to identify the key points from the features. The feature extractor is made up of four convolution layers and three pooling layers, while the joint predictor is made up of three convolution layers. This outputs 18 heat maps, one for each joint (key point), including the head, shoulders and knees.

Following this, the heat maps are input into a trained probabilistic model for pose estimation. This uses existing 3D skeletons obtained using motion capture techniques, and corresponding 2D heat maps to output the 3D skeleton that most closely corresponds to the supplied heat maps.

The 3d skeleton is then projected back into two dimensional heat maps, after which the belief maps produced by the CNN and those produced by the probabilistic model are merged. The result of this is a set of 2D heat maps which come from the original image, but are given physical restraints by the probabilistic model, allowing for more accurate results. The second stage, and every stage after it, are then similarly structured to the first. They contain the same feature extractor, but the joint predictor takes as input both data from the feature extractor and the resulting heat maps from the previous stage. Like previously, this is done to give macroscopic context to the local heat maps to produce more accurate heat maps that take large correlations into account. The output of this is input into the probabilistic model, and the stage continues like the first. This continues for six layers, continuously refining the belief maps until a final set of heat maps is generated. The result is input into the probabilistic model once more, and its output is the final 3d pose. The system "outperforms all other methods in terms of average error showing a 4.7mm average improvement over our (their) closest competitor" [1].

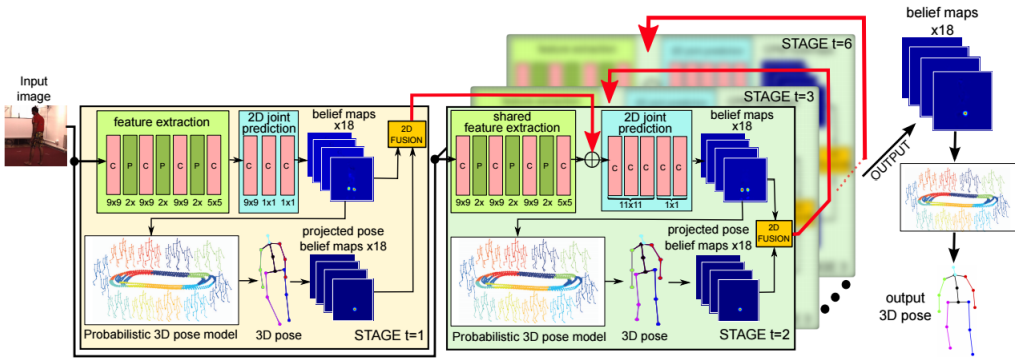

Figure 6: Diagram of multi stage neural network used for body pose estimation [1]

2.4 Object Interpretation

Thus far, papers discussed are examples of research utilising key point localization and reconstruction mostly within the realm of 3D human pose estimation. This is partially due to the fact that there is a large selection of available annotated images with corresponding 3D skeletons and models. Said models come from motion capture data using instruments specifically designed for humans. As the equivalent is not true of objects, it is harder to find data with which to train a neural network.

The paper "Single Image 3D Interpreter Network" by MIT [6] proposes a novel way of surpassing this problem. It discusses the use of a convolutional neural network with a single rgb image as an input to find the 3D skeleton structure of objects. It does so by utilizing annotated images of furniture along with a set of synthetic 3D models from the IKEA dataset [10].

The architecture of this system is a single stage network with the addition of a 3D interpreter to convert the 3D synthetic data into key point information. Firstly, the original image is put through an initial key point estimation based on the previously discussed paper [5]. This involves taking the image at three different resolutions, after which a local contrast normalization layer can be introduced. The three different resolutions are then input into identical networks which consist of a 5 x 5 convolution layer, a ReLU layer and a pooling layer followed by a 9 x 9 convolution layer and a second ReLU layer. A final pooling layer is then applied to the three branches of the network so that their outputs would all be of the same resolution. This outputs n many heat maps, corresponding to the amount of key points for the object.

The next part of the system consists of 3 network layers used for key point refinement. These layers use the known constraints of the object that is being searched for (found using the

synthetic data), and modify the key points accordingly to keep the object geometrically correct. An example of this is making sure that a chair's legs must be longer than the space between them, and if they are not, modifying those key points accordingly. This refines the heat maps, making them more accurate.

The heat maps are then input into a 3D interpreter network modeled using the synthetic data, which infers the final 3D skeleton corresponding to the input image. This is done by creating a model which approximates the 3D position of key points from 2D heat maps.

The model proposed by this research takes all the synthetic data of a given category (example: beds), and separates them into a base shape B_k and a specific model's internal parameters α_k. The base shape is a weighted sum of all the objects of the same category. This encodes macroscopic properties, such as that a bed's four legs are always the same length. The internal parameters on the other hand encode the specific properties of a given model, such as the precise length and with of the bed. A 3D model can then be projected into 2D key point coordinates, X, using the following equation. In this, P is a projection function, R is the rotation of the object, T is the translation of the object, K is the amount of objects in the data set, and α_k are the internal parameters of model K.

$$X = P(R\sum_{k=1}^{K}\alpha_k B_k + T) \tag{3}$$

The 3D interpreter network is then designed to minimize the difference between this function and the ideal output. The generated 3D key points are connected, and the resulting skeleton is output. Qualitative results show that this method allows for "2D keypoint estimation and 3D structure and viewpoint recovery, comparable to or better than the state-of-the-arts"[6].

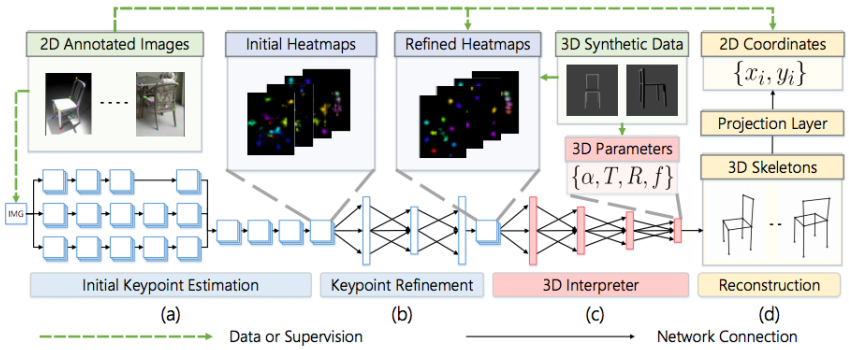

Figure 7: Diagram of neural network used for object structure estimation [1]

Another algorithm used for object interpretation is found in "A Simple, Fast and Highly-Accurate Algorithm to Recover 3D Shape from 2D Landmarks on a Single Image" [16]. The system takes a set of 2D coordinates, each of which correspond to a human keypoint for a given image. The coordinates are firstly normalized so that any image size can be reduced to a standardized. The normalized coordinates are input into a 6 layer, fully connected neural network which regresses the z coordinate so a 3D representation of the body's pose can be made (Figure 8).

