UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

GUILHERME ANTONIO CAMELO

# Human path prediction through machine learning trained with 2D video data and estimated 3D pose data

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. Rosa Maria Vicari

Porto Alegre
July 2019

**CIP — CATALOGING-IN-PUBLICATION**

Camelo, Guilherme Antonio

Human path prediction through machine learning trained with 2D video data and estimated 3D pose data / Guilherme Antonio Camelo. – Porto Alegre: PPGC da UFRGS, 2019.

84 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–Brasil, 2019. Advisor: Rosa Maria Vicari.

1. RNN, LSTM, Deep Learning, Machine Learning, Path Prediction, Kinect, Robotics, Pose, 3D Pose, 3D Pose Estimation, Openpose, Human3.6m. I. Vicari, Rosa Maria. II. Título.

## ABSTRACT

The interaction between robots and humans has been advancing at an ever-rising pace. Being aware of not only where humans are but also predict where humans will go and do next is an important feature to have in an assistant robot. The main goal of this project is to explore ways of improving this feature. Future trajectories of humans walking were predicted using deep learning fed with RGB data in a controlled environment in order to better assist humans when needed. Kinect was used to achieve that goal along with its RGB camera, infrared laser projector and infrared sensor. Data of people walking in a controlled environment was collected and a dataset was created. Data from *Human3.6m* dataset was used as well. The data was used to train RNN-LSTM models created to predict future paths. Openpose was used to identify humans and their body joints and create poses. 3D pose data was estimated from 2D pose data using a 3D Pose Estimator in order to recreate the path in the 3D space. An LSTM model with a 3D feature was created and trained with 3D estimated data. A path prediction model with a 3D element was assessed in comparison with a 2D path prediction model. While models were able to learn from the data and present good predictions in some cases, they were not able to learn in other cases outputting bad predictions. The metrics used to get quantitative results presented limitations to measure the predictions. Limitations of using a 3D Pose estimator for 3D path reconstruction were described. As a result of our project, models that predict future path of people with different designs and performances were developed. As contribution, a dataset of 113 GB containing people walking in a controlled environment was created and a methodology to estimate 3D path information from 2D Pose was proposed.



**Keywords:** RNN, LSTM, Deep Learning, Machine Learning, Path Prediction, Kinect, Robotics, Pose, 3D Pose, 3D Pose Estimation, Openpose, Human3.6m.

**Predição de caminho de humanos usando machine learning com dados de video 2D e dados estimados de pose 3D**

## RESUMO

A interação entre robôs e humanos vem avançando a um ritmo cada vez maior. Estar ciente não só de onde os humanos estão, mas também prever aonde os humanos irão e o que farão é um recurso valioso de se ter em um robô assistente. O principal objetivo deste projeto é explorar formas de melhorar essa habilidade. Trajetórias futuras de humanos caminhando são previstas usando deep learning alimentado com dados RGB em um ambiente controlado com a intenção de melhor assistir humanos quando necessário. Kinect foi usado para atingir esse objetivo junto com suas câmeras RGB e infravermelha. Dados de pessoas caminhando em um ambiente controlado foram coletados e um dataset foi criado, bem como foram utilizados dados do dataset Human3.6m. Os dados foram usados para treinar modelos RNN-LSTM criados para prever caminhos futuros. Openpose foi empregado para identificar humanos e suas articulações do corpo, além de criar poses. Os dados de pose 3D foram estimados a partir de dados de pose 2D usando um estimador de pose 3D, reconstruindo um caminho no espaço 3D. Um modelo LSTM foi desenvolvido e treinado com dados adquiridos dos poses 3D estimados. O modelo de predição de trajetória 3D foi avaliado em comparação com um modelo de predição de caminho 2D. Embora os modelos pudessem aprender com os dados e apresentar boas previsões em alguns casos, eles não foram capazes de aprender em outras situações. As métricas utilizadas para obter resultados quantitativos apresentaram limitações para medir as previsões. As limitações do uso do estimador de Pose 3D para reconstrução de caminho em 3D foram descritas. Como resultado do nosso trabalho, modelos que prevêem o caminho futuro de pessoas com diferentes designs e performances foram desenvolvidos. Como contribuição, um dataset de 113 GB contendo pessoas andando em um ambiente controlado foi coletado e uma metodologia para estimativa do caminho 3D baseado no pose 2D foi proposta.

**Palavras-chave:** RNN, LSTM, Deep Learning, Machine Learning, Predição de caminho,Kinect, Robotica, Pose, 3D Pose, Estimador de Pose 3D ,Openpose, Human3.6m.

## LIST OF FIGURES

## LIST OF TABLES

## LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| CNN | Convolutional Neural Network |
| ML | Machine Learning |
| DL | Deep Learning |
| PCD | Point Cloud Data |
| ROS | Robot Operating System |
| ANN | Artificial Neural Network |
| NN | Neural Network |
| RNN | Recurrent Neural Network |
| LSTM | Long-Short Term Memory |
| JSON | JavaScript Object Notation |
| XML | eXtensible Markup Language |
| NSD | Norsk Senter for Forskningsdata |
| RGB | Red-Green-Blue |
| RGB-D | Red-Green-Blue-Depth |
| RAM | Random Access Memory |
| ADE | Average Displacement Error |
| FDE | Final Displacement Error |
| GPU | Graphics Processing Unit |
| NLP | Natural Language Processing |
| NLP | Frames per Second |
| GRU | Gated Recurrent Unit |
| IR | Infrared Radiation |
| POI | Point of Interest |
| AP | Activity Path |

MSE Mean Square Error

HRI Human Robot Interaction

**CONTENTS**

**Acknowledgments**

This project was created during a collaboration between the University of Oslo (UiO) and Federal University of Rio Grande do Sul (UFRGS). A scholarship was used for the exchange program in the University of Oslo funded through/supported by Norwegian Centre for International Cooperation in Education (SIU) as a part of the project Rio Grande do Sul and Oslo collaboration on AI and Robotics (ROCAIR) project, under grant agreement 10128 and by Research Council of Norway (RCN) and SIU as a part of the project Collaboration on Intelligent Machines (COINMAC) project, under grant agreement 261645.

During the exchange program at UiO, Professor Dr. Jim Tørresen guided the work performed. Significant collaboration was also provided by UiO Ph.D. Candidate Justinas Miseikis.

The scholarship used in Brazil was financed by the Brazilian Institution CNPq (Conselho Nacional de Desenvolvimento Cientifico e Tecnologico).

# 1 INTRODUCTION

Robotics technology has been growing at astonishing rates. The use of deep learning made it possible to accomplish tasks that would be considered impossible before. The combination of robotics and machine learning has been creating solutions that never cease to surprise what would be expected from computers.

The interaction between computer and humans is becoming more common by the day, and several technologies have been presented where robot assistants help humans in many different ways. One of which is robots being used as assistants to elderly people, helping in several daily tasks such as in (POLLACK et al., 2002). In this context, the robot has to move accordingly to the position of the human depending on a given task. This can be quite sensitive since depending on the severity of the situation the robot must be able to evaluate with precision what it should do. It should not take too long to arrive at its goal, but must also avoid going too fast in a way that would create a risk to hit the person being assisted. That is one of the reasons why being able to predict the path of humans walking is a good feature to have on a robot.

In this context, human detection is an important feature needed. There are several works that present video-based techniques to detect pedestrians walking on the streets, such as (DOLLAR et al., 2012), and (MIŠEIKIS, 2012) that uses multiple cameras to improve video surveillance. In this dissertation, we perform skeletal detection and afterward create an artificial neural network models capable of predicting future paths of a person walking. The main goal is to explore ways to create good predictions of the trajectory of humans by creating a dataset and designing deep learning models that could contribute to current mobile robotics. Improving this feature would make robots able to be preemptive, moving out of the way when not needed, and approach when its service is required. Thus, always being in the most appropriate place at the right time.

In order to reach the goal of predicting what will be the next path taken by a human in an environment where a robot acts as an assistant, it is first necessary to detect humans. Several works such as Openpose (WEI et al., 2016) (CAO et al., 2017) (SIMON et al., 2017) and Alpha Pose (XIU et al., 2018) (FANG et al., 2017) are already able to identify the whole human pose as well as several body parts successfully. In this research, Openpose is used to detect the human body since its considered the state of the art in this context. Sequential images of people walking are used as an input to Openpose, and the result is assembled and inputted into different neural networks (NN).

It is expected from this project to have a neural network able to predict correctly the future paths that a human will take. To collect data a Kinect was used to record people performing tasks in a kitchen environment and RGB and RGB-D were recorded to create a dataset. Apart from that, a known dataset, *Human3.6m*, was used for experiments with 3D estimated data.

Extracting the path of humans walking in an environment from video is a challenging task. By using a skeletal detection 2D pose information can be acquired in each frame. Therefore, by assembling this information throughout the frames it is possible to recreate a path. It is important to note that this path is a 2D projection of the 3D space and is prawn to errors and lack of accuracy. Instead of using only 2D path reconstruction, a 3D pose estimator is used to create a 3D path in the 3D space. Two NN are designed to deal with each data structure. The goal of this approach is to discover if an estimated 3D information can improve the predictions for the case studied.

Many elements can influence the final performance such as model architecture, data quality, and experimental obstacles like the angle of recording and variation of illumination. During this project, those factors are assessed and ways to overcome the issues encountered are explored.

The contributions of this work are the following: a dataset of people walking in a controlled environment performing different tasks was presented, a methodology for predicting 2D and 3D path based on video using deep learning models was created, a framework for human walking analysis was developed, and the limitations of using 3D pose estimator were outlined.

The rest of this document will present the background needed and the related work in chapter 2. The methodology used in the experiments can be seen in chapter 3. A proof of concept experiment with mouse movement prediction employing one of the neural network designs is present in Chapter 4. The initial approach taken as well as the following points of interest approach is shown in chapter 5 followed by the results of the approach. Chapter 6 exposes the 3D pose estimator as well as the comparative approach used to measure the 2D RNN (LSTM) and 3D RNN (LSTM) models used, comparative results of both models are evaluated. The issues encountered along the way are in Chapter 7. Finally, the conclusions and future work can be verified in Chapter 8. The acknowledgment is in Chapter 6. Additional work relevant for decisions made during the master that does not influence directly the final work is presented in the appendices.

# 2 BACKGROUND AND RELATED WORK

This chapter presents the basic concepts related to this work in Section 2.1 and the related work in Section 2.2. The concepts presented here are related to fundamentals of neural networks, focusing on Recurrent Neural Networks (RNN) with Long-Short Term Memory (LSTM) layers, the learning models trained, the techniques and programs used for pose recognition as well as the frameworks used to perform the data collection.

## 2.1 Background

During this project, many technologies and approaches were used and experimented with. In this section the basic background needed for the technologies used is explained.

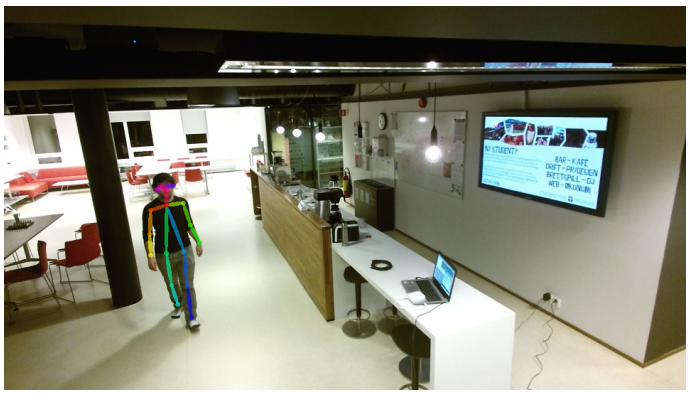
An important part of our project is to identify the human skeleton and its body joints. That can be done with Openpose. Openpose is a real-time multi-person system that is able to identify the position of several body joints when an image with a person is provided, having as result the detection of the human body, hands, and facial keypoints. It is possible to see the outcome of Openpose on a single frame in Figure 2.1.

A RGB-D image is just like a RGB image with an extra channel that corresponds to the image depth. For each pixel, the depth channel of the image keeps the information of the distance between the image plane and the corresponding object in the RGB image. We are able to create an RGB-D image once we have the RGB and depth data of the same scene.

Openpose can use many sources like image, video, webcam and IP camera as input. The output can be basic RGB image, image or video of keypoints, and keypoint saved in several data formats (such as JSON, XML, etc). In this project, we used several frames as input after extracting it from RGB and IR videos recorded with Kinect. The output recorded was the original image with the human body detection plotted on top, as well as the body keypoints.

The data collected using the Kinect was recorded with the Robot Operating System (ROS) from (QUIGLEY et al., 2009). ROS is a collection of frameworks focused on development for robots. It has many tools and libraries that assist with developing complex robotic systems. ROS has the concept of topics. A topic identifies the content of a message. Kinect outputs a few different topics and the ones of interest to us that were

Figure 2.1: Example of Openpose being used in one of the experiments. It is possible to see that openpose can successfully recognize the human skeleton. The image resolution is 960x540.



recorded were \kinect2\sd\image_color_rect , \kinect2\sd\points, and \tf. Those topics are responsible for streaming the RGB and depth data captured. The advantage of using and recording ROS topics is that we can save them in `.rosbag` files and playback when needed. Using this system allows us to have high performance and avoids de-serialization and re-serialization of the messages. This means that when the playback is executed, the data will preserve the original order. For this reason, ROS was chosen.

In this project, we use machine learning (ML) to try to predict future information based on past data. (ZHAO, 2018) describes it as follows, “Algorithms that parse data, learn from that data, and then apply what they’ve learned to make informed decisions”. We work with a specific branch of ML, the deep learning (DL). DL creates neural networks (NN) with many layers between the input and output. This structure with many layers is whats characterizes the *deep* in the name and allows for the model to learn and make intelligent human-like decisions.

The use of ML and DL is highly dependable on the context. DL is more appropriate for complex problems and it can provide very precise prediction and human-like decision. In another hand, creates a big overhead when it comes to execution time and hardware needed. Usually, running DL models demand powerful GPUs and can run for large amounts of time. In order for the DL models to learn a big datasets are often required. When dealing with simpler problems ML is more appropriate. It can run on low-end machines, have excellent performance with small and medium datasets and usually be trained in a smaller amount of time.

According to (BROWNLEE, 2019), we can say that deep learning neural network basically tries to learn a mapping function from inputs to outputs. Many things are in-

volved in this process. Defining the hyperparameters of the model as well as the weights are important and sometimes extensive parts of the process of creating and training a neural network. It is such an integral part of the ML development that (GOODFELLOW; BENGIO; COURVILLE, 2016) states: "It is quite common to invest days to months of time on hundreds of machines in order to solve even a single instance of the neural network training problem."

The problem that NN solve can be described as a function approximation problem. Neural networks use the training dataset to try to learn to approximate an unknown mapping function. The way to do this is by learning weights and the model parameter for a specific network design. The training is iterative and the same process happens many times. Each specific set of weights is executed with a number of examples from the training dataset and a model error is calculated, often referred to as loss. The loss function is used to evaluate a state of the model with specific weights. The closer to zero the loss function gets the better performance the neural network has. Each training iteration uses a number of examples from the training dataset and that number is called batch size which is highly dependent on the problem dealt with.

In each cycle of the learning algorithm, a learning rate is defined. That rate controls how much to update the model weights on each iteration. It is said that an epoch has been concluded once a whole cycle is done and a complete pass through the training dataset is performed. We can also set the number of epochs that our model will execute. Usually, at the end of the epoch we assess the performance of the model by looking at the loss function. The required number of epochs can vary a lot according to the problem and is often set after training and assessing the model on each epoch by checking the loss function. Once there is no improvement in the loss function for a number of epochs, the model has reached a good level of training.

Before deciding the learning model used, we needed to analyze and characterize our problem. Since our aim is to predict future $(x, y)$ and $(x, y, z)$ positions for the detected person, we are dealing with a regression problem. The artificial neural network chosen for our learning architecture is a Long Short-Term Memory (LSTM) network (HOCHREITER; SCHMIDHUBER, 1997). LSTM is a type of Recurrent Neural Network (RNN) quite successful in many applications that deals with regression. In the field of language modeling and prediction, LSTM became the leading approach due to its robustness and success rate to deal with time series problems. Since our problem falls into time series category, LSTMs are used. Keras framework from (CHOLLET et al., 2015)

is used in our work. It implements an LSTM described in (GERS; SCHMIDHUBER; CUMMINS, 1999). The internal structure can be seen in Figure 2.2 from (ALTCHE; FORTELLE, 2017).

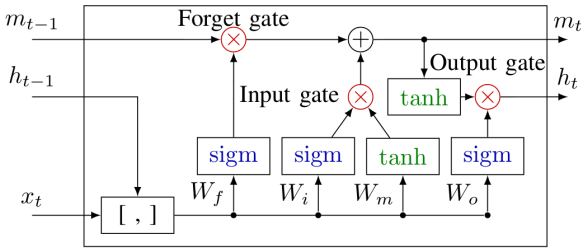
One of the key features of LSTM is the cell's memory represented in Figure 2.2 by $m_t$. The LSTM cell uses different "gates" (operations) based on the input $x_t$, the memory of the previous state $m_{t-1}$ and the previous output $h_{t-1}$. Based on $x_t$ and $h_{t-1}$, the input gate will decide how much information will be stored in the memory. The forget gate is responsible for deciding how much to forget from previous internal state $m_{t-1}$ based on the input. The output gate creates a new cell output based on the input gate and a number of previous state outputs. This robust architecture and its features allow the LSTM to correlate long-term relations between the data and perform learning of long sequence patterns. For this reason, this model is considered to be powerful for time series prediction.

As presented in (BROWNLEE, 2018), we can have 3 different LSTM designs: one-to-one LSTM - input and output of size one; many-to-one LSTM - input sequence of size $n$ and output of size one; and many-to-many LSTM - input and output of size $n$. We experimented with both many-to-one and many-to-many LSTMs. During the development of the models, we designed a many-to-one LSTM and used the output of the previous timestep as input to the next timestep in an interactive way. Having one prediction per time step, we end up with several size 1 predictions performed sequentially to create a series prediction. The problem encountered with this method was error propagation. Feeding the output of timestep 1, that had a very insignificant error as input to other timesteps, caused a much higher error along future timesteps.

We found that this particular problem was not present in many-to-many LSTM design due to the Time Distributed wrapper provided by Keras framework. Using this wrapper we input a sequence of size $n$ and receive as output a sequence of size $n$ as well. It applies a layer to every temporal slice of an input sequence and outputs equal dimension predicted sequence.

With the goal of comparing LSTMs in a 2D space and a 3D space, we developed 2 LSTM models that differ in the input dimension. Our sequential model is composed by a layer of 64 LSTM cells configured to return sequences rather than single values(Keras $returnSequences$ argument set to $True$), followed by a Dense layer with the dimension of each element in the sequence outputted. For 2D-LSTM the dimension of the dense layer is 2, and it is 3 for the 3D-LSTM. Mean square error loss function, ADAM optimizer,

Figure 2.2: Representation of the LSTM implemented by Keras. The activation functions used are sigmoid(sigm) and hyperbolic tangent(tanh). The previous output is concatenated in [,] with the current input. From (ALTCHE; FORTELLE, 2017).




and linear activation function were used.

We employed experiments with distinct camera angles and trained different models for the walking action. Besides that, we employed evaluation protocols often used in the literature that splits training, validating and testing data. Each person walking is referred to as a subject; in particular, subjects S1, S2, S3, S4, S5 were used for training, while subject S6 was used for testing. Since training and testing data are from different subjects the models never had contact with the particular person that it is trying to predict.