The first 5 layers of the network have 2n neurons, where n is the amount of keypoints, as these need to process the x and y values for each keypoint. The last layer has n neurons, each of which corresponds to a keypoint's respective z coordinate. The activation function for each layer is a hyperbolic tangent function.

In addition, before the network, the system includes a layer dedicated to compensate

for missing 2D data and its respective depth. The recurrent layer[17] uses the available keypoint coordinates to regress the missing ones. The available and estimated coordinates are fed into the rest of the network.

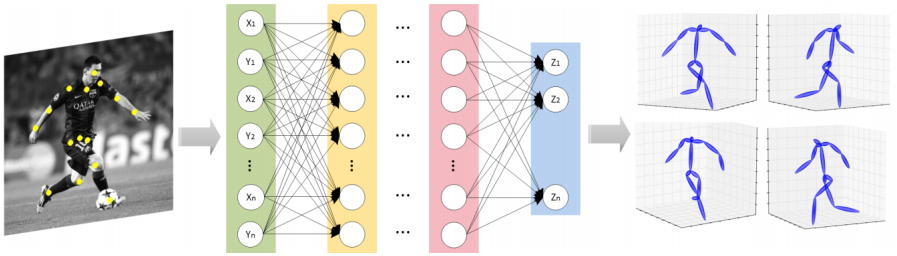

Figure 8: Diagram of fully connected neural network for z dimension regression [16]

3D human keypoint coordinates are used to train the network. Firstly, like the 2D input coordinates, the 3D coordinates are normalized to "eliminate the effect of scaling and translation of the 3D object"[16]. The normalized coordinates are then projected onto a 2D plane using a weak perspective camera model. Before doing so, the 3D keypoint coordinates are augmented to create more training data for the system. This involves translating the models, rotating them, changing the camera model's focal length and adding noise.

Finally, the system can take a set of 2D coordinates, which are first standardized. This outputs the z coordinates, which, when concatenated with the x and y, form a full set of 3D coordinates. The results of this system are shown to be "significantly more accurate than those given by previously defined algorithms"[16]. This is the case with multiple datasets that were tested, including human body poses, human face structures and car structures. In addition, the neural network runs very fast, allowing for faster than real time regression. In addition, the network also benefits from the fact it can be trained with a small sample of data.

3 System Architecture

3.1 Overview

The main goal of this research project is to expand the latest version of the previously discussed multi staged body pose estimation system found it Tome et al.[1] and Wei et al.[2] to include the structure and pose reconstruction of objects. In particular, the examined objects belong to the keypoint-5 data set[10], specifically the chair data set. The approach taken is to keep the multi staged nature of the convolutional neural network for keypoint identification (Figure 9) and applying the 3D interpreter from " Simple, Fast and Highly-Accurate Algorithm to Recover 3D Shape from 2D Landmarks on a Single Image" [6] to be able to lift 2D key points into 3D (Figure 10).

3.2 Keypoint Localization

The first layer of the keypoint localization system is composed of two sections. The first is a feature detector. This consists in a series of convolutional and pooling layers that are designed to extract image features such as corners and edges. By identifying the essential features in the image and dismissing the redundant ones, the system is able to estimate the keypoints with more clarity. The second section is used for keypoint prediction. This takes the output of the first section and, through a series of convolutional layers, outputs n+1 heatmaps, where n is the amount of

keypoints in the image. The first n heatmaps represent each keypoint of the object, while the n+1th heatmap is a combination of every keypoint present in the image.

Once the initial heat maps have been calculated, the input for the second stage consists of a concatenation of four sets of data. These are the output of the first stage, the original image, an intermediary convolutional layer from the of the keypoint estimator and a heatmap showing the position of the object being evaluated. The first 3 arrays are combined to give the network a wider set of data with which it can estimate the keypoints. The fourth set of data, the heatmap centered on the chair, is used to make the network ignore anything that does not fall within the position of the object.

This concatenation is fed into a second stage that is composed of a series of convolutional layers. The second stage is used to refine the keypoint position by analyzing the output of the first stage globally. A global approach puts each initial keypoint estimation in the context of every other keypoint estimation. By doing so, the system can eliminate the misclassification of the keypoints, leading to more precise results. Once again, the second stage outputs a set of heatmaps that corrispond to the estimated keypoint positions, which are then fed into the 3D interpreter.

3.3 Lifting into 3D

The 3D interpreter is used to regress the 3D location of each keypoint using the heatmaps estimated in the previous section. The architecture from Wu et al. [6] was used seeing as it is a fast network and was shown to work on many different sets of 3D data. Though the most popular models being analysed are those of motion captured humans, the use of synthetic models, though slightly dissimilar to reality, has shown to provide similar results [18]. This is done through the process of data augmentation and the inclusion of noise.

Firstly, the system then finds the point with the maximum value from each of the n heatmaps and assumes these are the locations of the keypoints. It then normalizes the u,v keypoint coordinates using equations 4 and 5 to eliminate size differences between each image. In addition this allows to approximate $\widehat{x}_{ij} = \widehat{u}_{ij}$ and $\widehat{y}_{ij} = \widehat{v}_{ij}$.

$$\widehat{u}_{ij} = \frac{u_{ij} - \overline{u_i}}{(\sigma(u_i) + \sigma(v_i))/2} \tag{4}$$

$$\widehat{v}_{ij} = \frac{v_{ij} - \overline{v_i}}{(\sigma(u_i) + \sigma(v_i))/2} \tag{5}$$

Like in the original publication, the system takes every set of $\widehat{u}_{ij}$, $\widehat{v}_{ij}$ values and inputs them in a 6 layer, fully connected network. The network then outputs the depth coordinate for each respective set of 2D coordinates. The values $\widehat{x}_{ij}$, $\widehat{y}_{ij}$ and $\widehat{z}_{ij}$) can then be concatenated to form the full set of 3D coordinates. Finally, the keypoints are appropriately connected to form a 3D skeleton of the structure. This shows both the structure of the object and it's pose in 3D space.

4 Implementation

4.1 Keypoint Localization

This section explains the process of implementing and training the multi stage keypoint localization network.

4.1.1 Parameter Alteration

The keypoint estimation system was subject to many parameters that had to be altered in order to generate appropriate results. Firstly, the first stage of the network was trained using input images and heatmaps with size 368x368 (as per the original network). Therefore, the images had to be resized while keeping their aspect ratio in order to avoid information distortion.

The system was then left to run with a learning rate of $1 * 10^3$ for 50 epochs using a xavier initialization. In addition, the input data used was an image with its values ranging between 0 and 255. Although the results of this trial lead to a decrease of the loss function, the mean difference between the coordinates yielding the highest value in the output and the ground truth remained constant.

Following this, the addition of normalizer layers[19] was used with the aim of a better learning process that is less reliant on weight initialization. In addition, the use of dropout layers[3] with a dropout rate of 0.5 was also experimented. Finally, the input format of the images was converted to a range that spanned -0.5 to 0.5 as the normalizing of input images has shown to improve the effectiveness of training convolutional neural networks[11]. Unfortunately, very similar results to the previous version were achieved.

The next modification to the network was the initialization of the hidden layers using the weights calculated while training human keypoint estimation in the paper by Tome et al. [1]. This was done as some of the features needed to be extracted are shared between human keypoint estimation and object keypoint estimation. The result of this was the clear extraction of keypoints within the image. Though keypoints were being extracted, the results showed unexpected keypoints in each heatmap.

Finally, it was determined that the size of the input image was a fundamental factor in the functionality of the network. The original dataset images were of various sizes, and to fit within a 368x368 array, some had to increase in size and others had to decrease. For those that had to increase in size, the resizing of the image was carried out using interpolation between the original pixels. Though this is not visible to the eye, the change in size created artifacts that affect the network's ability to analyze the image. After making the change to the network, allowing it to receive images of size 200x200, the generated results were as expected.

4.1.2 Data Generation

The data used to train the keypoint localization network[6] compromised of a series of images displaying furniture. Each of the images has an associated set of n 2D coordinates, where n is the amount of keypoints the object has. The data set holds images of chairs, beds, sofas and tables. The designed system was trained using the chair dataset, which holds 1750 image/keypoint combinations for testing and 422 for training. The chairs used were composed of 10 keypoints which included 4 feet, 4 seat extremities and 4 upper back extremities.

To produce data in an appropriate format to train the network, it must be modified. Firstly the images are loaded along with their associated keypoints. The images are reshaped to have their largest size be of 200px, and their smaller side scaled appropriately. The smaller side is then padded to make the final image 200px by 200px, as this is the input size for the network.The keypoints are then scaled to the same size as the image, and ground truth heatmaps are generated for each keypoint. This is done by using an array of zeros and placing a multivariate gaussian distribution with its mean as the coordinates of the respective keypoint and a variance σ. Finally, the n+1th heatmap is generated by using an array of ones and subtracting from it the sum of the rest of the heatmaps.

To then compensate for the small amount of training images, the data is augmented and

more training data is generated. This is done by rotating the image to different angles, as well as vertically mirroring the image. The angles used were -45° , -15°, 0°, 15° and 45°. This process increases the amount of training images by ten times, making the network less likely to overfit and allowing it to output more precise results.

Finally, sets of 100 images and heatmaps are saved in numpy files, which can later be read one at a time by the network. The data is divided this way to remove the possibility of running out of memory while executing the program.

4.1.3 Network Structure

Before being input into the first stage, the values of an RGB image are first divided by 255 and have 0.5 subtracted from them. This normalization is done to bring the images from the range [0 255] to the range [-0.5 0.5]. The resulting image is then placed into the feature extractor section of the network. This consists of 2 3x3 convolutional layers, followed by a 2x2 pooling layer, 2 3x3 convolutional layers, a 2x2 pooling layer, 4 3x3 convolutional layers, a 2x2 pooling layer and finally 2 3x3 convolutional layers. All the convolutional layers in the network use a ReLU activation function.

The output of the feature extractor is then fed into the keypoint estimation network. This consists of 5 3x3 convolutional layers followed by a 1x1 convolutional layer. This outputs n+1 heatmaps which represent the object's keypoints. The output is then scaled back to the original size of the heatmaps and compared to the ground truth through a mean squared error loss function 6.

$$cost = \sum_{i=1}^{n} \frac{(y_i - f(x_i))^2}{n} \tag{6}$$

The input for the second stage is the concatenation of the original image, the output of the fifth convolutional layer of the keypoint estimator and a heatmap showing the position of the object being evaluated. As the objects in the data set are mostly centered in their respective image, the heatmap used is a gaussian distribution. This distribution is given a mean at the center of the image and a variance σ.

The keypoint refinement network consists of 5 7x7 convolutional layers followed by a 1x1 convolutional layer. Like the previous stage, this network outputs n+1 heatmaps corrisponding to the images' keypoints. Finally, another mean squared error loss function 6 is used to train the second stage of the network. Both loss functions are optimized using gradient descent. The entire system is shown in figure 9 and was implemented using the tensorflow library.

4.1.4 Training Methods

The network is initiated by loading the weights and biases from "Lifting from the Deep: Convolutional 3D Pose Estimation from a Single Image"[1] for all the hidden layers. Next, sets of 100 images and heatmaps are loaded at a time and batch sizes of 10 are input into the network. The sets of images are randomly chosen to make overfitting less likely. The first stage is then trained with a learning rate of $1 * e^{-3}$ while ignoring the second stage. This is done to reduce the complexity of the model being trained, leading to faster training and more precise results. Once the first stage is trained, the entire system undergoes training with a learning rate of $1 * e^{-2}$. The training occurs for a number of epochs and testing data is evaluated after every set of images is used for training. The system used for training contained an NVIDIA TITAN X and 64GB of RAM.

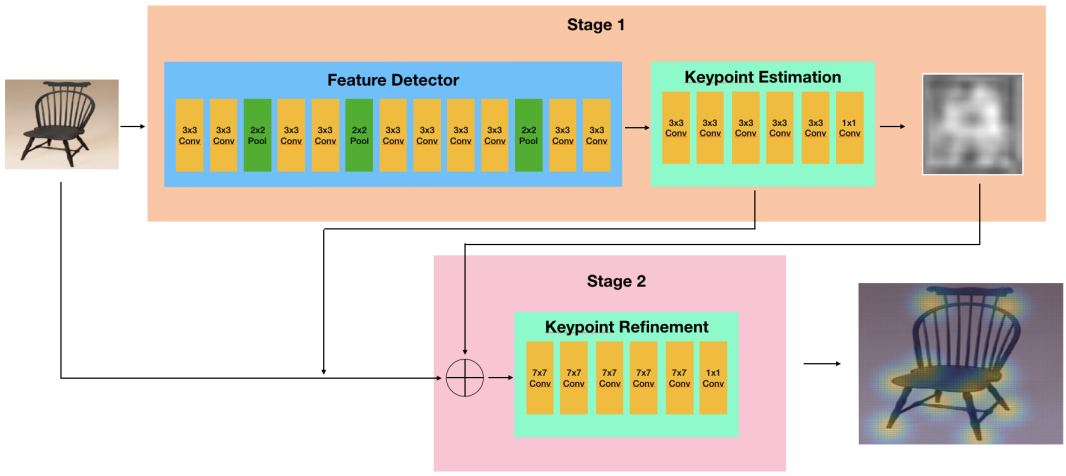

Figure 9: Diagram of keypoint localization network used in the system

4.2 Lifting into 3D

This section explains the process of implementing and training the fully connected 3D interpreter network.