LSTM usage has been rising significantly and has been used in several different contexts, presenting great performance in the area of natural language processing(NLP). It has been applied in machine translation from two different languages treated as a sequence to sequence learning problem known as *seq2seq* as it was done in (SUTSKEVER; VINYALS; LE, 2014). It was used in the context of image captioning where a frame is interpreted and a text describing its content is outputted, as it is performed in (VINYALS et al., 2015). A similar approach is done in (VENUGOPALAN et al., 2015), where a description of events happening in a video is outputted using the LSTM. It was used as well in handwriting generation in (GRAVES, 2013) where results appear to be from humans writing.

Many of language-related problems solved with LSTMs are interpret as a seq2seq problem. One sequence is received and another is outputted. In the language context, we know that the sequences are made of character, so we deal with discrete data. However, LSTM power to learn what to remember and what to forget also works with continuous data. The main difference resigns in the problem often being of regression nature when it comes to continuous data and a classification problem when looking at discrete data, such as text. Usually, LSTM in the context of NLP uses softmax as activation function whereas in the context of path prediction sigmoid or linear activation functions are used.

## 2.2 Related Work

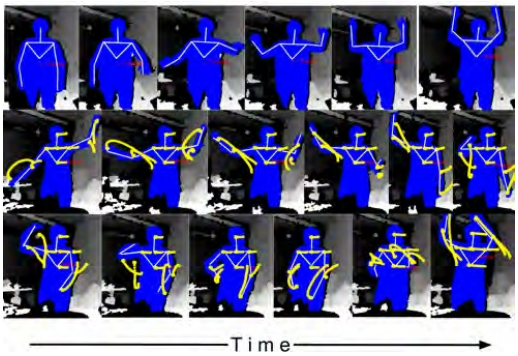
Predicting human movements using neural networks is a task that faces several challenges. A large number of approaches have been presented to better predict human movements. A recurrent goal in the papers addressed here is Human Robot Interaction (HRI). There are three different results aimed by the neural networks in this context: predicting future poses, predicting path and predicting/classifying action. Since there are similarities in the data collection, training, feature extraction and model, work related to those three approaches will be considered. In this chapter, we will also exhibit papers related to time-series predictions that use RNN-LSTM from other contexts such as car movement prediction, text modeling/prediction, between others. Even though the contexts are distinct, the approaches used in such projects are similar to ours due to the similarity in the data.

In (BÜTEPAGE; KJELLSTRÖM; KRAGIC, 2017), an approach for HRI using conditional variational autoencoder is presented. It predicts a window of human motion based on past frames. It uses skeletal data obtained from RGBD depth images and can predict up to 1.660 seconds into the future. Each key point is composed of a $(x, y, z)$ coordinate and a rate of 30 frames per second was used for the recordings. Besides that, 80 minutes of recordings were used for the training and 10 minutes for the validation.

According to (BÜTEPAGE; KJELLSTRÖM; KRAGIC, 2017), one of the challenges is that there is a lack of an accurate model of natural human motion. They attempted to use only 8 body keypoints representing the torso, the arms and the head, combined with 4 models. All joints are translated into a local reference frame that had the root joint (head) as the center. There is one model for the root joint, another for the torso with 4 keypoints, and one more for each arm with three keypoints. The data recorded represents a person standing in front of a camera and performing an action, but the person does not walk around. It is possible to see in Figure 2.3 the results acquired in the bottom rows. The top row is an example of the training data and the second and third show a sample of future trajectories predicted. They create models to predict human movements, but they do not predict the path of the person walking like our approach.

The evaluation method used in this project is the average of the motion prediction error (MPE): in each timestep, a mean square error is taken on all tests and summed over each joint. Since we do not predict all body joints, we will be using a different methodology.

Figure 2.3: In the first row of the image we have an example of training data in RGB depth image. The second and third show a sample of future trajectories predicted. Image extracted from (BÜTEPAGE; KJELLSTRÖM; KRAGIC, 2017)

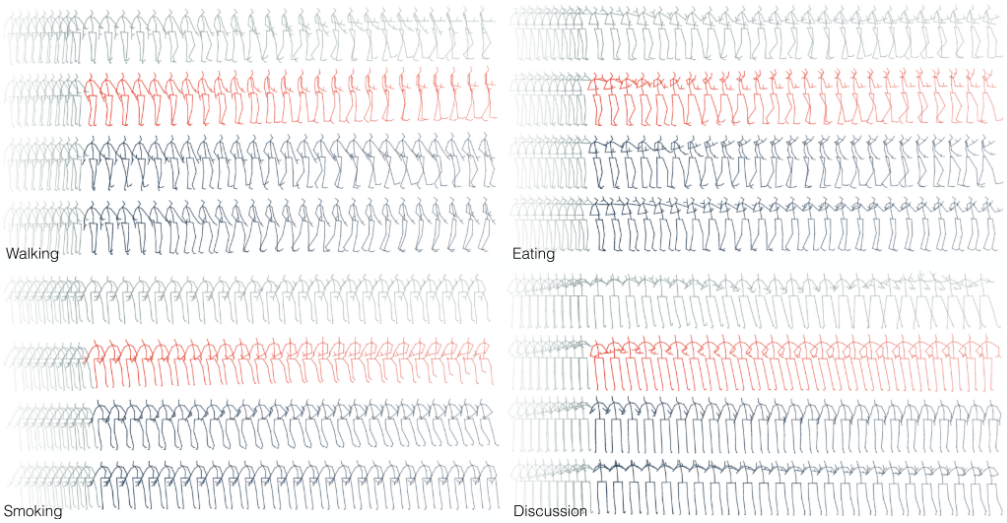



That project differs from ours on significant points, specially because it predicts keypoint movements of a person in the same place, while we predict the movement of the person walking. It uses RGB depth images while we use RGB images. It records with a 30 FPS rate, while our experiments are performed with only 6 FPS and the data from the *Human3.6m* dataset from (IONESCU et al., 2014a) present in some experiments uses 50 FPS. In addition, the models are distinct.

(MARTINEZ; BLACK; ROMERO, 2017) shows a state-of-the-art method on human motion prediction for actions such as walking, eating, smoking and discussing. One surprising conclusion from the paper is that a simple baseline approach can achieve state-of-the-art performance, which means that no motion modeling has to be done. Creating a model that describes correctly physical limitations and intentions of subjects is a complex task. Making the model learn from observation is an ideal form of modeling and learning. The paper uses 3D poses in order to learn the model of human motion and predict future poses based on past ones. It uses Gated Recurrent Unit (GRU) instead of the usual LSTM employed for this problem. Gated Recurrent Unit is a simplified version of LSTMs, but its internal structure is simpler - which means it is faster to train. It tests on *Human3.6m* dataset (IONESCU et al., 2014b) with the actions: discussing, eating, greeting, phoning, posing, purchasing, sitting, sitting down, smoking, taking a photo, waiting, walking, walking a dog and walking together. In Figure 2.4 it is possible to see the results of different algorithms compared with the ground truth (top row).

This work differs from ours for using RGBD instead of RGB and predicting poses instead of paths. The evaluation methodology used is a measure of the euclidean distance between ground truth and prediction in angle-space. Since we are working with 2D we will not be using that evaluation methodology. In some experiments, we also use the data

Figure 2.4: This figure represents 2 second motion generation by different algorithms. The top row is the ground truth, the second line is SRNN and the bottom rows present two variations of the algorithm presented in the paper. Image from (MARTINEZ; BLACK; ROMERO, 2017)

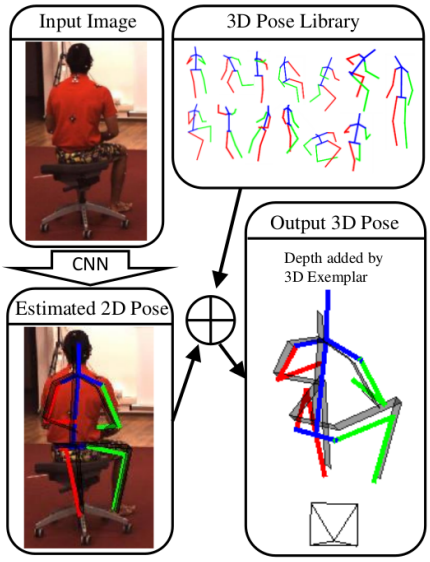



from dataset *Human3.6m* but we focus mainly on the action walking and predict the path.

The state of the art work from (CHEN; RAMANAN, 2017) presents an approach to estimate 3D body pose with a single 2D RGB image of a person and is used in one of our experiments. The fundamentals of this approach were shown in (LEE; CHEN, 1985) that tried to infer the 3D joints from their 2D projections. In (LEE; CHEN, 1985), it was discovered that given the lengths of the bones, it becomes a binary decision tree and each branch represents two possible joint states relative to its parent. Observing joint constraints, the tree is pruned but many options were still possible. Following this approach (JIANG, 2010), created a large pose database and using nearest neighbor queries was able to solve many ambiguities. Finally, (CHEN; RAMANAN, 2017) was able to reach a state of the art approach by creating a library with tuples of estimated 2D poses with known 3D poses. After this, a nearest-neighbor search was used to match 2D pose wanted with the 2D pose from the library. The result is the 3D pose from the tuple that has the highest matching with the 2D pose wanted.

As shown in Figure 2.5, the NN receives an image of a person and from that, a 2D off-the-shelf pose estimator can be used (such as Openpose) to get a 2D pose represented by 14 body keypoints. The real performance improvement from this work comes from the use of a 3D pose library. This library has 3D Poses acquired with motion capture techniques that are very accurate. For each 3D Pose, a corresponding 2D pose projection is created so the 3D library is filled with (3D, 2D) tuples. Each element of the tuple corre-

Figure 2.5: This image is the overview of the 3D pose estimation approach from (CHEN; RAMANAN, 2017). First, it estimates a 2D pose from an image and uses matching on a library of 3D poses to estimates the depth. The gray skeleton represents the ground-truth and the prediction is represented by the colored skeleton.

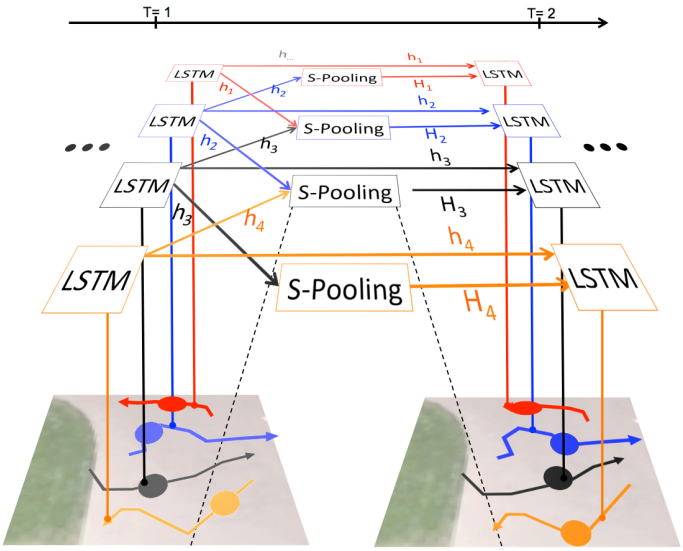



sponds to the same person described in two dimensions and three dimensions. Having this equivalence makes it possible to use matching of a 2D pose input with a tuple of (2D,3D) Pose. This is the technique employed to acquire the 3D poses used in our project. Without this, we would not be able to estimate the 3D data of a person in order to predict the path in the 3D space.

Another field of research related to human trajectory prediction is the prediction in crowded spaces. Predicting humans walking in this context focuses on the impact that humans have on each other when walking close by. This is focused on multiple people trajectory prediction. In (ALAHI et al., 2016), an approach is presented taking into account social norms that pedestrians follow, either to avoid obstacles or to accommodate and keep a distance from a fellow pedestrian. It is interpreted as a problem of sequence generation. It uses an LSTM to learn general human movement and predict the trajectories. Some approaches predict long sequences with LSTM but the dependence between multiple correlated sequences is not captured by other methods while Social-LSTM tries to solve that. It uses two data sets - ETH from (PELLEGRINI et al., 2009), and UCY from (LERNER; CHRYSANTHOU; LISCHINSKI, ) - to test and compare results. In their model they use a separate LSTM for each trajectory in the scene, meaning that each person has a different LSTM, and the networks are connected through pooling so the layers can share information. They compare results with and without pooling and concluded that pooling yields better results. For each person in the scene, xy-coordinates are used

to describe the trajectory and a prediction of a certain length is outputted. The model is composed of an embedding dimension of 64, spatial pooling size of 32, hidden dimension of 128 for all LSTM models, embedding layer with a rectified linear unit (ReLu). The general structure of the model can be seen in Figure 2.6. The average crowd density for the experiments was 30 people per frame. The experiments are taken from a top view of the people walking.

Figure 2.6: This image shows the structure of Social-LSTM. Each trajectory has its own LSTM and a pooling layer connects the LSTM layers, so information of close trajectories can be exchanged. Image from (ALAHI et al., 2016)




It uses 3 metrics to evaluate the predictions, average displacement error (ADE); final displacement error (FDE); and average non-linear displacement error (NL-ADE). These metrics are explained in Section 3.7. A 2.5 frame rate is used, and 8 frames are observed to predict the next 12 frames. Since we have a similar input and output result, the metrics used in this paper are the same ones employed in our research. The main difference from this paper is that we only look at one person and the camera is not far away from the person walking.

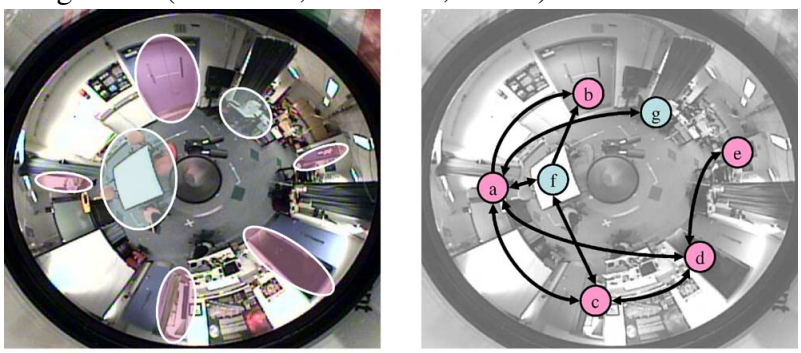
In (XU et al., 2018), a similar approach was taken but focused on collision avoidance, a similar LSTM structure from (ALAHI et al., 2016) but the pooling layer is a repulsion pooling layer focusing on avoiding collision by using a function that simulates repulsion. It also uses ADE, NL-ADE, and FDE as metrics to evaluate the performance as used in our project. It uses 8 frames to predict the next 12. After running experiments of Social-LSTM, Collision-FreelLSTM, a simple LSTM, Linear trajectory avoidance and Social Forces, the best results were acquired with Collision-FreelLSTM closely followed

by Social-LSTM. This shows that the pooling layers have a good impact on the final result.

Figure 2.7: It is possible to see 3 cases of prediction using 3 different methods. The ground truth is represented in yellow. Collision-FreeLSTM is presented in red, Social-LSTM in green, and Social Forces in blue. Image from (XU et al., 2018).

In Figure 2.7 from (XU et al., 2018) it is possible to see prediction results of Collision-FreeLSTM technique (in red) being compared with Social-LSTM (in green) and Social Forces (in blue) technique. The ground truth (in yellow) is also shown. The best prediction comes from Collision-FreeLSTM. The people recorded are very distant from the camera so in 12 frames they have a small movement as it is possible to see in the image. Whereas in the data used in our experiments the movement is quite significant: the person walking is very close to the camera so a great distance is covered in a few frames.

Figure 2.8: a) Points of Interest and b) Action Paths are presented for trajectory learning experiments. Image from (MORRIS; TRIVEDI, 2008b).




(a) (b)

In (MORRIS; TRIVEDI, 2008b) a survey on different vision-based trajectory learning is presented for the context of surveillance. The technique used and discussed is successful for trajectory learning in many different contexts. It is based on the definition of points of interest (POIs) and activity paths (APs). The POIs are areas of the image in which subjects perform actions or in which they enter or leave the scene. APs are the path taken for each activity. In Figure 2.8 it is possible to see an example of POIs and APs of a

situation discussed on (MORRIS; TRIVEDI, 2008b). This is the approach used by us in one of the experiments that can be seen in Section 5.2.

# 3 METHODOLOGY

This chapter presents the methodology as well as the work performed in this project. The main frameworks used in this project were Robot Operating Systems (ROS) (QUIGLEY et al., 2009), Openpose (WEI et al., 2016) (CAO et al., 2017) (SIMON et al., 2017), Tensorflow (ABADI et al., 2015) and Keras (CHOLLET et al., 2015). Python (Python Core Team , 2019) was used to develop the NNs.

Some work has been done towards the setup of the software and hardware. Kinect 2.0 was used to capture the data and had to be calibrated prior to the data collection. Also, short experiments were done as proof of concept using techniques that intended to be used in future parts of the project such as mouse prediction experiment in Chapter 4.

Main experiments that trained, tested and predicted the path of the people walking as well as Openpose conversion from RGB frames to bodyjoint coordinates were performed in the same equipment. It was an ASUS laptop with Intel Core i7 7500U, with 2.7GHz, RAM memory of 8 GB, Nvidia GEFORCE 930MX Graphical Card with 2Gb dedicated RAM, running Linux Ubuntu 16.04 operating system. The process that converted a 25 bodyjoint into a 14 bodyjoint represented by step 6 of Figure 6.3 was executed in another machine. This machine was also used for step 8 and it was a Positivo computer with inter Core i5-3470 3.2GHz, RAM 4GB, running Linux Ubuntu 16.04.

This chapter begins dealing with machine learning technologies, presents the framework and equipment used, shows the data collected, the dataset created and the external dataset used, and ends with the evaluation metrics used.

## 3.1 Machine Learning Methodology

The methodology followed was the one presented in (FRANÇOIS, 2008). This work presents a well-defined methodology and standards for projects related to data and machine learning. It compares 4 different methodologies directed to machine learning project and extracts the common steps to create guidelines. The compared methodologies are Fayyad from (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996), Cios from (CIOS; KURGAN, 2005), SEMMA from (INSTITUTE, 2008), and CRISP-DM from (CHAPMAN et al., 2000). The comparative result showing which steps are performed by each project can be seen in Table 3.1. We also outline our approach and the steps present in our work.