4.2.1 Data Generation

As previously stated, the data used to train the 3D interpreter system consists of 3D coordinates corresponding to furniture key points. These have been extracted from synthetic models in the IKEA dataset [10] for use in the paper "Single Image 3D Interpreter Network"[6]. Firstly, the coordinates are normalized using equations 7,8 and 9, where $\overline{x_i}$, $\overline{y_i}$ and $\overline{z_i}$ are the mean values of each dimension and $\sigma(x_i)$ and $\sigma(y_i)$ are the standard deviation of each dimension. This results in the coordinates all being of the same scale, eliminating differences between different model sizes.

$$\widehat{x}_{ij} = \frac{x_{ij} - \overline{x_i}}{(\sigma(x_i) + \sigma(y_i))/2} \tag{7}$$

$$\widehat{y}_{ij} = \frac{y_{ij} - \overline{y_i}}{(\sigma(x_i) + \sigma(y_i))/2} \tag{8}$$

$$\widehat{z}_{ij} = \frac{z_{ij} - \overline{z_i}}{(\sigma(x_i) + \sigma(y_i))/2} \tag{9}$$

Following this, the models are projected onto a 2D surface using a weak perspective projection 10. This produces a set of 2D coordinates from the 3D model. Before projecting the model, data augmentation is applied onto it. By modifying the data, a large amount of training samples can be derived by a small amount of original samples. This is due to the fact that the same 3D model can result in many variations of 2D projections. The models are therefore rotated and scaled to different sizes. In addition, gaussian noise is added to the 2D projections to account for imprecisely localized keypoints when testing the model. The 2D keypoint coordinates are then used as the input values while the normalized z coordinate for each keypoint is used as the output value. Overall, the amount of data used is 178695 samples, 500 of which are reserved for testing purposes.

$$\begin{bmatrix} u_{i1} & u_{i2} & ... & u_{in} \\ v_{i1} & v_{i2} & ... & v_{in} \end{bmatrix} = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \end{bmatrix} \begin{bmatrix} x_{i1} & x_{i2} & ... & x_{in} \\ y_{i1} & y_{i2} & ... & y_{in} \\ z_{i1} & z_{i2} & ... & z_{in} \end{bmatrix} \tag{10}$$

4.2.2 Network Structure

The network architecture consists of 6 fully connected layers (Figure 10). The input layer of the network contains 2n neurons (for the x and y inputs), where n is the amount of keypoints. The 4 hidden layers also contain 2n neurons, and the output layer contains n neurons (for the z output). The activation function used for each of the layers is a hyperbolic tangent as seen in equation 11 where a is the input vector, Ω is the neuron's weight and b is the neuron's basis vector. The cost function used is an L2 mean squared error (equation 6) and the optimizer used is gradient descent.

$$\widehat{a_l} = tanh(\Omega_l a_l + b_l) \tag{11}$$

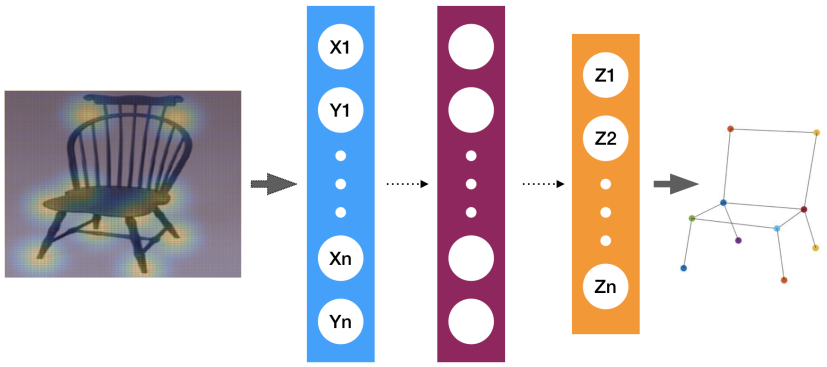

Figure 10: Diagram of 3D interpreter neural network

4.2.3 Training Methods

The learning rate was set as 0.1 and the initiation of the layers was carried out as a Xavier initiation [20]. The model was set to run for a number of epochs until it reached convergence with a batch size of 50. In addition, once every epoch the model was evaluated using testing data to make sure it was not overfitting.

5 Evaluation

5.1 Keypoint Estimation Evaluation

This section evaluates the first and second stage of the keypoint estimation system.

5.1.1 First Stage

There were 2 values being monitored during the training of the first stage. The first of these is the distance between the coordinates at the maximum value of the output and the coordinates at the maximum value of the ground truth for each heatmap. The second is the loss function output (Figure 11). For pixel distance measuring purposes, the n+1th heatmap was ignored, as its only use is for multi stage training. The network was ran for 10 epochs, which is when it was assumed that the lowest pixel distance for this system had been achieved.

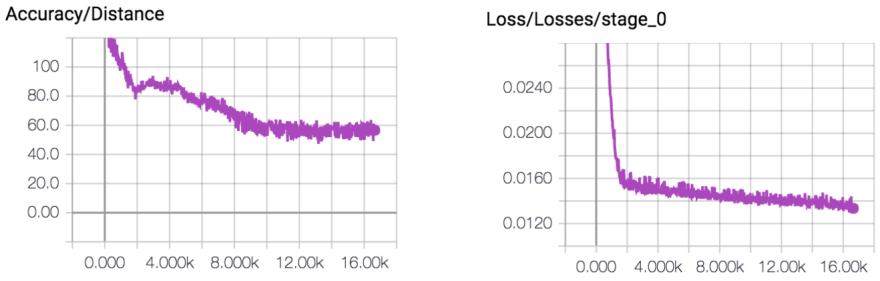

Figure 11: Loss value and maximum pixel distance for stage 1 using test data

The mean distance between maximum pixels was found to be 57.34px. The difference corresponds to 20.27% of the diagonal length of the image: 282.84px. This is a poor result, which can also be seen by examining the heatmap outputs found in figure 12. The displayed figures are that of the original image input, taken from the test data set, and all the output heatmaps. It can be seen that while some areas are being highlighted, no single zone is precisely identified as a keypoint. In addition, borders around the heatmaps can be noticed, which don't convey any relevant information. This output is therefore inadequate for any use, which is why the second stage of the network is needed.

5.1.2 Second Stage

As per the previous network stage, the second stage was also evaluated by analyzing the loss of the system and the difference between maximum values between heatmaps (Figure 13). The output of the first stage was fed into the second stage's input, and the entire system was left to run for 60 epochs. This is when the pixel distance had stopped changing and the loss had stopped decreasing. The system including the second stage outputs a loss 54.17% lower than the isolated first stage. This is due to the heatmap refinement having access to both the original image and output from the first stage, allowing it to be corrected.

In addition to a lower loss, the full system containing both stages outputs a mean distance between maximum pixels of 41.44px. This value corresponds to 14.65% of the image's diagonal size, which is drastically lower than that found in the first stage. The outputs of the system can be seen in figures 14 and 15. It can be noticed that the keypoints of the chair are being identified with precision, and separated from the background of the image. This can be seen in both the n heatmaps, and in the n+1th heatmap, which clearly shows every keypoint. In addition, occluded keypoints are also extracted, such as the back left seat in figure 15.

The main error that arises from the outputs is the identification of multiple keypoints in the same heatmap. As shown in the results, for each height level of the chair, all the keypoints are identified. In both examples, all four feet keypoints are identified in all four feet heatmaps, all four seat keypoints are identified in all four seat heatmaps and both top keypoints are identified in both top heatmaps. This suggests that the network is able to differentiate between the different categories of keypoint, but is not able to estimate the exact member of each group.

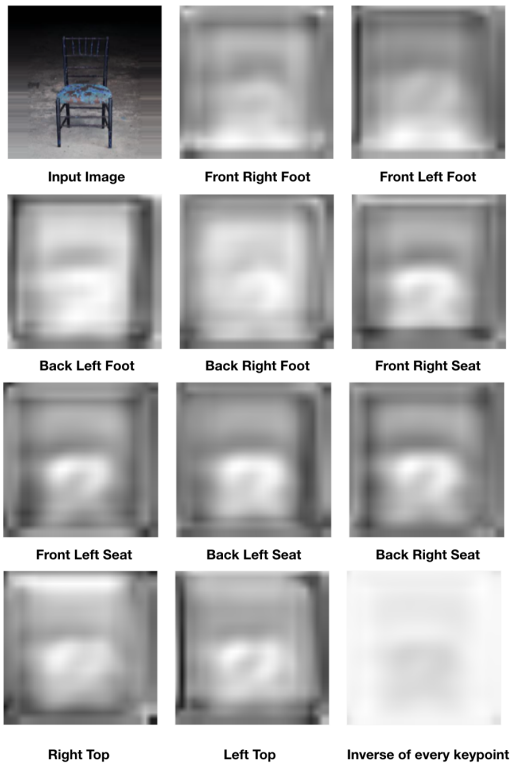

Figure 12: Example of stage 1 output heatmaps using test data

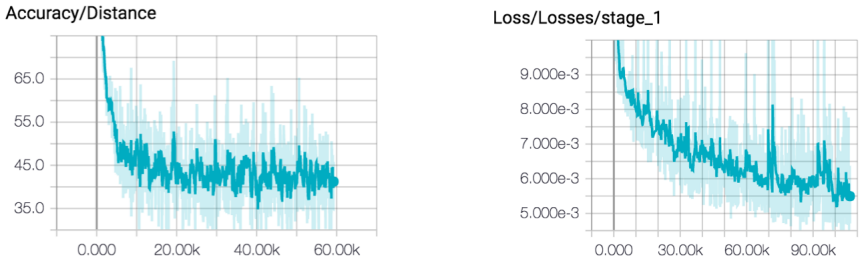

Figure 13: Loss value and maximum pixel distance for stage 2 using test data

Given this error, the data produced by the network is not sufficient to be used in conjunction with the 3D pose interpreter. There are various reasons for which the system may not able to differentiate between keypoints of the same type.

Firstly, the network on which the keypoint identification system was based estimates human keypoints. While it was hypothesized that this architecture would translate from human to object keypoint recognition, there are some factors that differentiate the two physical structures. The most prominent of these is the fact that human bodies contain features that facilitate the estimation of their orientation. An example of this is human feet compared to chair feet. In a chair, all four feet look the same and do not have any protruding feature revealing their orientation. On the other hand, human feet will always face away from the human, which makes it clear to know what direction the body is facing. Their individual orientation also helps convey which foot is the right one and which is the left.

Another feature that hinders object recognition is that, while chairs are always symmetric,