Table 3.1: All steps involved in four proposed machine learning methodologies from (FRANÇOIS, 2008). We also show the steps used in our approach.

| | Step | Fayyad | Cios | SEMMA | CRISP-DM | Our Approach |
|---|---|---|---|---|---|---|
| 1 | Objective determination | **X** | **X** | | **X** | **X** |
| 2 | Data collection | **X** | **X** | **X** | **X** | **X** |
| 3 | Data cleansing | **X** | **X** | **X** | **X** | **X** |
| 4 | Data reduction | **X** | | **X** | **X** | **X** |
| 5 | Problem reformulation | **X** | | | | **X** |
| 6 | Data exploration | **X** | | **X** | | **X** |
| 7 | Tools selection | **X** | | **X** | | **X** |
| 8 | Model construction | **X** | **X** | **X** | **X** | **X** |
| 9 | Model validation | **X** | **X** | **X** | **X** | **X** |
| 10 | Result interpretation | **X** | **X** | **X** | **X** | **X** |
| 11 | Deployment | **X** | **X** | | **X** | **X** |

Our project was designed according to the steps described in Fayyad methodology. Since it is able to break down the methodology in modular parts and has a step of problem reformulation that happens after collecting, cleaning, and getting in touch with the data that was necessary for our project. Each step of our work is described individually considering those 11 steps:

- **Objective determination:** In this step, we determined the goal of our project. Initially, it was to predict the movement of multiple people walking in a controlled environment with RGB and RGB-D. With this data, we would have a 3D representation of a person walking in a 3D environment. After this, ML models would be applied to predict the path in the 3D and 2D space. The objective was redefined in future steps.
- **Data collection:** The data collection was performed in a controlled kitchen environment and RGB and infrared (IR) data were captured. After exploring and experimenting with this data, additional experiments with data from *Human3.6m* dataset were made. The data collection process was performed with the consent of subjects involved and is better explained in subsection 3.3
- **Data cleansing:** After collecting the data it was necessary to look at it and discard the irrelevant information, check the quality and correct the discontinuities. This was a step that had a lot of influence on the final problem definition that changed in the problem reformulation step. Data cleansing was performed several times during the whole process since the data has a huge impact on the training of the model and the results. A few problems were found after looking at the data such

as the occlusion of a person walking behind objects; the data discontinuity happens due to a few miss detection so data suddenly drops to 0 as can be seen in section 7.1 and this had a significant negative impact on the learning of the model; the lack of tracking of Openpose made it hard to distinguish data from one person to another, problem described in subsection 3.3, so instead of multiple people in the scene only one was used; the RGB data had good quality and it was possible to detect the person in the whole area of the environment, but the infrared had a limited range, shown in Section 7.3, in which it could detect the person walking so the problem reformulation step was influenced by that as well. The paths walked by the subjects were long and wide, and were outside the IR range. This problem can be seen in section 7.3. IR data was not used in the final experiment and only RGB data had good enough quality to be used.

- **Data reduction:** In this step, the data had to go though a few adjustments. To correct short discontinuities we used interpolation described in subsection 7.1. We had to be sure that the data was being read sequentially after being recorded, and using ROS assured the original order of frames which is paramount for time-series prediction. Another process related to this step was the normalization of the data before using as input in the model. The data had to be normalized to assume values between 0 and 1. $x$ $y$ and $z$ were normalized according to the frame width, height, and depth of the scene. After the training, the data had to be de-normalized so the prediction would assume real-world values.
- **Problem reformulation:** This step was reached a few times, and problem reformulations usually had a strong relation with data issues. Because of the infrared short range, explained in 7.3, we stopped using IR data and only used RGB data. At this point, we were dealing with a 2D path prediction problem that only used $x$ and $y$. We also reformulated the problem and did experiments with 3D estimated data using the dataset *Human3.6m* and compared path prediction using 2D data with path prediction using 3D estimated data.
- **Data exploration:** In this step, we analyzed the data captured and took relevant decisions regarding the possibility of using the data. We plotted the data and looked at it from different angles. Looking at the data we realized that a wrong detection was happening, people that appeared from time to time in the television that was present in the scene were recognized and disturbed the results. This problem is described in subsection 7.2. Data limitations were also discovered in this step.

- **Tools selection:** At this point, we made tests and decisions regarding what tools to use for the creation of the model and experiments. The development environment used was Pycharm (PYCHARM, 2016), which made it possible to debug the code and to look at the data during the execution. The system used to collect and extract the data was ROS. Keras 2.2.4 on top of Tensorflow 1.12.0 was the framework chosen to create and experiment with different models reaching the decision of using LSTM. Both Many-to-One LSTM and Many-to-Many LSTM were used.
- **Model construction:** The model construction was a step of heavy load programming and tests with distinct models were done using different algorithms, data configurations, shapes, and sizes. The 3 main models used were a Many-to-One 2D-LSTM, Many-to-Many 2D-LSTM, and Many-to-Many 3D-LSTM better explained in the background section 2.1.
- **Model validation:** The data was split in different ways depending on the case. Some experiments were trained, tested and validated with different data of the same subject, while others were trained with data of one subject and tested with data from other subjects. All experiments had a split of 60 % training 20 % validation and 20 % testing. In some cases experiments were done in a way that data from one subject was used to train and data from other subject was used to test. The upside from this approach is that the test data is from a subject never seen by the neural network. Other experiments used data from the same subject but from different paths. Data from the testing, validation, and training never overlapped.
- **Result interpretation:** Our results were interpreted according to measurement metrics that compare the predicted path with the ground truth. The metrics employed were FDE and ADE as used in papers related to path prediction of people. More on the evaluation metrics can be seen in subsection 3.7. It was also possible to plot the results and asses whether the predictions are good or not in a qualitative way.
- **Deployment:** The preprocessing is computation-heavy and could not be done in realtime with the resources available. Therefore, our models were tested only with prior recorded data, meaning that no model was used with realtime data. Given that the appropriate equipment was available it would be be possible to apply the models in realtime. In that case, significant changes would have to be done in the architecture of our approach.

## 3.2 Kinect Calibration

Before starting to work with Kinect, it was necessary to calibrate the device. The data collected were RGB and infrared (IR) images. In order to capture infrared (IR) data, a calibration of the Kinect camera to define distortion coefficients was necessary. Kinect has one RGB camera, one infrared camera, and IR projector. Both cameras have to be calibrated separately as well as together. There are 3 kinds of calibration that had to be done: RGB camera calibration, Infrared camera calibration, and synchronization between the two cameras. The calibration needs to be done only once and has a distortion coefficient matrix as a result. Calibration is needed due to lens distortion and is more common in wide-angle lenses. When precision is needed, calibration is paramount.

In (OPENCV, 2017) the theory used for the calibration is presented. OpenCV takes radial and tangential distortion factors into account. The radial distortion factor is calculated in equations 3.1 and 3.2. Therefore, for a given $x$ and $y$ pixel point in the undistorted image, we will have $x_{Rad_{distorted}}$ and $y_{Rad_{distorted}}$ in the distorted image.

$$x_{Rad_{distorted}} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.1}$$

$$y_{Rad_{distorted}} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.2}$$

Tangential distortion happens when the plane of the image being taken is not parallel to the lenses - which is the case for the data dealt with in this work. Therefore, the tangential distortion factor can be calculated with the equations 3.3 and 3.4, which are used to reach the distortion coefficient matrix in equation 3.5.

$$x_{Tang_{distorted}} = x + [2p_1 xy + p_2(r^2 + 2x^2)] \tag{3.3}$$

$$y_{Tang_{distorted}} = y + [p_1(r^2 + 2y^2) + 2p_2 xy] \tag{3.4}$$

$$distortion\_coefficients = (k_1, \quad k_2, \quad p_1, \quad p_2, \quad k_3) \tag{3.5}$$

A camera matrix is used to create a correlation between physical world units and pixels. The optical centers $c_x$ , $c_y$ and the focal lens $f_x$, $f_y$ are calculated and used in the

matrix multiplication equation 3.6 to make a conversion between units.

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.6}$$

The process of calibration calculates those two matrices. A chessboard was used for the calibration and several images were captured. The chessboard was positioned in different locations and angles, new patterns are found with every new image and the matrices are updated. This is made to cover the majority of the image and correct any possible distortion in several areas of the image. Roughly, between 100 to 150 frames were used in the calibration performed.

Part of the process can be seen in Figure 3.1. In the 4 pictures in Figure 3.1, it is possible to see many patterns being recognized by OpenCV due to different positions and angles that the chessboard is in. The matrices are updated at each new image. In this process, it is necessary to gather a lot of images with diverse distances from the focal points and distinct angle so the calibration can be done properly. The practical process of calibration followed is described in (WIEDEMEYER, 2017).

Figure 3.1: Chessboard being capture in different angles and recording different patterns to update the camera matrix and the distortion coefficient matrix.

Even though the calibration process was made and the IR data was captured, it was not possible to use the IR data due to the long distances of the paths recorded. The range of the IR camera is much different than the RGB camera. The Kinetic depth range goes from 0.8m to 4m and the path taken by the subjects exceeds this range. This is better explained in Section 7.3.

### 3.3 Dataset Collected

After the process of calibration, the data that Kinect provided was more reliable, so it was possible to start the data collection to create our dataset. The data collection was made by using ROS to capture and store the data in rosbag format. All files were kept in external hard drives. In order to be able to perform diverse experiments, many types of data were captured from the same experiment. Also, distinct experiments and data extraction were performed. The data captured was RGB and infrared(IR). The RGB images had 960x540 resolution. The depth images had 623x537 resolution. Having these data combined with the keypoints of the human body, it is possible to create more robust complex prediction models. The frame rate used was 6 FPS and it was used partially because of computer limitations since higher FPS caused memory problems given that only 8 GB RAM was available. The frame rate chosen is expected to be enough to record and analyze the data for our goals.

ROS framework was used to collect and store the data in `.bag` format. When using the Rosbag tool, it is possible to record from and playback to ROS topics. It allows high performance and avoids de-serialization and re-serialization of the messages. Meaning that the original order of the messages will be preserved when the playback of the data is performed.

After recording the data, each experiment is converted from .bag to video format. The video is then inputted into the Openpose system. At this point, we have two types of data: the keypoints presented in JSON and the original image with the human joint detection plotted on top of the respective joints.

Several issues were encountered while collecting the data and some modifications in the original project had to be done. Having more than one person recorded at the same time was a problem since Openpose does not have people tracking. The output has the format described in Section 3.6. This means that when reading the JSON file that represents human keypoints it is not possible to keep track of which keypoints belong to which person through the time. It is possible to see in the snippet below the format outputted by a frame with two human bodies detected. In this case, throughout the frames, it is not possible to set the two people apart without additional tracking techniques. To focus on the path prediction problem, we decided to perform experiments and collect data of a single person walking instead of applying individual tracking. Applying tracking techniques would bring new issues such as people crossing paths or being behind each

other. In order to focus on the NN model, only datasets with one person at a time were used, and the data recorded also had one person at a time present in the scene.

```
{
  "people": [ {
     "pose_keypoints":[842,322,0.885,878,322,0.848...], }],
    [ {
     "pose_keypoints":[438,151,0.827,451,156,0.830...], }]
}
```

All data from our dataset was recorded in a kitchen environment. The data description can be seen in Table A.1 in the appendices section. Experiments were performed by a few people, but the majority of the data recorded was performed with only one subject that was available for most recordings. We had 4 data collection setups. In the setup number 1 and 2 initial data was recorded containing people walking in the kitchen randomly and performing everyday actions at will. No particular goal or path was taken and the data was not labeled. That data was used in the initial experiments described in Section 5.1. The data was stored in .rosbag files the setup 1 had size 5.21 GB and setup 2 had size 26.03 GB.

The data from setup 3 was recorded envisioning experiments such as the ones presented in Figure 2.8 from (MORRIS; TRIVEDI, 2008b) in which points of interest (POIs) are defined and activity paths are performed between the POIs. We recorded people walking to specific points of interest in the kitchen area and performing some action. The actions were: sitting on the back table (bt), sitting on the side table (st), going to the coffee machine (cm), and going to the espresso machine (em). The paths towards the actions started on other points of interest of the kitchen: the bottom left (bl), the bottom right (br), the top left (tl), and the top right (tr). Activity paths were recorded to and from the point of interest, and each path was performed 5 times, therefore each activity path was observed at least 5 times. Each path was labeled with the final POI. The points of interest can be seen in Figure 5.3 and a few of the activity paths taken between the points of interest can be seen in Figure 5.4. In setup 3 a total of 38.04 GB were recorded. Setup 4 had 43.46 GB recorded but was not used in any experiment in this project and can be used in future work.

From the rosbag files, we run Openpose on the data and acquire more detailed and descriptive data in the form of a human pose. That data is preprocessed before being used in the experiments. In each frame, we have 25 bodyjoints that are $x$ and $y$ coordinates and

describe the body of the humans detected in the image. With this data, many actions can be performed. Once we have two or more sequential frames we can calculate the velocity of each joint in the frame. We can use the pose to infer the 3D pose and the depth of the person, as we do in section 6.1, and we can recreate the path of the person detected. The data from this dataset was used in a few experiments so far but many relevant experiments can be designed with the dataset and are part of our future work.

### 3.4 Data collection Approval

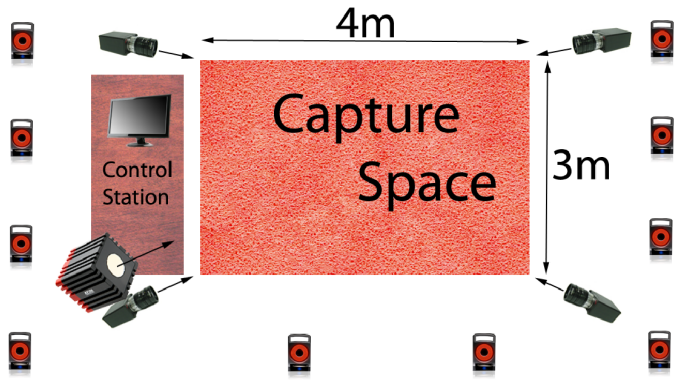
In order to collect the data from people, it was required to apply for a data collection approval with Norsk Senter for Forskningsdata (Norwegian Center of Research Data). After this approval, a form was created and can be seen in Appendix B. It is important to mention that the data was used only from the users that read, consented and signed the form.

### 3.5 Human3.6m Dataset

*Human3.6m* is a 3D Humanpose dataset widely used for bench-marking 3DPose algorithms created by (IONESCU et al., 2014a). It contains video of 11 human actors performing several actions in a controlled indoor environment. Some of the actions are walking, sitting and phoning, walking a dog, between others, but the only that interests us is the walking scenario. The scenes are captured from 4 different camera angles at the same time at 50 frames per second. We can see camera arrangement on the environment in Figure 3.2. The space is 4 by 3 meters and there are 4 cameras recording each subject from different angle. In our experiments, we distinguish the cameras by naming them A, B, C, and D. In total, *Human3.6m* contains 3.6 million frames of humans walking, with labeled poses of actors performing various tasks as well as the mocap data of each experiment. Having such a robust dataset is what makes it possible to have data-driven 3D pose estimation models trained.

In one of the approaches taken by our project, only data from *Human3.6m* was used in order to avoid problems related to data quality. When using the data collected a few data-related issues were encountered. By using this dataset we can be sure that the data has good quality and we are able to focus on experimenting with the LSTM models.

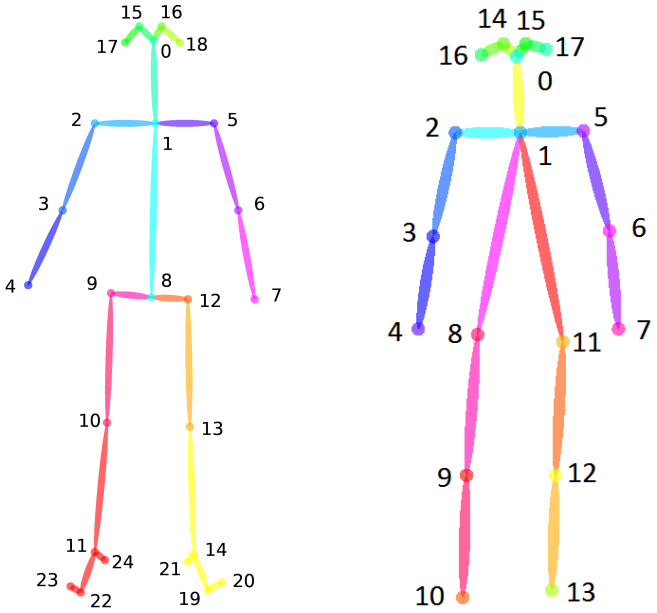
Figure 3.2: Camera arrangement of the environment in which the recording of the *Human3.6m* data was performed. We see that there are 4 cameras all recording from different points of the room with distinct angles. Image from (IONESCU et al., 2014a)




### 3.6 Openpose Data - Human Keypoint Format

Openpose outputs the human body detected as a set of keypoints and each one represents a part of the body. Depending on the version used it is possible to obtain 25 or 18 keypoints to recognize the body. Besides the body, face and hand are also detected, 20 and 69 keypoints are used for the recognition respectively. It is possible to see a representation of the keypoints in Figures 3.3, 3.4, 3.5 from (WEI et al., 2016).

Figure 3.3: Representation of the 25 body keypoints detected by Openpose on the left, and 18 body keypoints representation on the right. The number of keypoints changes depending on the version of Openpose. During the project, we used both. Images from (WEI et al., 2016)




At the beginning of the project, the latest version of Openpose used 18 body key-

points, so all algorithms were developed to fit that human body model. During the development, a new version of Openpose with higher accuracy was published using 25 body keypoints so the following experiments were adapted to use the latest version of Openpose.

Figure 3.4: Representation of the 69 face keypoints detected by Openpose from (WEI et al., 2016)

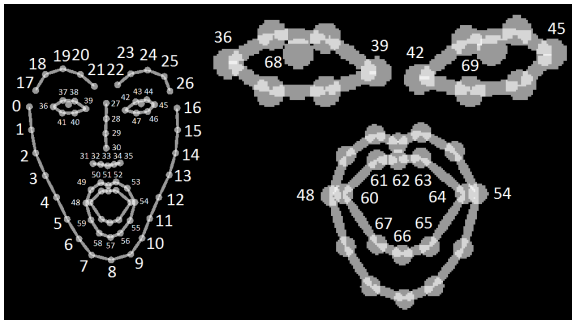
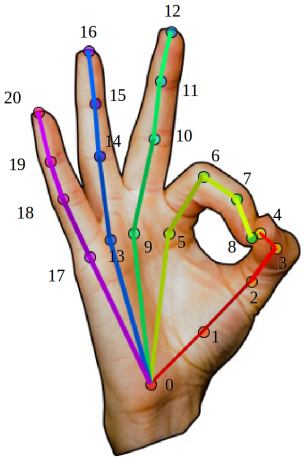


Figure 3.5: Representation of the 20 hand keypoints detected by Openpose from (WEI et al., 2016)



The data is outputted in JSON format and several flags are available to get different data formats. It this project, only the keypoints of the neck were used as shown in Figure 3.3. Each keypoint is composed of three variables: $x$, $y$, and $c$. The parameters $x$ and $y$ represent the position of the keypoint on the $x$ and $y$ axis of the image. The parameter $c$ represents the certainty of the reading. The range of $c$ goes from 0 to 1 while $x$ and $y$ will be in a range from 0 to the height and width of the image. The images have a height of 540 and width of 960. The parameters $x$ and $y$ are outputted with values from 0 and 960, and 0 to 540, respectively, and are normalized before being inputted into the NN model, assuming values between 0 and 1.

### 3.7 Evaluation Metrics for xy-coordinate trajectory prediction

The metrics used to evaluate the results of our prediction are the same as those used in (ALAHI et al., 2016) and (XU et al., 2018) for path prediction in crowded spaces. They compare the result of the prediction with the ground truth using the metrics in Equations 3.7 and 3.8. Those are the metrics:

- Average Displacement Error (ADE) : The average displacement error is the mean square error (MSE) of each predicted point of a trajectory compared to the real points. $M$ represents the number of trajectories predicted and $T_{obs}$ is the timestep of the last observed sequence, meaning that the prediction will start from the next timestep. $T_{pred}$ is the last timestep of the predicted sequence. $(\hat{x}_i^t, \hat{y}_i^t)$ represents the predicted $x$ and $y$ position at timestep $t$ and of trajectory $i$, while $(x_i^t, y_i^t)$ represent the ground truth of the $x$ and $y$ for a timestep $t$ and sequence $i$.

$$ADE = \frac{\sum_{i=1}^{M} \sum_{t=T_{obs+1}}^{T_{pred}} [(\hat{x}_i^t - x_i^t)^2 + (\hat{y}_i^t - y_i^t)^2]}{M(T_{pred} - T_{obs+1})} \tag{3.7}$$

- Final Displacement Error (FDE): The final displacement error evaluates the distance between the true and the predicted final destination in the last position at timestep $T_{pred}$.

$$FDE = \frac{\sum_{i=1}^{M} \sqrt{(\hat{x}_i^{T_{pred}} - x_i^{T_{pred}})^2 + (\hat{y}_i^{T_{pred}} - y_i^{T_{pred}})^2}}{M} \tag{3.8}$$

- Mean Square Error of $x$ and $y$ (MSE): The mean square error of $x$ and $y$ was also measured separately in order to asses the predictions in each axis.

# 4 MOUSE PREDICTION EXPERIMENT

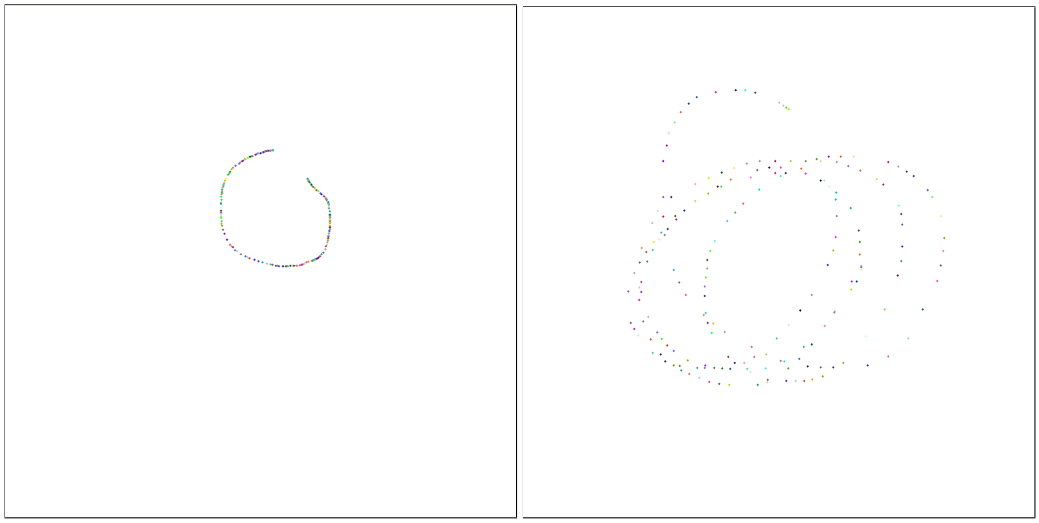
The initial results from the first models created were just random data. A lot of effort was employed into changing the model to see if the output would presented improvement but nothing changed. After following the study of (IVANOV, 2017) that presents many reasons why neural networks do not work the results started to improve. In this article, the importance of the data quality is stressed. No good prediction can come from a perfect model fed with bad data. The data can have many problems that are not easy to detect especially if there is a lot of complex data. Apart from the many different techniques and hints presented to investigate and enhance the data, a valuable suggestion is given: try solving a simpler version of the problem. This can help find where any issues are and that is what is proposed in this chapter. A simpler version of our problem is solved and accurate data is used.

After being guided by (IVANOV, 2017), it was suspected that the data extracted from Openpose was the main reason for the problems presented in the first experiments with the model. For that reason, an experiment with similar data was made. In order to acquire data without any noise, the $x$ and $y$ position of the mouse cursor were recorded. This data was used to train an RNN model with LSTM layers. Since the data from the mouse has the same shape as the data used in our walking path prediction experiments it was possible to reuse the RNN model for that problem. After verifying that the model works with the data extracted from the mouse it is possible to single out the problem and be sure that it is related with the data extracted from the walking direction prediction experiments. This experiment was built on top of an example available at the repository (MAJUMDAR, 2017) that detects the mouse movement in real time and plots the trace of it in the screen in real time. We used the parts of the program that track and plot the mouse movement and design our model with few adaptions.

First, the training process takes place, and the movement of the mouse is recorded. The more data recorded the better the predictions will be. There is one issue that must be considered while acquiring the training data: the mouse speed. The mouse speed has a significant influence since the sampling will change. If the mouse is moved fast the sample points will be far apart and if the mouse is moved slowly the sample points will be close together. This influences the prediction and will not be addressed since this experiment is done with the intention of being a proof of concept. This problem will be addressed in the main experiment.

In Figure 4.1, it is possible to see the difference in the sample size when the mouse is moving slow (on the left) and when the mouse is moving fast (on the right). The trace shown in both pictures have been taken in the same time frame of 2 seconds. This shows the difference in sampling on different mouse speed. This means that if the same movement is done at different speeds, training with one will not help the prediction of the other. Prediction is highly correlated with the speed in this case.

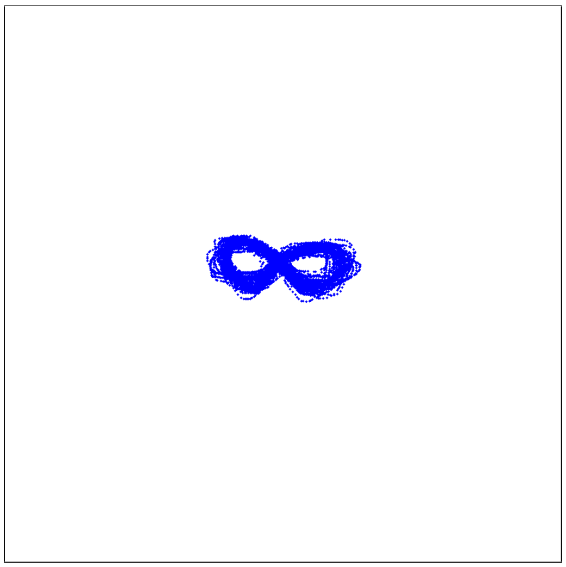
Figure 4.1: Image of the mouse movement being captured during 2 seconds: when the mouse is moving slow (on the left) and when the mouse is moving fast (on the right). This shows that even if the same movement is being done, the actual sequences will be so different that training in one speed will not help predict movements that are at a different speed.



In order to show that the model can predict future positions of $x$ and $y$, a simple experiment was made. The training was done performing repeated movements that resemble the infinity symbol. In Figure 4.2 it is possible to see the data used to train the model. In this experiment, 60 timesteps were used, and the prediction size was 12. This means that based on previous 60 mouse positions the next 12 mouse positions will be predicted. An RNN with LSTM layer with 64 cell state size and a dense layer were used. The batch size for this experiment was 6000 and the model was trained for 5000 epochs. The model follows the Many-to-One LSTM design meaning that only one point is predicted at each timestep.

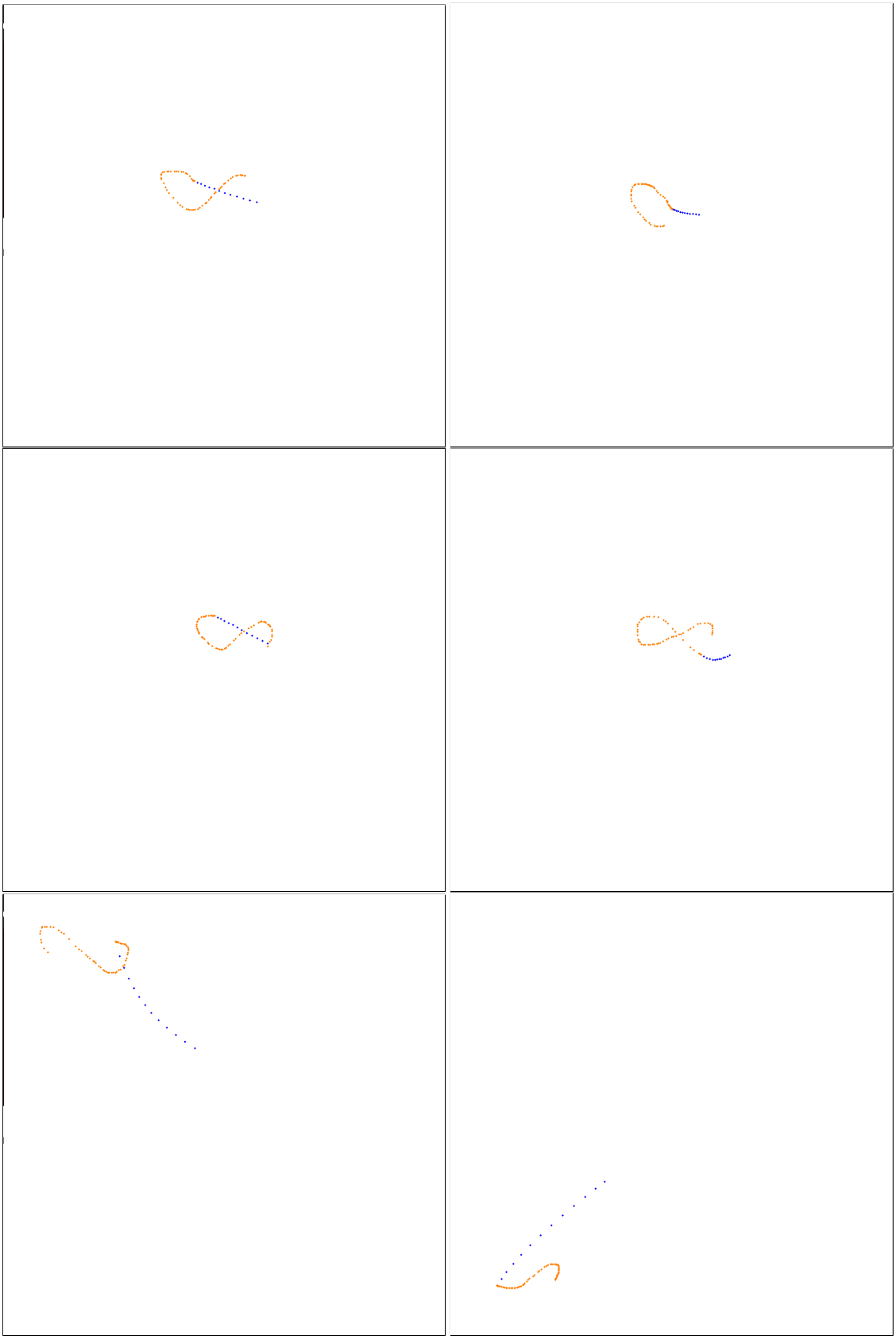
Figure 4.3 shows some real-time results acquired after training with the data presented. The predictions are based on the data that resembles the infinity symbol. For that reason, only infinite symbols done in the nearby position and similar mouse speed can be predicted successfully. As can be seen in Figure 4.3 whenever an attempt to predict the infinite symbol away from the center, the prediction has a strong bias towards the center.

Figure 4.2: The image represents the data used to train the model to predict mouse movement that resemble infinity symbol in the middle of the screen. The mouse trace is represented by the blue lines



This experiment was relevant to reach a few conclusion that can be applied to our problem. We discovered that the speed of the subject being tracked is very relevant to the problem when it comes to learning specific movements. Therefore, subjects were advised to walk with constant speed. We also discovered that the model can learn a simple path that has curved and straight parts.

Figure 4.3: Different predictions after training the model with infinity symbol. The real trace of the mouse is seen in orange and the prediction can be seen in blue.



# 5 POINTS OF INTEREST APPROACH

This chapter presents the initial exploratory approach taken towards the goal of path prediction using LSTM models followed by a points of interest approach performed with data collected during the project.

## 5.1 Initial Approach

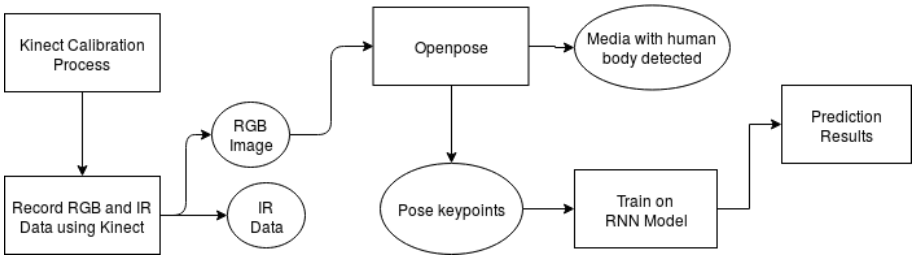
In order to work on the model, it was necessary to treat the data and have it in a format that was suitable for the model to train on as described in Chapter 3. After preprocessing the data it was discovered the limitation on the range of the infrared camera, as seen in Section 7.3. We had two different types of data describing the same event for each frame: the sequence of images of people walking and the sequence of human body poses detected by Openpose. One possibility was to use the two types of data in different models and combine them to achieve better results: the images would be fed to a CNN and the sequential keypoints would be fed to an RNN-LSTM. Since the data was redundant, only the keypoints were used and the approach focused only on the LSTM. The overview of the initial approach can be seen in Figure 5.1.

Figure 5.1 shows the steps of this approach. Initially, calibration of Kinect was needed as explained in Section 3.2; after that process, it was possible to start the data collection. The second block of Figure 5.1 represents RGB and IR data being recorded. Duo to the range of IR detection being smaller than the area of the path, only RGB is used. RGB data is then fed into Openpose. One of Openpose output is the media (sequence of frames or video) with the pose recognized and plotted on top of the human in the original picture as can be seen in Figure 2.1. The other output from openpose is the 18 or 25 body keypoints outputted in JSON format as shown in Section 3.6.

The pose keypoints are used to train an RNN-LSTM model that works with one central keypoint representing the neck of the person walking. This simplification is done to initially have a simple working model that can be redesigned and gradually become more complex. Once the training is completed we have a model that predicts the path of a person walking.

Several experiments were made to extract data from situations in which there is only one person and one keypoint per frame. Therefore, in this initial model, the data will be a sequence of $x$, $y$, and $c$ variables, and each triple represents the neck recognized in a

Figure 5.1: Schematics illustration of the general approach of the project.




frame. A sequence of size $n$ will be used as training data while a sequence of size $m$ will be used as labeled data. This approach is interactive and employs a many-to-one LSTM model design. Each iteration, i.e., each time we predict one future point, the previous $n$ points are being used. Apart from the first iteration, the future points will be predicted using results from past iterations as input.

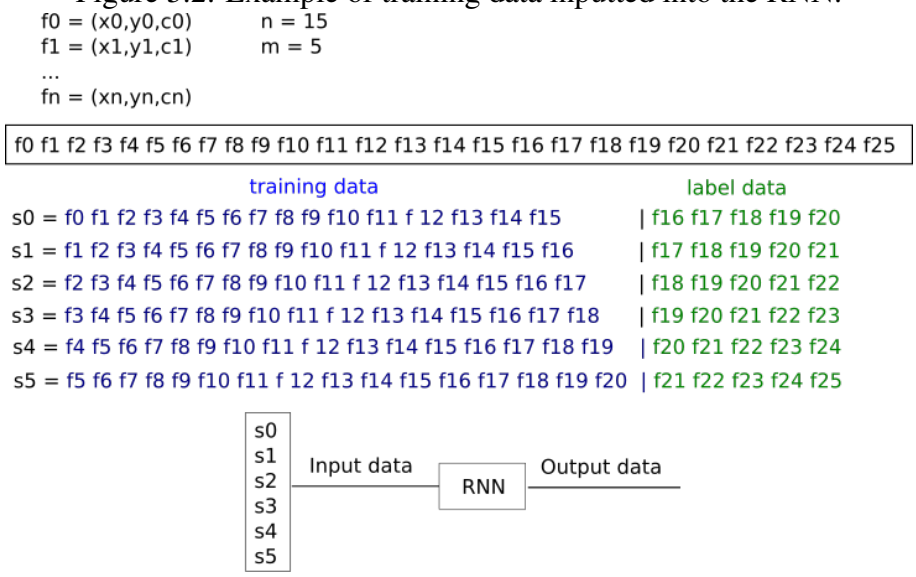
In Figure 5.2 it is possible to see one example of training data setup where $n$ assumes size 15 and $m$ assumes size 5. Each element $fn$ has $x$, $y$ and $c$ information that represents the neck keypoint. The size of the total sequence is 26. This means that with this sequence it is possible to have 6 training sets represented by the sequences $s0$ $s1$ $s2$ $s3$ $s4$ $s5$ $s6$. We can think of the training sequences as a sliding window that will use previous $n$ information to predict the following point, and then “slide" using the recently predicted point to predict the next point. The output of the model will be only $x$ and $y$ because there is no use in predicting $c$ which represents the certainty of the Openpose detection.

Since RNN with LSTM layers are capable of taking previous data into account, allowing information to persist through time, it is an appropriate neural network to use in our case. In order to predict future trajectories, several previous keypoints have to be taken into account.

It is important to have reliable data and Openpose helps to assure that. Along with every keypoint reading, there is also a certainty $c$ reading. Before considering a specific point as a valid detection, the certainty is checked and only values with certainty higher than a threshold are used. The threshold used was 0.5. It is important to realize that if the certainty is low for many keypoints in the sequence, the data will not have meaning and might have a bad impact on the final model. Verification is performed before inputting the data into the model and if the data has bad quality it is discarded.

One of our goals is to define what is the size of the initial sequence and what is the size of the predicted sequence that outputs best results. To have this information we

Figure 5.2: Example of training data inputted into the RNN.




performed experiments with a different number of frames and distinct prediction size. In order to evaluate the results, the metrics described in Section 3.7 were used.

Initially, the results were not good due to bugs, data problems and inappropriate design decision in the model as well as the shape and values of the hyperparameters. The research provided by (IVANOV, 2017) was used as a reference to debugging and correcting common mistakes in the model. Significant improvements happened after applying some techniques such as normalization of $x$ and $y$ values, verification of initial data, verification of features and labels, reduction of the learning rate, usage of linear activation function, and change of hyperparameters. We were able to reach good predictions in some cases but not satisfactory in others. We can see in the prediction results that the prediction often assumes values quite similar to the ground truth. This is shown and evaluated in Section 5.2.

## 5.2 Points of Interest Approach

In this experiment instead of collecting data of a person walking randomly, goals were created within specific areas of the kitchen environment. The design of this approach was based on the POI/AP discussed in (MORRIS; TRIVEDI, 2008b). They define Points of Interest (POI) as regions of the image in which a person performs an action or enters or leaves the scene. It also defines Activity Paths AP, and tries to learn the activity paths between the points of interest (POI/AP). An example of POIs and APs of a work discussed

in (MORRIS; TRIVEDI, 2008b) can be seen in Figure 2.8 in Section 2.2. This approach has been successfully used to learn trajectories in many contexts such as people walking indoor (MORRIS; TRIVEDI, 2008a), people walking outdoors (ROBERTSON; REID, 2005), people on a parking lot (JIAO et al., 2004), car interaction (KAMIJO et al., 2000) and on car traffic (JUNG; HO, 1999).

In this approach, one of our goals is to predict which point of interest (POI) the person was walking towards in the kitchen environment. With that in mind, new recordings were made with setup 3 from the dataset described in Table A.1. In this experiments, the paths taken by the subjects are goal-oriented, meaning they come from a specific point of interest and go to another. The areas in which a person can start and end the trajectory are shown in Figure 5.3. Every goal POI is associated with a specific action. Having the same activity path AP recorded many times improves the chances of the model learning the AP. Thus, five recording are performed from each initial/final POI to each goal POI, meaning that each action has 5 recordings starting from all initial/final POI.

Figure 5.3: This image presents the points of interest(POIs) from the experiment. The trajectories recorded are from and to one of the POIs.