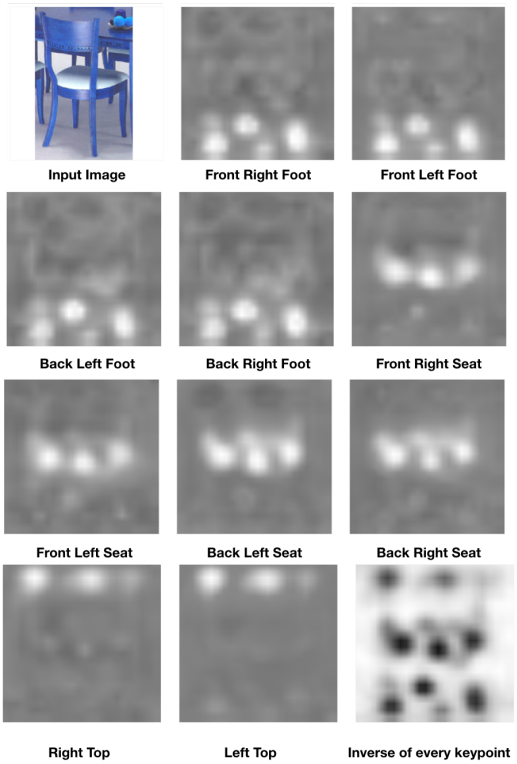

Figure 14: First example of stage 2 output heatmaps using test data

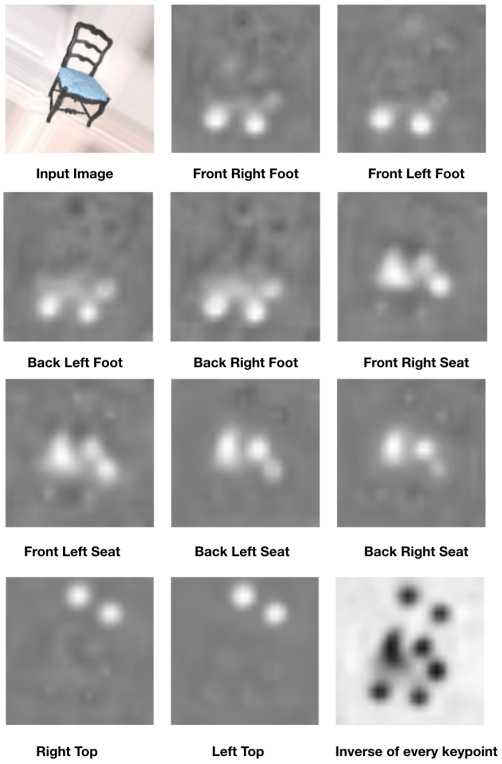

Figure 15: Second example of stage 2 output heatmaps using test data

humans are not. Body poses can exist with arms, legs and joints being positioned in many different arrangements. By then having a data set that introduces these distinct poses, the network can be trained to recognize them and differentiate between sides of the body, which is not possible with furniture. In addition, relationships between body extremities and joints can be made that further ease the estimation process.

Finally, it is possible that the training data set did not have enough training images and coordinate combinations. The initial number of these was 1750, but after data augmentation the amount of training images resulted in 17500. The number of original images is therefore small when compared to image datasets used for similar purposes[2].

5.2 3D Interpreter evaluation

The 3D interpreter was left to run for 200 epochs, which is when it reached convergence. The final error results were calculated by finding the mean absolute difference between the ground truth value and the output value using the testing data. Using this method, the mean error for a set of 500 training models was 0.0786 with a standard deviation of 0.101. This is measured in normalized units seeing as all the input and output data is normalized. The results of the system are comparable to state of the art systems which lift 2D values into 3D (Table 1).

Publication	Normalized mean error
Ruiqi Zhao et al. [16]	0.0299
Zhou et al. [21]	0.0653
Ramakrishna et al. [22]	0.0983
This paper	0.0786

Table 1: Table comparing the mean error of models used to lift 2D data into 3D

The testing data being subjected to the interpreter network can be seen in the examples found in figure 16. These show the original 3D model after being put through data augmentation, its 2D projection onto a plane, and the recomputed model after feeding the 2D coordinates into the network. The visual results shown, along with the numerical data, suggest that the system can successfully lift a set of 2D points into 3D.

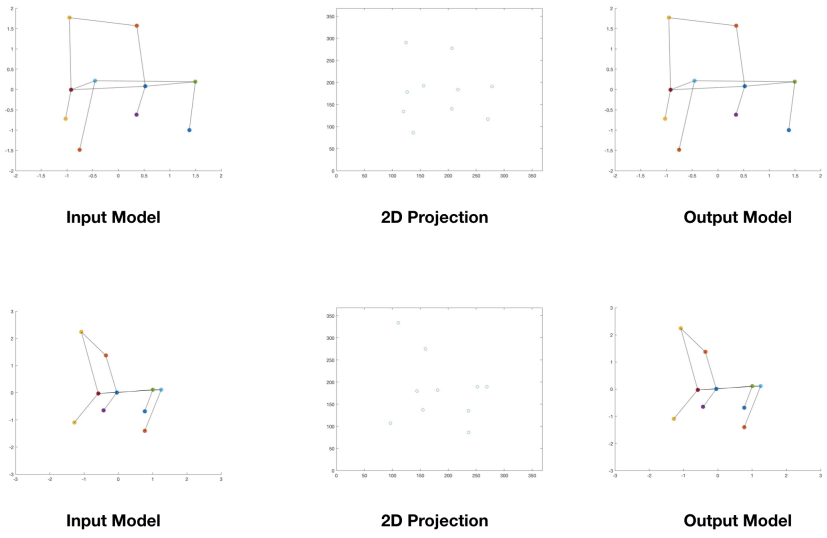

Figure 16: Diagram of 3D synthetic models, their 2D projection, and the the regressed 3D structure from the 3D interpreter network

To further test the 3D interpreter, the ground truth heatmaps from the keypoint estimation data set were input into the system. The outputs can be seen in figures 17, 18 and 19. These span from -2 to 2 units in all 3 dimensions. It can be seen that the 3D skeletons display the correct orientation and depth for figures 17 and 18. Although figure 19 also estimates the correct dept, it also shows slight irregularities in its proportions, such as the two front keypoints being further apart than the four posterior keypoints. This is due to the camera being placed above the chair, leading to an unexpected projection of the object. In addition, further irregularities may be due to the fact that a weak perspective projection is used when training the model. This limitation can lead to unexpected output models which would require the regression of a fully perspective projection.

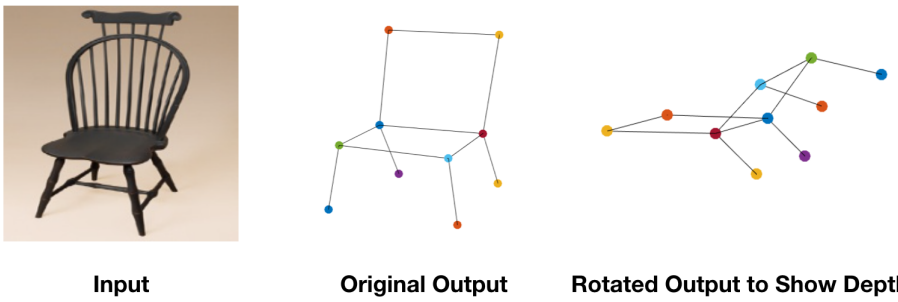

Figure 17: Diagram of 2D ground truth heatmaps lifted into 3D using the trained model

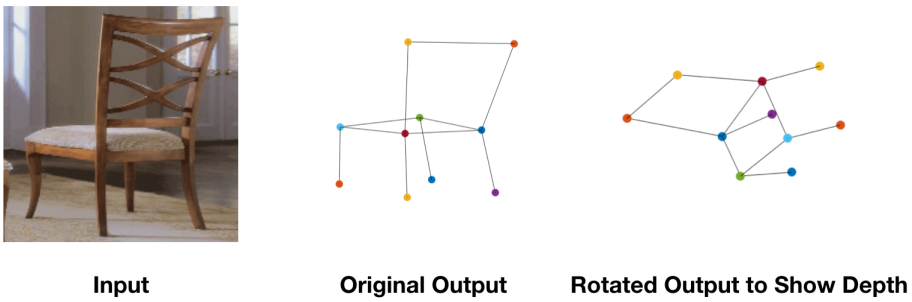

Figure 18: Diagram of 2D ground truth heatmaps lifted into 3D using the trained model