The goal POIs are the regions of the image in which the subjects perform actions and they are:

- Back Table (bt)
- Side Table (st)
- Coffee Machine (cm)
- Espresso Machine (em)

The initial/final POIs are the regions of the image where the subjects enter and

leave and they are:

- Coming from bottom left (bl)
- Coming from bottom right (br)
- Coming from top left (tl)
- Coming from top right (tr)

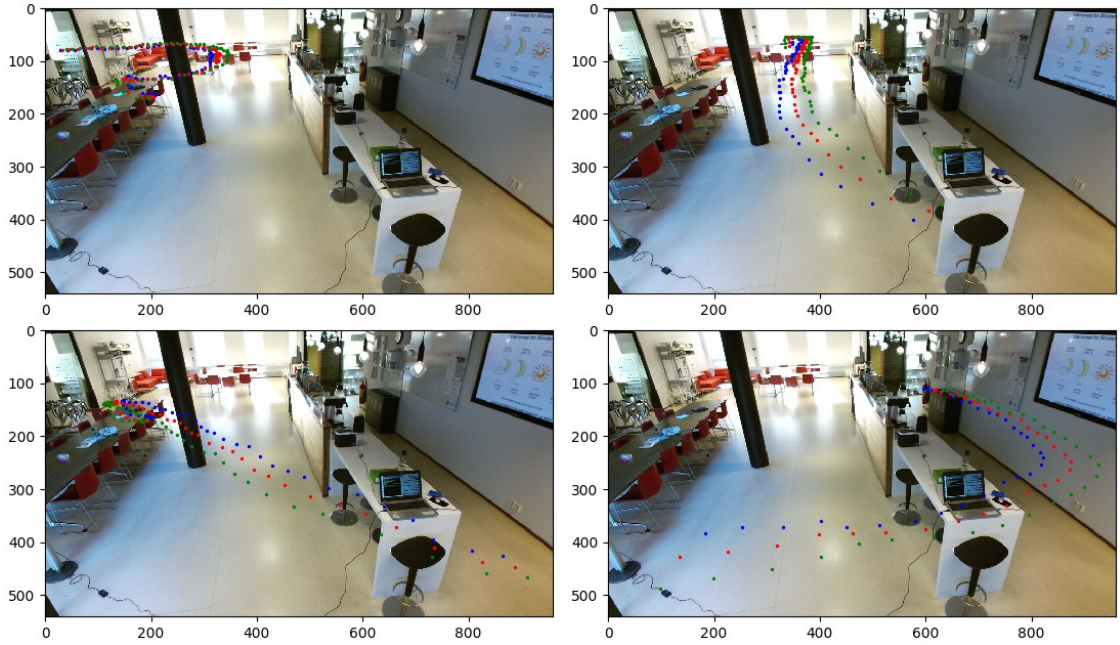
We can see examples of a few of the activity paths (AP) performed by the subjects of the experiments in Figure 5.4. The activity paths were acquired by capturing specific body keypoint throughout the frames. In this figure, the body keypoints being plotted are 2, 1, and 5, that represent respectively right shoulder, neck and left shoulder (as seen in Figure 3.3). The right shoulder is plotted in blue, the neck in red, and the left shoulder in green.

On the top left of Figure 5.4, we see an activity path (AP) that goes from the point of interest (POI) side table (st) to the top left (tl) POI where the person leaves the room. On the top right of the figure, the AP shows the path from the bottom right (br) region to the back table (bt). On the bottom left, an AP from side table (st) to the bottom right (br) is shown. On the bottom right we see the AP from the POIs bottom left (bl) to the coffee machine (cm). One thing we can assume once we identify this AP is that the person will soon use the coffee machine (cm) for instance.

For the experiments of this approach, a total of 91 recordings were done. From those recording, more than 200 trajectories were extracted. During the preprocessing stage, each trajectory was labeled with the final goal. This was done keeping in mind the possibility of designing an experiment and model that uses Hot-one encoded strategy. Short experiments were performed but did not yield good results so the strategy was discarded.

The model used to predict the trajectory of the data recorded uses an LSTM layer of size 64 followed by a dense layer of size 2. The LSTM is a Many-to-One design, so the predictions are acquired interactively, i.e. each prediction outputs only one $x$ and one $y$; to predict the second $x$ and $y$ the previous predicted $x$ and $y$ will be used as input. Executions were made with different sizes of the sequence inputted ($I$) and of the prediction ($P$). The prediction sizes were $5, 8, 12, 15$ and the input sizes were $8, 12, 15, 20, 30$. Each execution was measured with the metrics ADE, FDE and mean square error for $x$ and for $y$. This was done in order to verify how far into the future the model can predict a person walking and how far back it has to look. By measuring MSE for $x$ and $y$ separately it is possible

Figure 5.4: Activity Paths (APs) from subjects collected in the experiments. In the figure, blue represents right shoulder, red represents the neck and green represents the left shoulder of the person observed. On the top left of the figure an AP from POIs side table (st) to top left (tl) is shown. On the top right of the figure we have an AP from bottom right(br) to back table (bt). On bottom left of the figure an AP from side table to bottom right, and on the bottom right of the figure an AP from bottom left (bl) to the coffee machine (cm).




to see how good the predictions are on each axis. It is important to note that the model of each execution had individual training with the same dataset but with different sizes of training sequences.

## 5.3 Results

The results from the points of interest approach are presented in this section. In Table 5.1 it is possible to see the metrics for each execution. It is important to note that the model of each execution had individual training with the same dataset but with different sizes of training sequences. The best results came from experiment number 1 with an input size of 8 timesteps and prediction size of 5. It got an average displacement error of 281.59, a final displacement error of 659.08, and a mean square error for $x$ of 238.28 and for $y$ of 43.31.

Papers that use these metrics use 8 frames to predict the next 12 frames (XU et al., 2018) (ALAHI et al., 2016). We will present the results for the experiment that has the same setting (experiment number 3) with higher details as well as the issues encountered.

Table 5.1: Metrics ADE, FDE and mean square error for $x$ and $y$ presented for experiments with different sequence size and prediction size.

| Experiment nr. | Input size - $I$ | Predicitions - $P$ | ADE | FDE | MSEX | MSEY |
|---|---|---|---|---|---|---|
| 1 | 8 | 5 | **281.59** | **659.08** | **238.28** | **43.31** |
| 2 | 8 | 8 | 567.32 | 1578.76 | 492.87 | 74.45 |
| 3 | 8 | 12 | 1273.08 | 3791.65 | 1131.55 | 141.53 |
| 4 | 8 | 15 | 1844.69 | 5706.93 | 1586.99 | 257.69 |
| 5 | 12 | 5 | 1536.32 | 4711.97 | 1321.26 | 215.06 |
| 6 | 12 | 8 | 1382.0 | 4220.5 | 1190.04 | 191.96 |
| 7 | 12 | 12 | 1489.2 | 4599.83 | 1294.17 | 195.03 |
| 8 | 12 | 15 | 1605.58 | 5047.79 | 1415.81 | 189.77 |
| 9 | 15 | 5 | 1472.23 | 4599.51 | 1298.37 | 173.86 |
| 10 | 15 | 8 | 1387.86 | 4329.89 | 1222.39 | 165.47 |
| 11 | 15 | 12 | 1426.57 | 4462.32 | 1257.31 | 169.26 |
| 12 | 15 | 15 | 1546.11 | 4959.69 | 1371.53 | 174.58 |
| 13 | 20 | 5 | 1451.92 | 4638.84 | 1285.97 | 165.95 |
| 14 | 20 | 8 | 1401.93 | 4449.19 | 1240.32 | 161.61 |
| 15 | 20 | 12 | 1433.87 | 4572.84 | 1268.67 | 165.19 |
| 16 | 20 | 15 | 1803.65 | 6001.77 | 1636.38 | 167.27 |
| 17 | 30 | 5 | 1718.43 | 5699.11 | 1557.64 | 160.78 |
| 18 | 30 | 8 | 1655.32 | 5474.21 | 1499.19 | 156.13 |
| 19 | 30 | 12 | 1629.82 | 5410.77 | 1472.41 | 157.41 |
| 20 | 30 | 15 | 1751.98 | 5983.1 | 1584.85 | 167.14 |

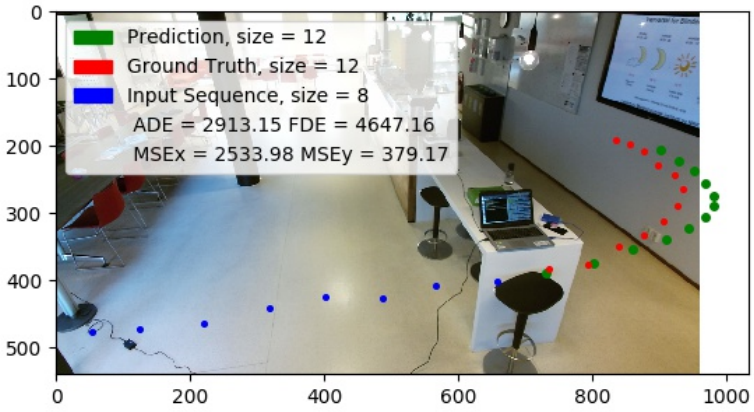
Experiment number 3 had the following results: ADE of 1273.08, FDE of 3791.65, a MSE on $x$ of 1131.55 and on $y$ of 141.53. We can see by the comparison of the MSE of $x$ and $y$ that the $y$ axis gets better predictions since it has much lower values and the same is true for other experiments. This can be due to the fact that more movement happens in the $x$ axis. Also, the resolution of the $x$ dimension is higher, since each frame have a width of 960 pixels($x$) and height of 540 pixels($y$).

This color scheme is the same for all the figures from this chapter: it is possible to see the input sequence in blue, the predictions in green and the ground truth in red. In each plot, we also have the value of the metrics. Another thing to notice is the $y$ axis, it is inverted because that is how openpose outputs the results, it inverts the $y$ axis before supplying the bodyjoints coordinates. Therefore the reference point (0,0) of $x$ and $y$ is on the top left of the graph in the figure.

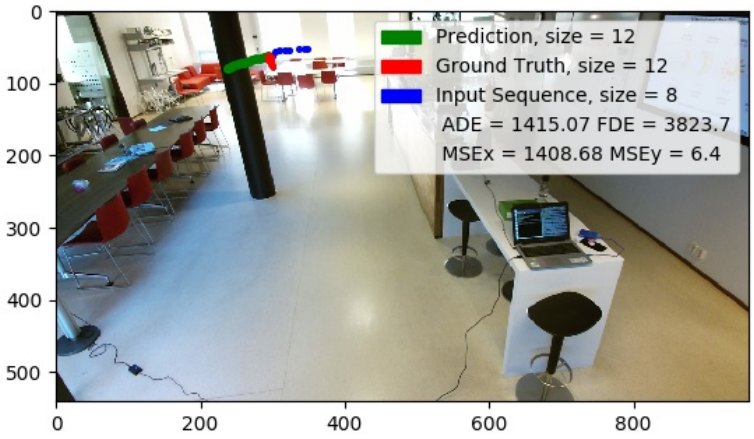
On top of Figure 5.5, we can see that the prediction 204 is visually quite similar to the ground truth but the metrics outputted have high values: ADE of 2913 and FDE of 4647 indicate a poor prediction. This only happens because the person walking is very close to the camera and any movement results in high displacement variation. If we

compare it with the prediction 30 in the bottom of Figure 5.5, which is of a trajectory in the back of the environment far away from the camera, it is clear that the prediction is not good compared to the ground truth, and still the metrics of this prediction are better than the metrics of prediction number 204, having ADE of 1415 and FDE of 3823. It shows that these metrics have limitations for the experiments conducted due to the difference of apparent speed when people are walking close to the camera and far from the camera.

Figure 5.5: This Figure shows the result of prediction number 204 on top and prediction number 30 on the bottom.




(a) Prediction number 204

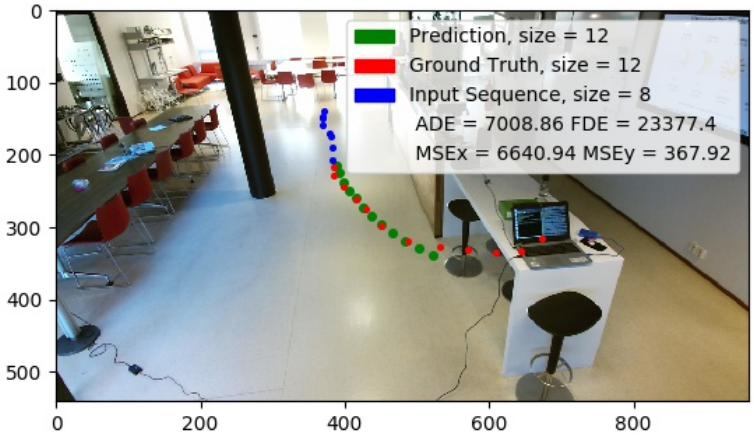



(b) Prediction number 30

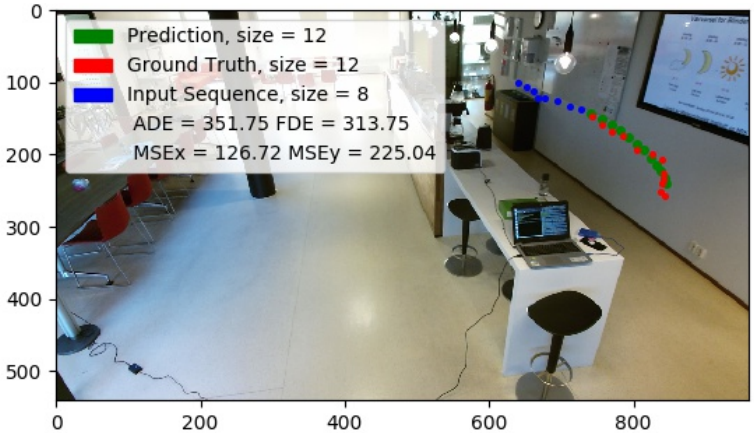
The results in Figure 5.6 from predictions number 10, 12 and 15 show good results in terms of direction: as we can see that the green and the red dots go in the same direction. Even though we have the same direction, the predicted and ground truth points have a mismatch. On top of Figure 5.6, we have prediction number 10 we can see that the model correctly predicts the direction but not the speed of the person so the final prediction is quite far from the expected, outputting a high value for the final displacement metric FDE of 23377.40. In this case, we verify one limitation when the path is correctly predicted

but the speed is not. That is one of the reasons the metrics are not a good quantitative estimation of prediction quality.

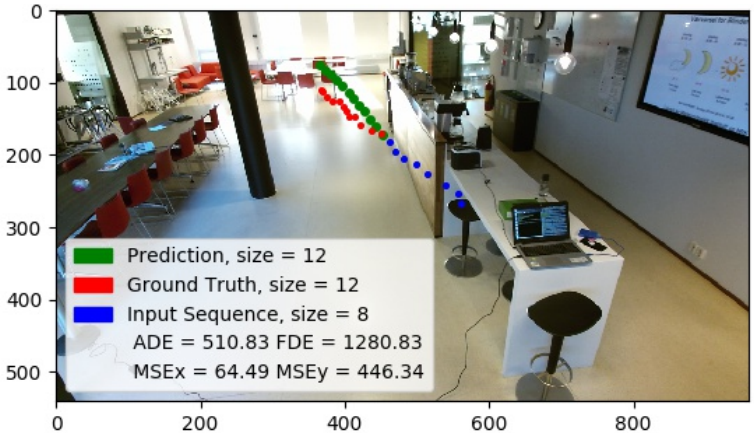
Figure 5.6: This Figure shows the result of prediction number 10 on top, prediction number 12 in the middle, and prediction number 15 on the bottom.




(a) Prediction number 10




(b) Prediction number 12




(c) Prediction number 15

Since each experiment takes the average of all predictions to present the metrics, it is very easily disturbed by cases such as prediction number 48 represented on *b)* of Figure

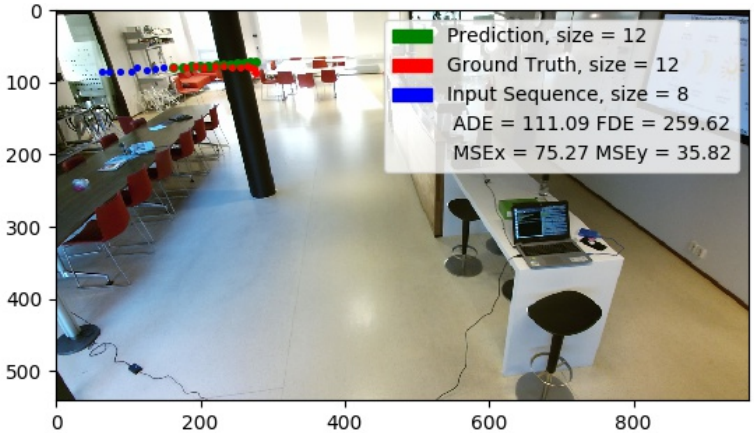
5.7, where a bad prediction is done and it is close to the camera. The model predicted that the person would take a turn and walk almost in the opposite direction, when in fact if made a right turn. Being close to the camera, the metric values will be high. When the same situation happens in the back part of the scene, further away from the camera, the metrics assume much lower values, even though the prediction is as bad as predictions performed closer to the camera. The metrics on this prediction were ADE of 77992 and FDE of 225378. These values are expected to be lower when further away from the camera.

In the bottom of Figure 5.7 (prediction number 66) we can see that the prediction does not go in the direction of the ground truth. This happens because we have similar training sequences going to the direction of the ground truth and to the direction of the prediction and the model was not able to learn enough to tell them apart. This hints to the fact that not enough data was used and with extra data it would be possible for the model to learn both trajectories without any need of clustering or similar techniques.

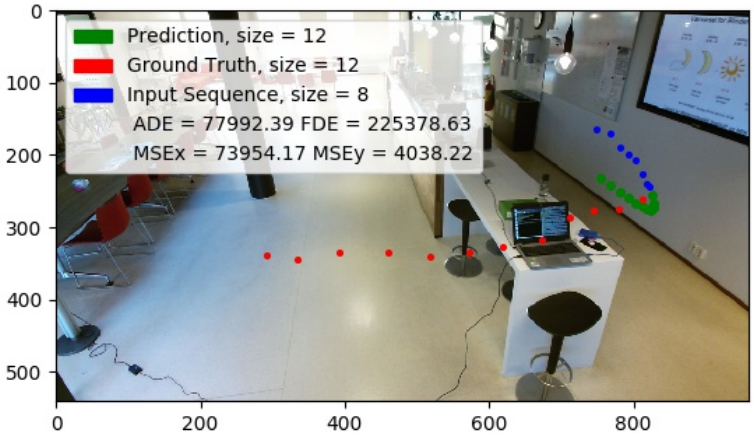
In Figure 5.8 (prediction number 24) it is possible to see that the ground truth is simply a person standing still, which means that the person was performing an action. In this case, the person was in front of the coffee machine. The model was not able to learn in all cases when a person is performing an action standing still, and this makes the metrics also have high values. This is a problem also faced in the related works (XU et al., 2018) (ALAHI et al., 2016).

When looking at the results from (XU et al., 2018) in Figure 2.7 it is possible to see that the people recorded are very distant and cover a small area in the time of the prediction. It is very different from our case since the people walking in our experiment are at times very close to the camera and sometimes away from it. From the camera point of view, the subjects that are close have a much greater displacement in comparison to the subjects in the back even though in the real world they walked the same distance. This has a significant impact on the metric calculation. The metrics outputted for a trajectory that occurs close to the camera, in the bottom of the frame, have a much higher value than the metrics for a trajectory that happens away from the camera. This problem can also be seen comparing the predictions number 204 in the top of Figure 5.5 and prediction 40 shown on top of Figure 5.7 - one is far from the camera and the other is close. Looking at them we can see that both represent good prediction that are very close to the ground truth. When checking the metrics we see that for prediction number 40 we have ADE of 111.09 and FDE of 259.62. Prediction 204, even though presents good prediction visually, has

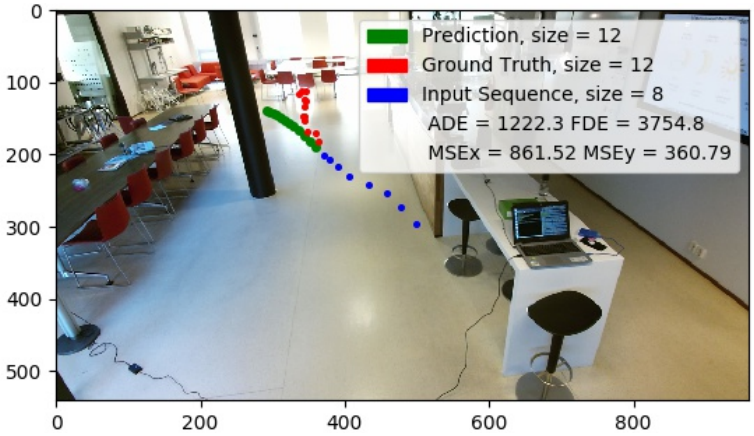
Figure 5.7: This Figure shows the result of prediction number 40 on top, prediction number 48 in the middle, and prediction number 66 on the bottom.




(a) Prediction number 40




(b) Prediction number 48

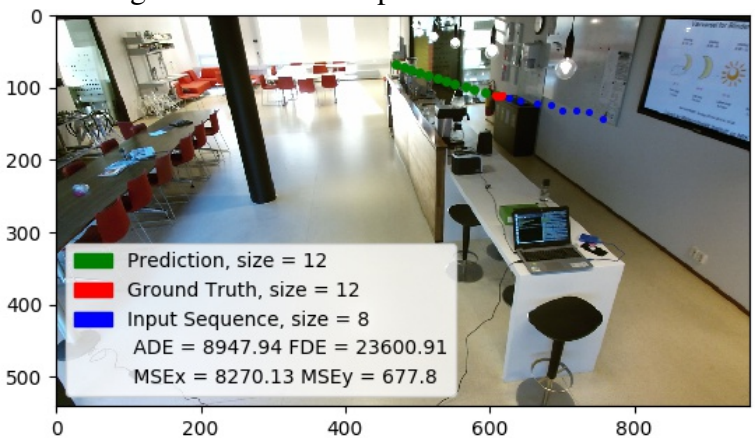



(c) Prediction number 66

an ADE of 2913.15 and an FDE of 4647.16. Looking at these two pictures it is clear that the metrics used have serious limitations when evaluating the prediction in our case.

In order to get around this problem, a few changes could be made. One of which could be to use data that does not have such big variation regarding the distance to the

Figure 5.8: Result of prediction number 24.




camera. Another option would be to have the 3D information of the position of the person. Instead of having only $x$ and $y$, also having the depth information $z$ would help with the metric problem. With a few adaptions to the metrics formulas, it would be possible to calculate the real distance between points in the 3D space, instead of the distance of 3D points projected into the 2D space captured by the camera. In the work of (CHEN; RAMANAN, 2017) a 3D pose estimator is made available. It uses a NN that receives as input a 2D pose and estimates a 3D pose.

Those two points along with the results from paper (CHEN; RAMANAN, 2017) were the motivation for the experiments with 3D-LSTM explained in Chapter 6.

# 6 COMPARATIVE PATH PREDICTION OF 2D AND 3D-ESTIMATED APPROACH

In this chapter, we present an approach motivated by the results from the points of interest approach seen in Section 5.3 and the work of (CHEN; RAMANAN, 2017). We designed experiments to compare two prediction models: one that uses 2D information and another that uses 3D estimated information. The 3D estimated data was acquired with a 3D pose estimator that is presented. Relevant information and tests regarding the 3D pose estimator used and its features and limitations are also discussed in this chapter.

## 6.1 3D Pose Estimator

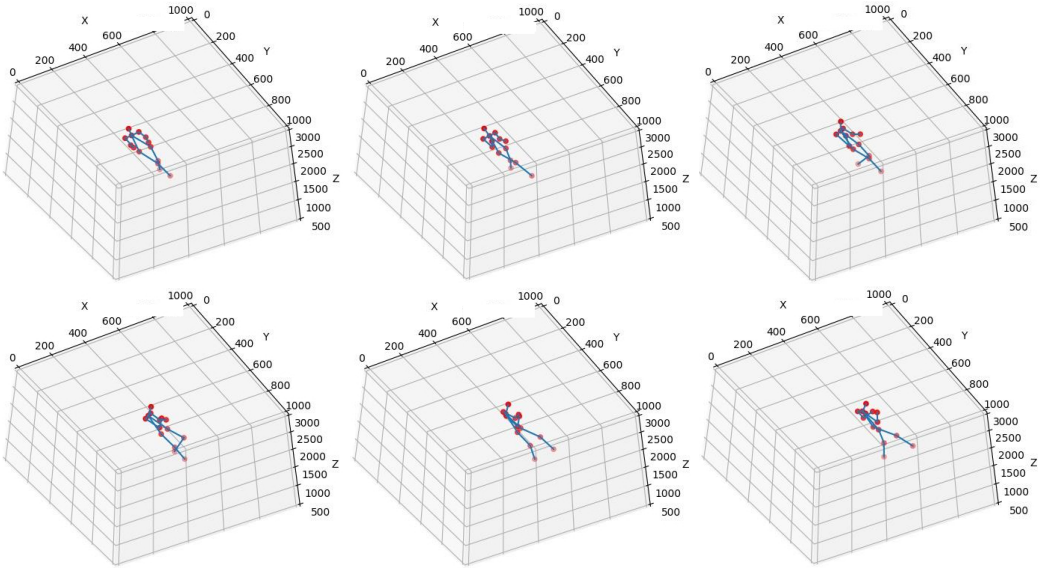
Our previous experiments showed us a few relevant points regarding our approach. As can be seen at the end of section 5.2 we believed that by having the 3D information of a person walking, some issues would be resolved. The 3D pose estimator provided by (CHEN; RAMANAN, 2017) is presented by the authors as the state of the art in 3D pose estimation based on 2D poses, therefore, it was chosen to be used in our project.

The 3D Pose Estimator was developed in Matlab (MATLAB, 2010) and was open code. The format used by the neural network was not the same employed in our experiments. Therefore, some modifications had to be done in the data and the Matlab code to run our experiments. After reshaping the data it was possible to use the 3D Pose Estimation on the dataset collected in the project and on the data from *Human3.6m* dataset. The data collected in the project presented some irregularities as shown in Chapter 7 (partially due to occlusion and partially due to low frame rate and other issues). For that reason, the 3D Estimation presented better results on *Human3.6m* dataset, so it was used for the experiments with this approach.

We tested the 3D estimator on the pose acquired by running Openpose on the *Human3.6m* data and in Figure 6.1 we can see the 3D pose estimation done when a 2D pose generated by Openpose is provided. When observed in sequence, it is possible to see an accurate representation of the human body walking in the 3D space (better visualized in a gif image or a video). We see that the NN from (CHEN; RAMANAN, 2017) is capable of providing a good 3D representation of the person walking.

From this point on, since the 3D pose estimation appeared to be possible, our approach changed and we used this estimated information in the following stages of our work. From the results of Figure 6.1, we assumed it was possible to keep track of one

Figure 6.1: 3D pose estimator used on data from *Human3.6m* dataset. The estimator received as input a 2D pose and was able to recreate this 3D pose one frame at a time. The figures are of a few spaced ordered frames of a person walking.



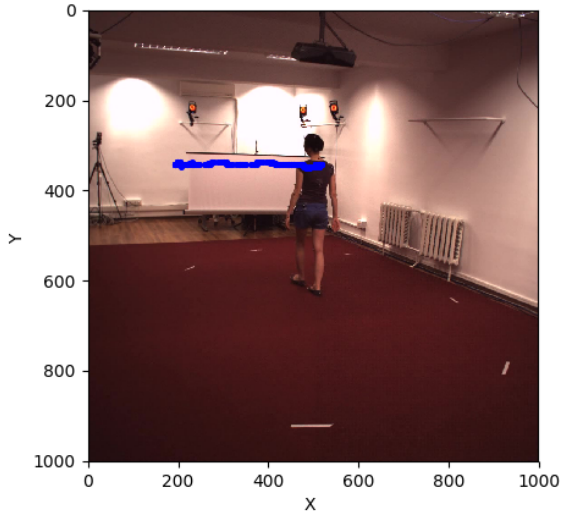
keypoint throughout the sequential frames and with that create a 3D path representation of a person walking based only on the RGB image of a person. Therefore several experiments were executed using the data from the estimator and the results can be seen in Section 6.3.

The predictions from the 3D-LSTM were not as good as expected and the output hinted that the problem was the data. After looking at the 3D sequence created by the 3D estimator we were able to see that even though the 3D creation from the 2D pose is a good representation of the pose as seen in Figure 6.1, the depth parameter does not remain constant between the sequential frames.

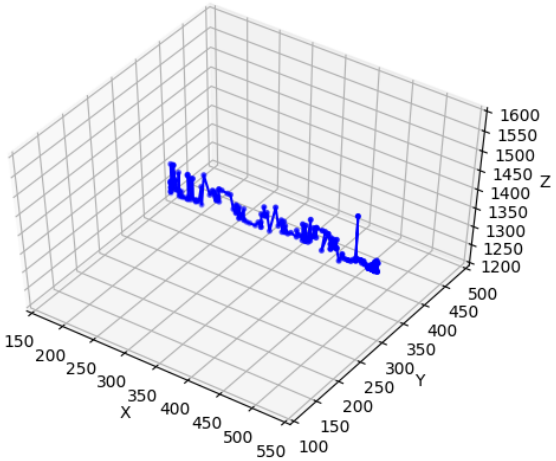
In Figure 6.2 we demonstrate the result of the 3D path reconstructed using 3D Pose Estimator. On $a)$ of the figure the 2D path is plotted with the environment as background. The same data is plotted on $b)$ but with the estimated depth after the reconstruction of the path in the 3D space. The path is seen in blue. It is clear that the path has significant discontinuities. Since the plot on $a)$ is performed with $x$ and $y$ and the path is smooth, the discontinuities happen in the $z$ data. On $c)$ of the figure a projection of axis $x$ vs $z$ is plotted. Here we see that the abrupt variation happens on $z$ axis. The same happens in $d)$ where $z$ is plotted with $y$. This makes it clear that the depth estimated from the pose is not consistent between frames.

After this analysis, we know that the estimator can create good plausible 3D poses individually, but is not able to keep a consistent depth between the frames. Even though

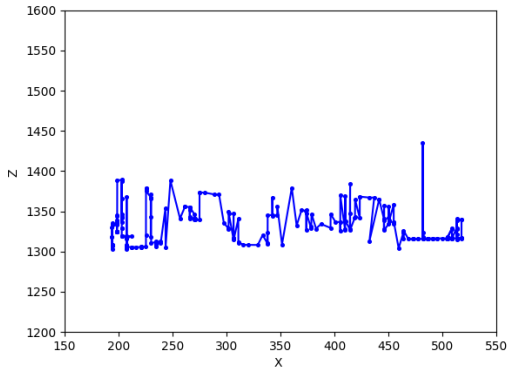
Figure 6.2: This figure shows the discontinuity in the depth estimated data. In $a)$ the $x$ and $y$ data are plotted in the environment representing the path. In $b)$ on the figure, $x$, $y$, and estimated $z$ are plotted, so we can see the path reconstruction after 3D pose estimator is used. In $c)$ a projection of the 3D path is plotted showing only $x$ and $z$ to make it clear that the discontinuities come from $z$ data. In $d)$ the same is done but using $y$ vs $z$.



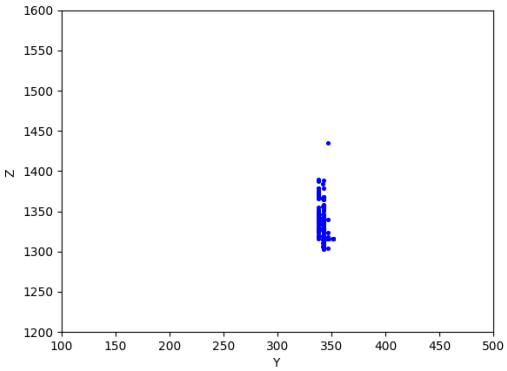

(a)




(b)




(c)




(d)

it creates issues to our work, this is a very relevant finding not only to our project but to any researcher that intends to use the estimator. The limitations discovered in this chapter must be taken into account before using the estimator from (CHEN; RAMANAN, 2017).

## 6.2 Comparative 2D-LSTM 3D-LSTM Approach

After performing experiments with the data collected, a few issues were encountered as shown in Chapter 7. In an attempt to isolate the problems related to data quality, data from the *Human3.6m* dataset was used. The dataset from the work of (IONESCU et al., 2014a) is presented in Section 3.5. By using this dataset we can be sure that the data has good quality and we are able to focus on experimenting with the model.

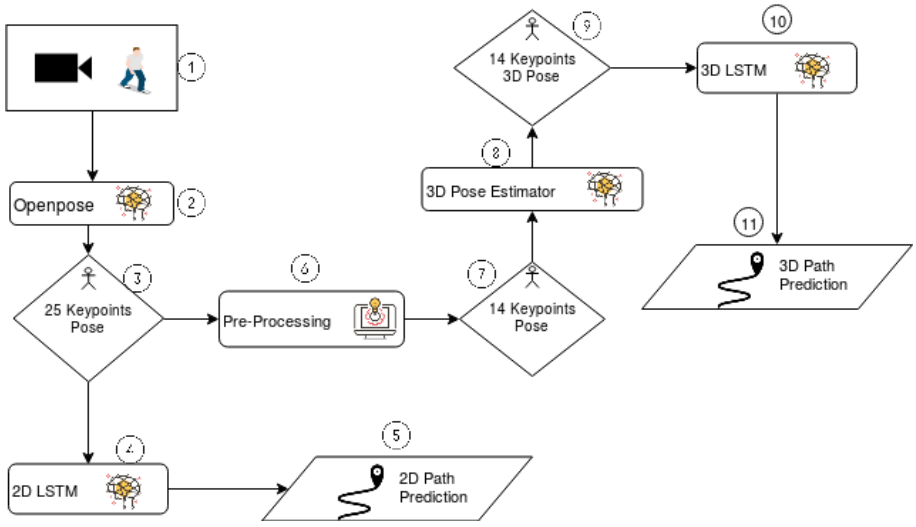
The predictions from previous experiments had a significant limitation. Since they were done using RGB images, the inputted data was a 2D projection of the 3D world. The depth information was not present, so the 2D path might not represent well the path that the person is walking on. Depending on the angle of the camera, the creation of the representation of the 2D path of a person can have big variations in different parts of the area observed. Having depth information can increase the quality of the path representations. To deal with those issues, we follow two different approaches.

In order to assess the different camera angles issue, we used the *Human3.6m* dataset. This dataset has 10 subjects walking in a controlled environment being recorded from 4 well-defined angles. Therefore, the same path has four perspectives. This allowed us to perform experiments with distinct setups using data from different cameras.

The whole approach of the experiments of this chapter can be seen in the concept map shown in Figure 6.3. The approach consists of providing a video of a person walking as input and receiving as output the prediction of the walking path using 2D data and the prediction of the walking path using 3D estimated data. The main goal of this experiment is to find out if estimating the 3D data will improve prediction results or not.

Step 1 in Figure 6.3 represents the data being captured. An RGB camera records a person walking. Step 2 represents the Openpose CNN being applied for every frame captured. Openpose receives an RGB image and outputs a 25 body keypoint pose that contains $x$,$y$ and $c$ information for every keypoint ( $x$ - X position, $y$ - Y position, $c$ - certainty of the reading). The output of Openpose is represented in step 3. After having the Openpose data it is possible to go to either step 4 or 6. Step 4 represents the 2D-LSTM model that uses $x$ $y$ path information to train and predict future paths. The result

Figure 6.3: Concept Map of the approach for evaluating predictions from 2D-LSTM in comparison to 3D-LSTM.




is represented by step 5 in Figure 6.3. This is the flow that has as results a 2D prediction from the 2D-LSTM.

To get to the 3D-LSTM, from step 3 the next step is 6. The output from Openpose has 25 keypoints but the estimator used 14 keypoints in order to predict the 3D pose. For that reason, it is necessary to transform the 25 keypoints into a body representation of 14 keypoints compatible with the representation expected from the estimator. This adaptation is performed in step 6 and has, as a result, a 14 body keypoint pose represented by step 7. Once the data is in the correct format it is possible to use NN that estimates the 3D pose, seen in step 8. It uses the 2D poses received and tries to match with the 2D part of a 3D pose library. The most similar match found is outputted from the Neural Network. Therefore, a 3D pose with 14 body keypoint is the result of the estimator, and that is represented by step 9.

After obtaining the 3D pose a 3D path is created and used as input in the 3D-LSTM in step 10. Step 11 represents the prediction that the LSTM provides. The result of this process is two different RNN-LSTM trained models. One deals with 2D data and the other deals with 3D data. Also, as a result, we have 2D and 3D prediction. With the predictions, it is possible to assess the quality of both models. Comparing both predictions it is possible to verify if the added 3D estimated depth improves results or not.

The output of the 3D-LSTM is a 3D path with $x$, $y$, and $z$, and our goal is to asses if and how much the 3D estimated information can improve the 2D prediction. For that reason, for both predictions, we use 2D evaluation metrics. For the 2D output, we can simply use the metrics and for the 3D output, we select only the $x$ and $y$ parameters, and

asses the 2D prediction in order to be able to make a comparison between the 2D and 3D-LSTM.

## 6.3 Results

Since the main goal of this work is to asses the 2D prediction from both models we evaluate the $x$ and $y$ values predicted in each LSTM designed by us. To do so we ignore the $z$ output of the 3D-LSTM and evaluate only $x$ and $y$ and see if the presence of the depth helps increase the accuracy of the predictions. In order to do that, we use the metrics described in Section 3.7. We have available the data of a few subjects walking in a recorded environment from dataset *Human3.6m*. We used subjects S1, S2, S3, S4 and S5 for training and S6 for testing. This assures that the model trained never had any contact with the testing data. This was performed with data of the camera recorded from 3 camera angles, and each subject walked for around 1 minute in 2 different routes. We are predicting different timesteps sequences (50, 75, 100, 150), using a 6000 training set size, 200 testing set size, and training for 35 epochs a 6000 batch size.

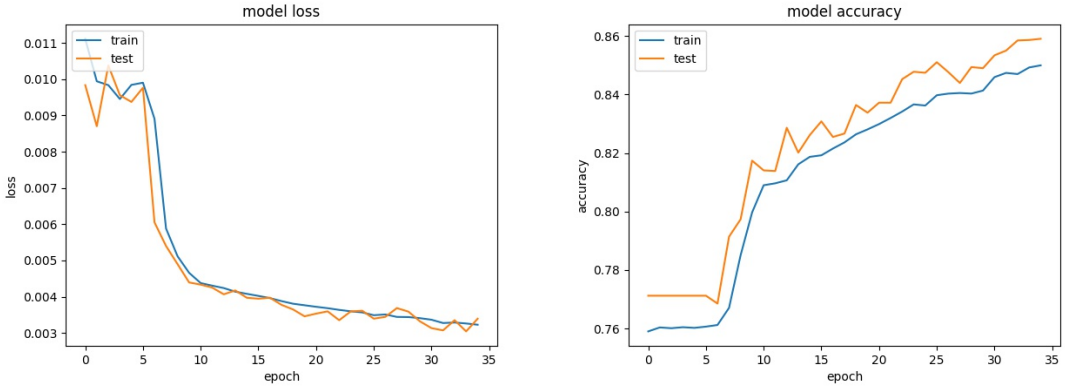
The epoch is a hyperparameter that indicates how many times the learning algorithm goes through all training dataset. So in each epoch, each sample from the dataset is be able to update and improve the parameter of the model. The epoch can have one or more batches, in our case we used 35 epoch and 6000 as the batch size.