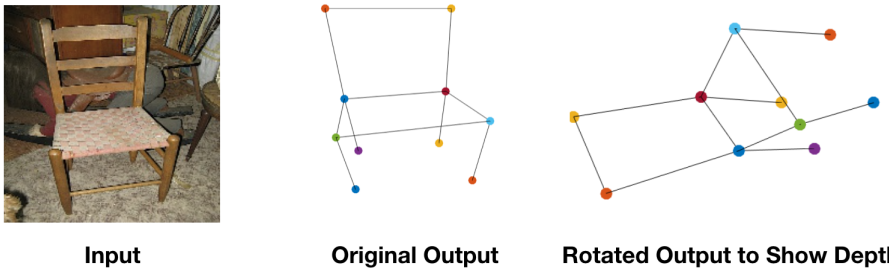

Figure 19: Diagram of 2D ground truth heatmaps lifted into 3D using the trained model

5.3 Full System Evaluation

Although the keypoint extractor system is not able to estimate exact keypoints, it still outputs useful data. This data, in certain circumstances, is enough to lift the 2D image into 3D.

The case in which this is true is when the chair being examined is directly facing the camera. By knowing the rotation of the object, the multiple keypoints being identified can be isolated with confidence. This is true because depending on the orientation of the chair, each keypoint's relationship to the frame of the image is known. For example, the right top keypoint will always be the upper left most keypoint, and the left back leg will almost always be the second keypoint from the bottom right. By exploiting this information, specific images can be evaluated using the whole system.

To translate the extracted keypoints into applicable heatmaps, the n+1th output heatmap was used. This contains every keypoint estimation on it. Using simple segmentation, the individual keypoints were extracted from the heatmap. By then examining their position relative to the image frame, new heatmaps were formed. The mean pixel distance between the keypoints found using the newly generated heatmaps and the ground truth was calculated to be 7.38px. This was found by evaluating multiple images with a chair facing the camera. This result is 561.51% more accurate than the output of the second stage and 734.01% more accurate of the output of the first stage. In addition, the mean distance is 2.61% of the diagonal of the image size, which is exponentially better than the previous results.

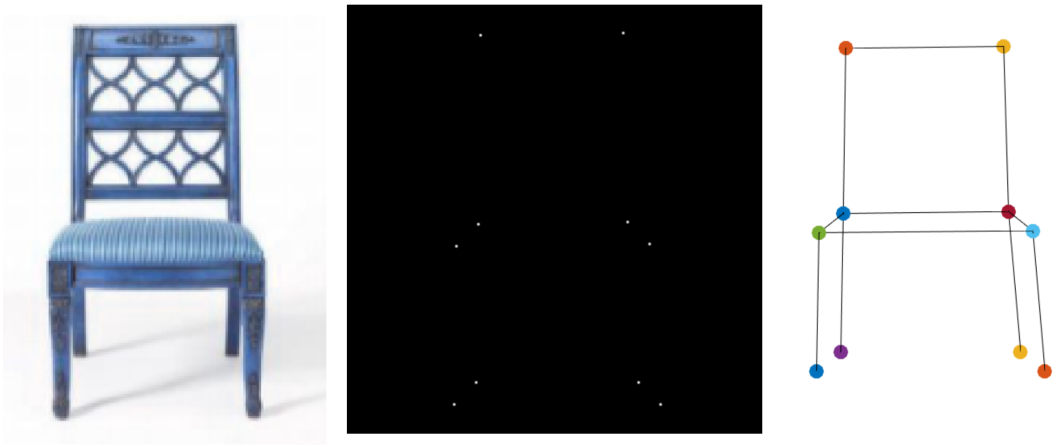

Figure 20: Diagram of original image, keypoint extraction after examining positional relationship with the image frame and 3D reconstruction

Though this is a very specific scenario, these results are accurate enough to produce a 3D representation of the skeleton of the object. These can be seen in figures 20, 21 and 22. These images show that the proportions and depth of the 3D object were successfully retrieved. A clear difference between all 3 3D skeletons reflects the different proportions of each chair in their respective original image, such as the width, height and depth of each of their structures.

Figure 21: Diagram of original image, keypoint extraction after examining positional relationship with the image frame and 3D reconstruction

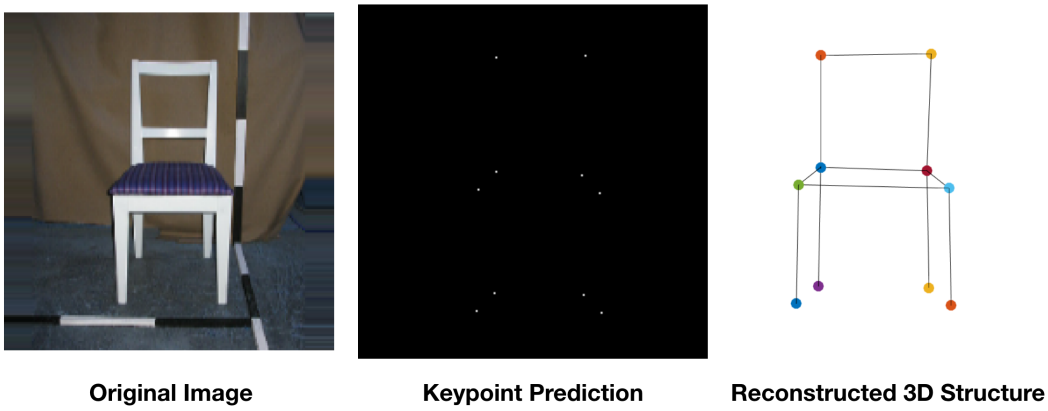

Figure 22: Diagram of original image, keypoint extraction after examining positional relationship with the image frame and 3D reconstruction

6 Conclusion and Further Work

6.1 Further Work

This piece of research has the ability to be extended in many ways. Firstly, the keypoint estimation neural network should be modified to add the ability of individual keypoint recognition. One of the ways of possibly doing this is by combining both the keypoint estimator network and the 3D projection network within the first stage of the system, as is done in Tome et al.[1]. This architecture would consist of the first stage of the keypoint estimation network returning a heatmap output. The output could then be projected into 3D using the object interpreter network. The 3D model would then be reprojected as a 2D set of keypoints, fused with the heatmap output, and fed into the second stage.

This system could provide the added ability to restrain keypoints from a wrong location by using the 3D interpreter. The reasoning behind this is that if the 3D interpreter network sees

that the relative 3D position of two keypoints is impossible, it will likely learn to detect this and understand how to solve it. This could involve moving individual keypoint locations or swapping two sets of coordinates. The suggested architecture would give the system a further global view of the data and it would have the ability to examine the final 3D output when computing loss.