We reached the number of 35 epoch after training the model many times with higher epoch number and realizing that the improvement in accuracy was very small after epoch 35, and the reduction in the loss function was very small after epoch 35. Running many epochs would improve the result but using a large epoch number demands a lot of computational power and we have limitations regarding the hardware and equipment available. The weights of each of the models have been saved and the training can continue and be performed for a higher epoch number in the future stages.

In Figure 6.4, we can see the loss and accuracy progression through the epochs. These results are from model 3D-LSTM with sequence size 150 and camera angle A, execution number 8 from Table 6.1. At each epoch, the weights of the models are being updated and we can see the effect of that on the loss that reduced until it reaches 0.003 for both training and testing data by the end of epoch 35. We also see a rise in the accuracy that reaches values of 0.85 on the training data and 0.86 in the testing data.

The 2D-LSTM presented good predictions for some cases and bad predictions for

Figure 6.4: This image shows the loss and the accuracy at the end of each epoch during the training process of 35 epochs. The model trained was 3D-LSTM with sequence size 150 and camera angle A.

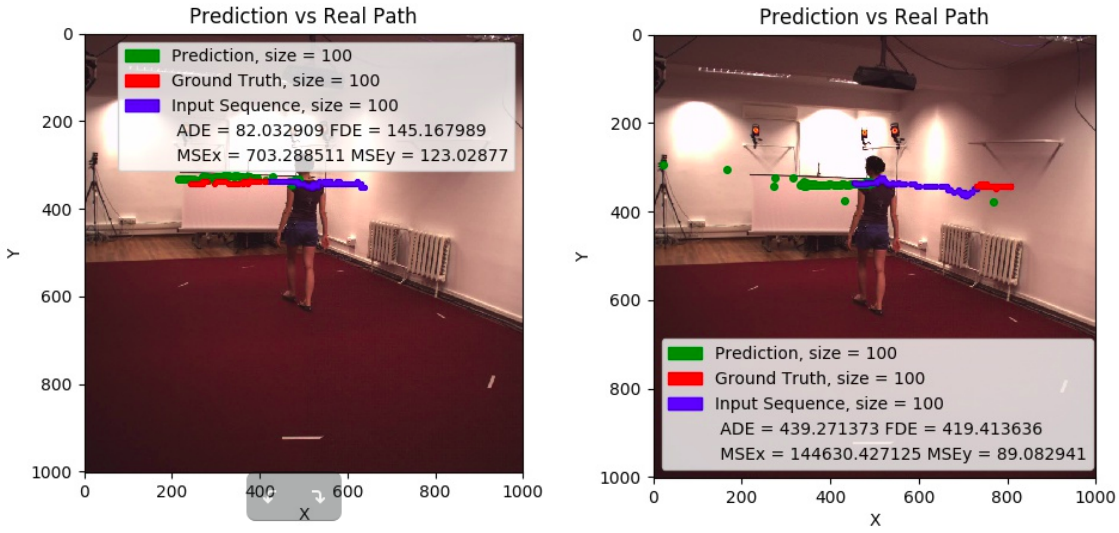



others. In Figure 6.5 we can see a good prediction case on the left in which a sequence of size 100 was used to train and predict. The background of the figure is a single frame of the environment shown in order to present the experiment scenario but is not related to the data being displayed. The distance between the predicted output and the ground truth is small. Using data from a few subjects, the 2D model was able to learn and predict the path of another independent subject walking. For that particular prediction the metrics presented lower error, the ADE was 82.03 and the FDE was 145.17. The problem of this model is that it still have cases in which it does not output a good prediction. It can predict well in certain cases but is not able to generalize for all paths. The average metrics for 200 predictions were ADE of 242.40 and FDE of 237.30. The metrics were affected by the cases in which the model is not able to learn from.

The 3D-LSTM predictions were not good. The model was not able to learn properly in any case. Looking at Figure 6.5 on the right we can see that the prediction (in green) is very far from the ground truth (in red). Apart from that, we cannot see any clear pattern being learned from the 3D-LSTM opposite to the 2D-LSTM. The outputted data seems scattered and does not present the behavior expected. The average ADE and FDE were high. The ADE metric had an ADE value of 439.27 while FDE reaches 419.41. If we compare both LSTMs we can see that adding the estimated 3D values did not improve the results - in fact, it did the opposite.

The results from the 3D-LSTM were worse than expected: the FDE and ADE had much higher values than the predictions from 2D-LSTM. The data that was input into 2D-LSTM was detected data and the data inputted into 3D-LSTM was estimated, so the quality of the estimation is a probable reason for the bad predictions. After exploring the

Figure 6.5: On the left, we have a case of a good prediction from 2D-LSTM. Looking at the predicted path (green) we can see that it is very similar to the ground Truth (in red). The input sequence is presented in green. We can see that for this case the 2D-LSTM is able to learn and outputs a good prediction. On the right, we show the common output from the 3D-LSTM that is not able to predict future paths. The predicted path (in green) presents a senseless output with isolated predicted dots.




sequential data outputted from the 3D Pose Estimator from (CHEN; RAMANAN, 2017), we were able to verify that even though the 3D Pose was a very realistic and plausible pose for the scenario, the depth between sequential frames was not consistent. This is the reason the 3D-LSTM is not able to learn, because the inputted data was not representing a clear and smooth 3D path, but rather a 3D path with discontinuities in the Z axis as shown in Figure 6.2.

This inconsistency in the 3D estimated data also affects the metrics and what was expected to be corrected with the depth information. As it was stated at the end of section 5.3, we assumed that the 3D estimated data would provide a good description of the 3D path. With that, it would be possible to calculate the metrics on the 3D space instead of in the 2D projection of the 3D space. That could not be done with a wrong 3D estimated data since we would have a wrong ground truth from which the error would be calculated.

In Table 6.1 we can see the value of the metrics for the experiments. Each experiment has its own model designed for different sequence sizes, camera angle and whether it is 2D or 3D-LSTM. The values shown in the table are from the average of 200 predictions in comparison with the ground truth. We have experiments from 3 camera angles and they are named A, B, and C and are placed on the edges of the recording environment so each path is recorded from different angles. Each line from the table represents a distinct model that was trained for 35 epochs. The weights for each model are stored and the

training can be continued for more timesteps in the future or used to perform predictions and calculate metrics.

Table 6.1: Table of metrics for execution with 2D-LSTM and 3D-LSTM on *Human3.6m* data. The metrics are ADE, FDE and mean square error of $x$ and $y$ for experiments with different sequence and prediction size.

| Nr. | Size | Model | Camera | ADE | FDE | MSEX | MSEY |
|---|---|---|---|---|---|---|---|
| 1 | 50 | 2D | A | 305.84 | 308.49 | 4418.60 | 313.44 |
| 2 | 50 | 3D | A | 330.00 | 321.68 | 7174.17 | 645.88 |
| 3 | 75 | 2D | A | 261.23 | 260.45 | 10038.43 | 382.93 |
| 4 | 75 | 3D | A | 313.60 | 342.80 | 15760.22 | 471.50 |
| 5 | 100 | 2D | A | 242.40 | 237.30 | 15492.75 | 439.47 |
| 6 | 100 | 3D | A | 285.30 | 269.91 | 16163.69 | 403.75 |
| 7 | 150 | 2D | A | 258.87 | 274.30 | 12646.50 | 617.01 |
| 8 | 150 | 3D | A | 249.89 | 260.60 | 22743.95 | 438.58 |
| 9 | 50 | 2D | B | 445.06 | 448.6 | 2585.90 | 236.11 |
| 10 | 50 | 3D | B | 464.79 | 433.63 | 28650.04 | 2487.12 |
| 11 | 75 | 2D | B | 421.45 | 425.32 | 4693.44 | 207.72 |
| 12 | 75 | 3D | B | 450.02 | 458.57 | 13708.16 | 1789.74 |
| 13 | 100 | 2D | B | 432.35 | 433.13 | 7884.99 | 209.32 |
| 14 | 100 | 3D | B | 482.20 | 506.57 | 21117.74 | 1678.85 |
| 15 | 150 | 2D | B | 429.28 | 418.34 | 13706.13 | 244.84 |
| 16 | 150 | 3D | B | 504.41 | 492.50 | 44838.88 | 5455.33 |
| 17 | 50 | 2D | C | 382.53 | 382.42 | 4638.87 | 152.66 |
| 18 | 50 | 3D | C | 564.28 | 549.24 | 42745.69 | 627.13 |
| 19 | 75 | 2D | C | 381.54 | 382.69 | 6937.58 | 156.37 |
| 20 | 75 | 3D | C | 776.24 | 779.14 | 351925.99 | 201185.91 |
| 21 | 100 | 2D | C | 367.27 | 366.53 | 11679.00 | 234.70 |
| 22 | 100 | 3D | C | 591.21 | 569.52 | 79054.22 | 543.80 |
| 23 | 150 | 2D | C | 368.66 | 361.35 | 20822.83 | 335.83 |
| 24 | 150 | 3D | C | 446.94 | 441.27 | 38697.57 | 2332.06 |

In Table 6.1 we can see that the error for the 2D model is in most cases lower than that error of the 3D model. This shows us that the addition of the 3D estimated data did not improve the predictions. In fact, given that the 3D estimator created 3D paths with strong discontinuities as seen in Figure 6.1, the addition of the 3D information made the prediction worst and made it much harder for the model to learn from the data since the data was inconsistent. It is possible to see that generally the model learns better from camera angle A and the best average error metrics came from 2D-LSTM model with sequence size 100 ADE 242.40 and FDE 237.30.

By looking at the MSEx and MSEy columns in Table 6.1 we can also note that the error in the $x$ axis is much higher than the error in the $y$ axis. This is due to the fact that the general displacement is much higher in $x$ axis than in $y$ axis. The variation in $y$ axis

is minimal.

We note that the 2D model is able to learn the path for some cases but not for others. This hints us to the fact that the model did not learn all it was needed from the data used, i.e., it did not generalize from the data. We can assume that by having more data as input and raising the learning period the model will present predictions that are better and can generalize.

# 7 ISSUES

Some problems were encountered while performing the experiments. Here they are addressed and the approach to solving them is presented.

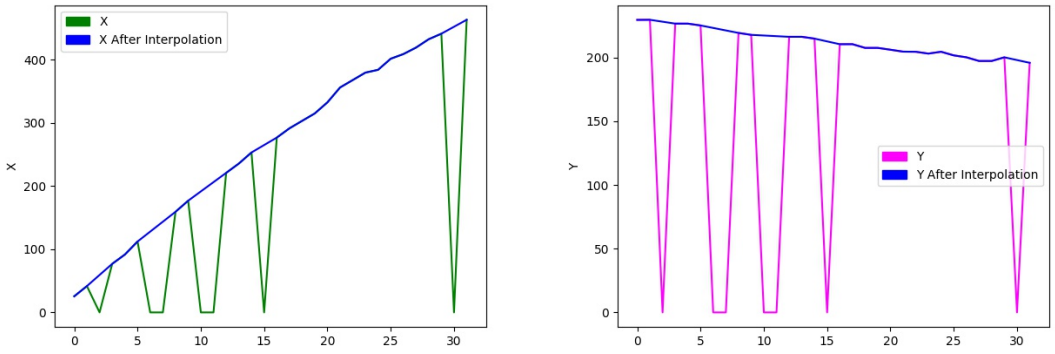
## 7.1 Discontinuity in the data

When using Openpose to extract body keypoints a certainty reading is provided for each keypoint. So keypoints that had an imprecise detection can be discarded. One of the problems encountered was that every time readings had low certainty the value is considered 0. This causes a discontinuity on the data and creates a bias towards zero in the prediction. Before this problem was addressed, predictions were all centered in zero. When there is a gap with 6 or more consecutive zeros there is no problem. We defined 6 as a threshold, so in that case, the sequence of zeros is considered a boundary and it is the end of a sequence. The model considers one sequence before and another sequence after the gap. When non-zero values start to appear again they become part of a new sequence. The problem is when there are intermittent groups of zero that are of size 6 or less. In the first and third column of Table 7.1, it is possible to see X and Y with some discontinuities in the sequence of values where zero is present. There are cases even worse than that where sequences of 5 zeros happen.

To fix this issue, linear interpolation was used to infer the value when the reading is 0. It is possible to see the result in column 2 and 4 of Table 7.1. Also in Figure 7.1, we can see the difference of the data before and after the interpolation. It is clear that running the model with the data before the interpolation has a negative impact on the prediction. As seen in Figure 7.1 this problem seems to be fixed.

After applying interpolation, a simple normalization was also performed. For the normalization, the values of $x$ were divided by the maximum value of $x$ which is the width of the image, and the values of $y$ were divided by the maximum value of $y$ which is the height of the image. After that, the quality of the predictions improved.

Figure 7.1: Example of problems with bad readings that cause intermittent zeros and the interpolation that fixes the issue. On the left of the figure, we have X values in green before interpolation: it is possible to see big discontinuities and this would affect final results. In blue it is possible to see the x values after the interpolation. On the right of the figure, we have the same situation but using values of Y.




## 7.2 Wrong recognition of people

During some experiments, unexpected recognition of people happened. In the recording environment a television was turned on and , as can be seen in Figure 7.2, people on the television were being recognized by openpose. This wrong recognition of people had a negative impact on the results. The solution was to blackout the pixels in the television in all frames collected during the experiment and run Openpose again. On the top right of Figure 7.2, it is possible to see the keypoints of people inside the television being recognized and on the bottom the blacked out television can be seen solving the issue.

## 7.3 Limited range of infrared camera

One of the issues encountered after the initial data collection was the range of detection of the infrared camera. The Kinetic used to collect the data has a depth range that goes from 0.8 meters to 4.0 meters and the path taken by the subjects exceeds this range. This limitation was a relevant point that shaped the following data collections and experiments.

In Figure 7.3, a subject is walking in the environment near to the camera while RGB and IR data are being captured. On the left, we see an image of the scene and on the right, the infrared detection of the same scene. The RGB images have 960x540 resolution

Table 7.1: Data with intermittent zeros

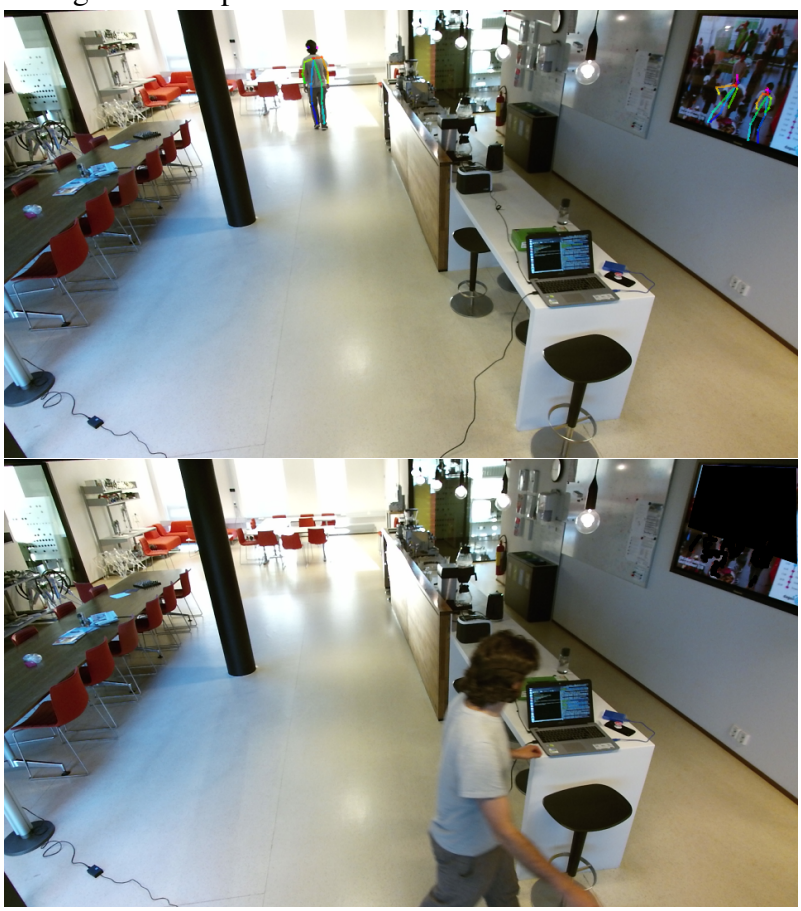
| X | X after interpolation | Y | Y after interpolation |
|---|---|---|---|
| 25.4669 | 25.4669 | 229.574 | 229.574 |
| 41.6258 | 41.6258 | 229.653 | 229.653 |
| 0 | 59.2328 | 0 | 228.149 |
| 76.8398 | 76.8398 | 226.646 | 226.646 |
| 91.5525 | 91.5525 | 226.633 | 226.633 |
| 112.15 | 112.15 | 225.211 | 225.211 |
| 0 | 127.827 | 0 | 223.255 |
| 0 | 143.504 | 0 | 221.299 |
| 159.182 | 159.182 | 219.343 | 219.343 |
| 176.743 | 176.743 | 217.816 | 217.816 |
| 0 | 191.421 | 0 | 217.308 |
| 0 | 206.099 | 0 | 216.800 |
| 220.777 | 220.777 | 216.292 | 216.292 |
| 235.472 | 235.472 | 216.296 | 216.296 |
| 253.048 | 253.048 | 214.881 | 214.881 |
| 0 | 264.830 | 0 | 212.687 |
| 276.613 | 276.613 | 210.493 | 210.493 |
| 291.259 | 291.259 | 210.513 | 210.513 |
| 303.068 | 303.068 | 207.593 | 207.593 |
| 314.823 | 314.823 | 207.596 | 207.596 |
| 332.372 | 332.372 | 206.163 | 206.163 |
| 355.917 | 355.917 | 204.651 | 204.651 |
| 367.661 | 367.661 | 204.555 | 204.555 |
| 379.425 | 379.425 | 203.117 | 203.117 |
| 383.862 | 383.862 | 204.567 | 204.567 |
| 401.408 | 401.408 | 201.722 | 201.722 |
| 408.766 | 408.766 | 200.218 | 200.218 |
| 419.049 | 419.049 | 197.316 | 197.316 |
| 432.255 | 432.255 | 197.32 | 197.320 |
| 441.018 | 441.018 | 200.205 | 200.205 |
| 0 | 452.021 | 0 | 198.041 |
| 463.025 | 463.025 | 195.878 | 195.878 |

while the depth images have 623x537 resolution. At that point in the space the person can be recognized well by the IR camera as can be seen.

Figure 7.4 shows the detection problem more clearly. On the left, we can see the subject walking on the back of the environment. However, when we look at the same scene being represented by the infrared data we only see the detectable area since the person is behind that area it is barely detected.

This was one of the reasons a 3D Pose estimator was used: since we were unable to detect the data, it was decided to estimate it with state of the art tools found on the

Figure 7.2: On the top right corner of the first image, it is possible to see two small skeletons being recognized inside the television. On the bottom, the solution to the problem is presented. Blacking out some pixels in the television area in all frames solved the issue.



literature such as (CHEN; RAMANAN, 2017). The data from the images was visualized and plotted in Rviz program which is a 3D visualization tool for ROS.