Another way to make a system that could possibly output the correct 3D keypoints at any angle would be through the use of a preconstructed 3D model, such as in Wu et al.[6]. In this case, the model would correspond to an average of many 3D structures of a given category (eg. chairs). This model could then be altered using given parameters, such as the width of a chair, the length of its legs, the chair's rotation, etc. This would eliminate the need of the current 3D interpreter network as it would not need exact keypoints to construct a 3D skeleton. In addition, this would make it impossible to output a model that has incorrect main features (the existence of legs), as the output would be based on a pre existing model. This interpreter could also be used in conjunction with the previously described multi stage reprojection network to garner both systems' benefits. The downside of this system would be the constraint of having to adhere to a given general shape. This would make it impossible for the network to interpret unusually shaped objects of the same category.

In addition to improving the keypoint estimation network, further work could also be applied onto the current 3D interpreter network. Although the system yields sufficiently good results, it is constrained by its assumption of a weak perspective projection of the images. This, as shown in the evaluation, leads to unexpected results within the x and y axes. A way to fix this would be to use an altered network architecture which allows the output of the height and width coordinates as well as the depth coordinate for each keypoint. This would result in not being constrained by the input values being normalized to adhere to the assumption of a weak perspective projection. Instead, the full 3D coordinates would be output, leading to more precise results.

Once a higher result accuracy is achieved there is many other improvements it could go through. The first is implementing the ability to simultaneously identify the structure of a human and a piece of furniture in the same frame. This could be done by using both the designed network and the original[1]. In addition, the system can be expanded by combining object recognition with structure identification. This would make it so that the structure being identified does not have to be specified before applying the system to it. This would be possible by training different networks using different objects (chairs, beds etc.). A system could then classify the object in a given image and use the appropriate network to output the correct 3D representation

A series of applications for real world use could also be built on top of a working system. Firstly, knowing the 3D structure of an object could be used for augmented reality purposes. An example of this is placing virtual objects on a real table by knowing the table's structure. Another use for this technology is robot vision. Knowing the skeleton of an object can allow a machine to know how to circumvent it or manipulate it. In addition, this system can be used for image retrieval. By knowing the 3D structure of an object, a system can be designed witch returns images with identical or similar structures. Finally, this technology can be used for the construction of object graphs. These give the ability to smoothly transition between objects by inputting slight variations in the structure. An example of this is ranging from images of chairs with short legs to chairs with long legs by knowing their 3D structure.

6.2 Conclusion

This project was dedicated to designing a system that, using a single rgb image of an object as an input, could output the object's 3D structure. This was done through the use of a keypoint estimator and a 3D interpreter. It was found that the trained keypoint estimator was able to, with relative accuracy, estimate the coordinates of the keypoints on the object. Although keypoints were identified, it was found that the system was not able to differentiate between

different keypoints on the same level (upper back, seat and legs).

Another finding from the project is how much the accuracy of an output can improve through the use of a multi stage network. By taking the original output and refining it, the system is able to extract useful data from an unusable output. In addition to the keypoint estimation, a working depth estimator using a 2D keypoint input was designed. The results of this were successful, as shown by its output models. Finally, the possibility of using the combined system for outputting a chair's 3D structure directly from an image was found. The limitation of this was that the image must portray the chair facing the camera in order to exploit the known relationships between the object and its frame. The results found from using this method showed an accurate representation of the output 3D model. In future iterations of this system, it the ability to calculate the 3D structure of images with objects in different orientation is expected, along with more precise overall results.

References

[1] D. Tomè, C. Russell, and L. Agapito, "Lifting from the deep: Convolutional 3d pose estimation from a single image," *CoRR*, vol. abs/1701.00295, 2017.

[2] S. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," *CoRR*, vol. abs/1602.00134, 2016.

[3] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[5] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," *CoRR*, vol. abs/1411.4280, 2014.

[6] J. Wu, T. Xue, J. J. Lim, Y. Tian, J. B. Tenenbaum, A. Torralba, and W. T. Freeman, *Single Image 3D Interpreter Network*, pp. 365–382. Cham: Springer International Publishing, 2016.

[7] M. Aubry, D. Maturana, A. Efros, B. Russell, and J. Sivic, "Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models," in *CVPR*, 2014.

[8] A. Kar, S. Tulsiani, J. Carreira, and J. Malik, "Category-specific object reconstruction from a single image," *CoRR*, vol. abs/1411.6069, 2014.

[9] M. F. Hansen, G. A. Atkinson, L. N. Smith, and M. L. Smith, "3d face reconstructions from photometric stereo using near infrared and visible light," *Computer Vision and Image Understanding*, vol. 114, no. 8, p. 942–951, 2010.

[10] J. J. Lim, H. Pirsiavash, and A. Torralba, "Parsing ikea objects: Fine pose estimation," in *2013 IEEE International Conference on Computer Vision*, pp. 2992–2999, Dec 2013.

[11] A. Karpathy, "Convolutional neural networks for visual recognition," 2017.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, p. 2012.

[14] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014.

[15] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele, "Deepcut: Joint subset partition and labeling for multi person pose estimation," *CoRR*, vol. abs/1511.06645, 2015.

[16] R. Zhao, Y. Wang, and A. M. Martínez, "A simple, fast and highly-accurate algorithm to recover 3d shape from 2d landmarks on a single image," *CoRR*, vol. abs/1609.09058, 2016.

[17] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[18] K. Krissian, G. Malandain, N. Ayache, R. Vaillant, and Y. Trousset, "Model-based detection of tubular structures in 3d images," *Computer Vision and Image Understanding*, vol. 80, no. 2, pp. 130 – 171, 2000.

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[20] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Y. W. Teh and M. Titterington, eds.), vol. 9 of *Proceedings of Machine Learning Research*, (Chia Laguna Resort, Sardinia, Italy), pp. 249–256, PMLR, 13–15 May 2010.

[21] X. Zhou, S. Leonardos, X. Hu, and K. Daniilidis, "3d shape estimation from 2d landmarks: A convex relaxation approach," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4447–4455, June 2015.

[22] V. Ramakrishna, T. Kanade, and Y. Sheikh, *Reconstructing 3D Human Pose from 2D Image Landmarks*, pp. 573–586. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

Appendix

Code repository

https://github.com/Andrydood/Msc-Final-Project

Past Work

Oral presentation

https://docs.google.com/presentation/d/1odaY3UWuR7vA0ZT9UOPWOP_T7FPuJbKg8VzRJfHHqT8/edit?usp=sharing

Literature review

https://drive.google.com/file/d/0B-x4QKaAc4M6cEhFZENfMFpMeXM/view?usp=sharing