Figure 7.3: Image of a subject walking close to the camera and being detected by both Kinect cameras. On the left, an RGB image of a subject walking is presented. The same scene is captured by the IR camera and the scene with depth recognition can be seen on the right side of the image.

Figure 7.4: Image of a subject walking far away from the camera in the back of the room. Both cameras are capturing the scene. On the left, we see the RGB image and the subject walking in the back of the room. In the image on the right, we can identify the issue the IR camera is not able to detect the person since he is out of the depth range.

# 8 CONCLUSION AND FUTURE APPROACH

This research explored strategies for improving path prediction of humans walking using video. Based on qualitative results it is possible to conclude that the models created are able to learn from the data and output good results in some cases, but are not able to make reliable predictions in other situations. Extracting the depth component from a 2D video is quite challenging and the model used did not present a consistent depth estimation between frames. The quantitative results indicate that the metrics have limitations when used in our approach. As discussed in (Phillips; Wheeler; Kochenderfer, 2017) the difficulty of creating efficient learning models resigns in the ability of the model to generalize well from the training data. This is even harder to achieve in LSTMs that are very difficult to train and require larger amounts of data than simpler models. (GOODFELLOW; BENGIO; COURVILLE, 2016) say that neural network training problems are very complex and take a long time to solve, from days to months. With that being said, the results from the 2D-LSTM show that the model is able to learn and predict the path for some cases, but not for others. This suggests that the model would benefit and present better results if more data had been provided during the training stage. The model would then be able to learn and predict paths for more cases.

The problems faced when using the metrics ADE (Average Displacement Error) and FDE (Final Displacement Error) were mainly due to the fact that the data being used was a 2D projection of the 3D world. That was one of the motivations to estimate the 3D data leading to the 3D pose estimator and the 3D-LSTM. It was not possible to conclude if the metrics would work well with that approach since the 3D pose estimator did not work as expected. In future work datasets with 3D detected data will be used avoiding the estimation in order to be able to have a conclusion on the matter of the metric as well.

The comparative results from 3D-LSTM and 2D-LSTM point out that there is no improvement when using the 3D estimated data with the 3D-LSTM. This means that either the 3D-LSTM does not perform good learning and prediction of the 3D path or the data does not have a good depth estimation. After achieving these results the 3D path reconstruction data was investigated. It was observed that even though the 3D Pose estimator creates a very realistic 3D pose for a single frame, it does not keep the depth of the pose consistent between similar poses from sequential frames. In other words, the 3D Pose Estimator works well for single frames and production of a plausible 3D pose but does not present a consistent result when it comes to predicting the depth of

sequential frames. Therefore, the data inserted into the 3D-LSTM was not a precise 3D path as expected: it had many sudden changes in the depth information, making it impossible to asses whether the 3D-LSTM improves the path prediction in comparison with the 2D-LSTM. Based on these conclusions, practitioners should avoid using the 3D Pose estimator from (CHEN; RAMANAN, 2017) as a way to estimate the depths in sequential frames.

Despite the issue with the 3D pose estimator, the approach presented in this research to asses LSTM with 2D and 3D data is still pertinent but need some changes to achieve its goals. Future approaches should experiment with different 3D pose estimators. The ones that have better performance when it comes to depth estimations between sequential frames with similar poses should be used. Another interesting approach to assess only the difference between the 2D-LSTM and the 3D-LSTM is to use data from a dataset that detects both 2D and 3D pose. That way no estimation is involved in the process. After being sure that the 2D and 3D data are accurate we can have an unbiased assessment of both LSTMs.

It is perceived that the experiments can be further explored. The frame rate of the data from the dataset *Human3.6m* was 50 FPS. This is a very high frame rate and restricts how much into the future prediction can be made. With 50 FPS and predicting 150 points into the future, only 3 seconds are being predicted. As some works do, downsample of the frame rate and new evaluation of the predictions should be performed. Also, experiments varying the size of the sequence, as well as with more data and from different public datasets can be performed.

In order to perform the experiments, a framework with functions directed to our research was created. In this framework, it is possible to load the walking data and perform several data processing techniques. Once the human poses are loaded into the framework it is possible to analyze the body keypoints all together or individually and calculate additional information, such as velocity and acceleration of each body keypoint. The framework is also organized in a way that developing and training new models should not take much effort and is considered to be another contribution from this work.

Another contribution from this work is the creation of a dataset with 113 GB of data of people walking in the same environment. The algorithms and models applied on the data in this research can be expanded and many other approaches can be taken to explore the data collected.

# Appendices

## AppendixA DATASET TABLE

In Table A.1 we describe the data collected. We had 4 setups from which the data was collected at different times. The table shows the setup number, the name of the rosbag file containing information, the size of the file, the number of frames recorded and a description of the actions recorded. When using the dataset this information can be very helpful. In order to reproduce the data, the files described in the table have to be open and played in ROS environment. In total, we recorded 113 GB of raw data, 28247 frames equivalent to 4708 seconds. The data from setup 1 and 2 were used in the initial approach from Section 5.1. The data from setup 3 were used in the points of interest approach from Section 5.2. The data from setup 4 were not used in the experiments but can be used in future work.

| Setup | Rosbag file | Size [GB] | Frames | Action |
|---|---|---|---|---|
| 1 | 2018-02-15-21-13-29 | 1.70 | 184 | Random Walk |
| 1 | 2018-02-15-21-14-13 | 1.10 | 92 | Random Walk |
| 1 | 2018-02-15-21-15-07 | 0.52 | 260 | Random Walk |
| 1 | 2018-02-15-21-15-54 | 0.38 | 194 | Random Walk |
| 1 | 2018-02-15-21-16-37 | 0.49 | 246 | Random Walk |
| 1 | 2018-02-15-21-18-49 | 0.44 | 224 | Random Walk |
| 1 | 2018-02-15-21-19-46 | 0.58 | 274 | Random Walk |
| | Sum of setup number 1 | 5.21 | 1474 | |
| 2 | 2018-02-15-20-52-53 | 5.30 | 496 | Random Actions |
| 2 | 2018-02-15-20-55-54 | 2.30 | 245 | Random Actions |
| 2 | 2018-02-15-20-57-27 | 1.92 | 212 | Random Actions |
| 2 | 2018-02-15-20-58-24 | 2.11 | 227 | Random Actions |
| 2 | 2018-02-15-20-59-25 | 1.82 | 196 | Random Actions |
| 2 | 2018-02-15-21-00-18 | 1.42 | 160 | Random Actions |
| 2 | 2018-02-15-21-01-05 | 3.18 | 345 | Random Actions |
| 2 | 2018-02-15-21-02-24 | 2.02 | 223 | Random Actions |
| 2 | 2018-02-15-21-03-34 | 0.90 | 98 | Random Actions |
| 2 | 2018-02-15-21-04-13 | 2.30 | 258 | Random Actions |
| 2 | 2018-02-15-21-09-38 | 1.01 | 112 | Random Actions |
| 2 | 2018-02-15-21-12-00 | 1.75 | 194 | Random Actions |
| | Sum of setup number 2 | 26.03 | 2766 | |

| | | | | |
|---|---|---|---|---|
| 3 | bottomLeftToBackTable1 | 0.43 | 219 | Path to Back Table (bt) |
| 3 | bottomLeftToBackTable2 | 0.44 | 223 | Path to Back Table (bt) |
| 3 | bottomLeftToBackTable3 | 0.42 | 212 | Path to Back Table (bt) |
| 3 | bottomLeftToBackTable4 | 0.42 | 210 | Path to Back Table (bt) |
| 3 | bottomLeftToBackTable5 | 0.39 | 199 | Path to Back Table (bt) |
| 3 | bottomRightToBackTable1 | 0.35 | 174 | Path to Back Table (bt) |
| 3 | bottomRightToBackTable2 | 0.31 | 155 | Path to Back Table (bt) |
| 3 | bottomRightToBackTable3 | 0.32 | 159 | Path to Back Table (bt) |
| 3 | bottomRightToBackTable4 | 0.33 | 164 | Path to Back Table (bt) |
| 3 | bottomRightToBackTable5 | 0.33 | 165 | Path to Back Table (bt) |
| 3 | middleKitchenToBackTable1 | 0.40 | 204 | Path to Back Table (bt) |
| 3 | middleKitchenToBackTable2 | 0.42 | 213 | Path to Back Table (bt) |
| 3 | middleKitchenToBackTable3 | 0.38 | 195 | Path to Back Table (bt) |
| 3 | middleKitchenToBackTable4 | 0.40 | 200 | Path to Back Table (bt) |
| 3 | middleKitchenToBackTable5 | 0.39 | 199 | Path to Back Table (bt) |
| 3 | topLeftToBackTable1 | 0.49 | 248 | Path to Back Table (bt) |
| 3 | topLeftToBackTable2 | 0.64 | 321 | Path to Back Table (bt) |
| 3 | topLeftToBackTable3 | 0.48 | 244 | Path to Back Table (bt) |
| 3 | topLeftToBackTable4 | 0.51 | 257 | Path to Back Table (bt) |
| 3 | topLeftToBackTable5 | 0.45 | 227 | Path to Back Table (bt) |
| 3 | topRightToBackTable1 | 0.45 | 229 | Path to Back Table (bt) |
| 3 | topRightToBackTable2 | 0.43 | 217 | Path to Back Table (bt) |
| 3 | topRightToBackTable3 | 0.39 | 198 | Path to Back Table (bt) |
| 3 | topRightToBackTable4 | 0.44 | 222 | Path to Back Table (bt) |
| 3 | topRightToBackTable5 | 0.41 | 205 | Path to Back Table (bt) |
| 3 | bottomLeftToCoffeeMachine1 | 0.35 | 179 | Path to Cofee Machine (cm) |
| 3 | bottomLeftToCoffeeMachine2 | 0.35 | 181 | Path to Cofee Machine (cm) |
| 3 | bottomLeftToCoffeeMachine3 | 0.38 | 192 | Path to Cofee Machine (cm) |
| 3 | bottomLeftToCoffeeMachine4 | 0.39 | 198 | Path to Cofee Machine (cm) |
| 3 | bottomLeftToCoffeeMachine5 | 0.38 | 193 | Path to Cofee Machine (cm) |
| 3 | bottomRightToCoffeeMachine1 | 0.22 | 111 | Path to Cofee Machine (cm) |
| 3 | bottomRightToCoffeeMachine2 | 0.24 | 122 | Path to Cofee Machine (cm) |
| 3 | bottomRightToCoffeeMachine3 | 0.20 | 105 | Path to Cofee Machine (cm) |

| | | | | |
|---|---|---|---|---|
| 3 | bottomRightToCoffeeMachine4 | 0.21 | 106 | Path to Cofee Machine (cm) |
| 3 | bottomRightToCoffeeMachine5 | 0.24 | 121 | Path to Cofee Machine (cm) |
| 3 | topLeftToCoffeeMachine1 | 0.68 | 343 | Path to Cofee Machine (cm) |
| 3 | topLeftToCoffeeMachine2 | 0.65 | 331 | Path to Cofee Machine (cm) |
| 3 | topLeftToCoffeeMachine3 | 0.69 | 349 | Path to Cofee Machine (cm) |
| 3 | topLeftToCoffeeMachine4 | 0.68 | 344 | Path to Cofee Machine (cm) |
| 3 | topLeftToCoffeeMachine5 | 0.65 | 331 | Path to Cofee Machine (cm) |
| 3 | topRightToCoffeeMachine1 | 0.44 | 225 | Path to Cofee Machine (cm) |
| 3 | topRightToCoffeeMachine2 | 0.44 | 222 | Path to Cofee Machine (cm) |
| 3 | topRightToCoffeeMachine3 | 0.44 | 225 | Path to Cofee Machine (cm) |
| 3 | topRightToCoffeeMachine4 | 0.46 | 235 | Path to Cofee Machine (cm) |
| 3 | topRightToCoffeeMachine5 | 0.41 | 208 | Path to Cofee Machine (cm) |
| 3 | bottomLeftToExpressoMachine1 | 0.41 | 210 | Path to Expresso Machine (em) |
| 3 | bottomLeftToExpressoMachine2 | 0.39 | 198 | Path to Expresso Machine (em) |
| 3 | bottomLeftToExpressoMachine3 | 0.43 | 221 | Path to Expresso Machine (em) |
| 3 | bottomLeftToExpressoMachine4 | 0.41 | 210 | Path to Expresso Machine (em) |
| 3 | bottomLeftToExpressoMachine5 | 0.42 | 214 | Path to Expresso Machine (em) |
| 3 | bottomRightToExpressoMachine1 | 0.23 | 118 | Path to Expresso Machine (em) |
| 3 | bottomRightToExpressoMachine2 | 0.27 | 137 | Path to Expresso Machine (em) |
| 3 | bottomRightToExpressoMachine3 | 0.24 | 124 | Path to Expresso Machine (em) |
| 3 | bottomRightToExpressoMachine4 | 0.26 | 135 | Path to Expresso Machine (em) |
| 3 | bottomRightToExpressoMachine5 | 0.26 | 135 | Path to Expresso Machine (em) |
| 3 | topLeftToExpressoMachine1 | 0.57 | 288 | Path to Expresso Machine (em) |
| 3 | topLeftToExpressoMachine2 | 0.56 | 283 | Path to Expresso Machine (em) |
| 3 | topLeftToExpressoMachine3 | 0.53 | 268 | Path to Expresso Machine (em) |
| 3 | topLeftToExpressoMachine4 | 0.57 | 289 | Path to Expresso Machine (em) |
| 3 | topLeftToExpressoMachine5 | 0.58 | 296 | Path to Expresso Machine (em) |
| 3 | topRightToExpressoMachine1 | 0.37 | 187 | Path to Expresso Machine (em) |
| 3 | topRightToExpressoMachine2 | 0.37 | 189 | Path to Expresso Machine (em) |
| 3 | topRightToExpressoMachine3 | 0.37 | 189 | Path to Expresso Machine (em) |
| 3 | topRightToExpressoMachine4 | 0.44 | 225 | Path to Expresso Machine (em) |
| 3 | topRightToExpressoMachine5 | 0.35 | 178 | Path to Expresso Machine (em) |
| 3 | bottomLeftToSideTable1 | 0.34 | 174 | Path to Side Table (st) |
| 3 | bottomLeftToSideTable2 | 0.34 | 172 | Path to Side Table (st) |

| | | | | |
|---|---|---|---|---|
| 3 | bottomLeftToSideTable3 | 0.36 | 186 | Path to Side Table (st) |
| 3 | bottomLeftToSideTable4 | 0.34 | 174 | Path to Side Table (st) |
| 3 | bottomLeftToSideTable5 | 0.32 | 166 | Path to Side Table (st) |
| 3 | bottomRightToSideTable1 | 0.32 | 165 | Path to Side Table (st) |
| 3 | bottomRightToSideTable2 | 0.25 | 126 | Path to Side Table (st) |
| 3 | bottomRightToSideTable3 | 0.23 | 120 | Path to Side Table (st) |
| 3 | bottomRightToSideTable4 | 0.24 | 125 | Path to Side Table (st) |
| 3 | bottomRightToSideTable5 | 0.24 | 123 | Path to Side Table (st) |
| 3 | middleKitchenToSideTable1 | 0.48 | 245 | Path to Side Table (st) |
| 3 | middleKitchenToSideTable2 | 0.47 | 238 | Path to Side Table (st) |
| 3 | middleKitchenToSideTable3 | 0.46 | 234 | Path to Side Table (st) |
| 3 | middleKitchenToSideTable4 | 0.46 | 236 | Path to Side Table (st) |
| 3 | middleKitchenToSideTable5 | 0.46 | 234 | Path to Side Table (st) |
| 3 | middleKitchenToSideTable6 | 0.46 | 232 | Path to Side Table (st) |
| 3 | topLeftToSideTable1 | 0.62 | 315 | Path to Side Table (st) |
| 3 | topLeftToSideTable2 | 0.54 | 273 | Path to Side Table (st) |
| 3 | topLeftToSideTable3 | 0.57 | 292 | Path to Side Table (st) |
| 3 | topLeftToSideTable4 | 0.57 | 291 | Path to Side Table (st) |
| 3 | topLeftToSideTable5 | 0.59 | 302 | Path to Side Table (st) |
| 3 | topRightToSideTable1 | 0.47 | 240 | Path to Side Table (st) |
| 3 | topRightToSideTable2 | 0.48 | 243 | Path to Side Table (st) |
| 3 | topRightToSideTable3 | 0.50 | 254 | Path to Side Table (st) |
| 3 | topRightToSideTable4 | 0.50 | 255 | Path to Side Table (st) |
| 3 | topRightToSideTable5 | 0.48 | 242 | Path to Side Table (st) |
| | Sum of setup number 3 | 38.04 | 19331 | |
| 4 | 2018-11-16-13-30-55 | 2.20 | 242 | Using Dishwasher |
| 4 | 2018-11-16-13-31-39 | 3.61 | 432 | Using Dishwasher |
| 4 | 2018-11-16-13-33-10 | 1.52 | 168 | Using Dishwasher |
| 4 | 2018-11-16-13-33-49 | 1.42 | 148 | Using Dishwasher |
| 4 | 2018-11-16-13-34-24 | 1.30 | 138 | Using Dishwasher |
| 4 | 2018-11-16-13-34-57 | 1.27 | 131 | Using Dishwasher |
| 4 | 2018-11-16-13-36-12 | 2.00 | 222 | Using Dishwasher |
| 4 | 2018-11-16-13-36-55 | 0.89 | 93 | Using Dishwasher |
| 4 | 2018-11-16-13-37-22 | 2.20 | 245 | Using Dishwasher |

| | | | | |
|---|---|---|---|---|
| 4 | 2018-11-16-10-22-42 | 2.26 | 244 | Getting Coffee |
| 4 | 2018-11-16-10-23-26 | 2.00 | 170 | Getting Coffee |
| 4 | 2018-11-16-13-29-11 | 2.42 | 262 | Getting Coffee |
| 4 | 2018-11-16-10-01-59 | 1.11 | 119 | Getting Coffee |
| 4 | 2018-11-16-10-03-50 | 1.00 | 109 | Getting Coffee |
| 4 | 2018-11-16-10-04-13 | 1.01 | 106 | Getting Coffee |
| 4 | 2018-11-16-10-05-23 | 0.95 | 100 | Getting Coffee |
| 4 | 2018-11-16-10-05-47 | 1.01 | 112 | Getting Coffee |
| 4 | 2018-11-16-10-06-10 | 0.93 | 98 | Getting Coffee |
| 4 | 2018-11-16-14-25-17 | 0.66 | 71 | Getting Tea |
| 4 | 2018-11-16-15-26-11 | 1.2 | 134 | Getting Tea |
| 4 | 2018-11-16-15-27-04 | 0.78 | 82 | Getting Tea |
| 4 | 2018-11-16-15-27-31 | 0.83 | 87 | Getting Tea |
| 4 | 2018-11-16-15-27-54 | 0.66 | 60 | Getting Tea |
| 4 | 2018-11-16-15-28-34 | 0.74 | 79 | Getting Tea |
| 4 | 2018-11-16-15-29-52 | 0.84 | 89 | Getting Tea |
| 4 | 2018-11-16-13-42-04 | 1.40 | 155 | Boiling Water |
| 4 | 2018-11-16-13-42-40 | 1.12 | 122 | Boiling Water |
| 4 | 2018-11-16-13-43-05 | 1.24 | 130 | Boiling Water |
| 4 | 2018-11-16-13-43-33 | 1.26 | 134 | Boiling Water |
| 4 | 2018-11-16-13-44-37 | 1.20 | 134 | Boiling Water |
| 4 | 2018-11-16-13-51-56 | 1.22 | 126 | Boiling Water |
| 4 | 2018-11-16-13-52-35 | 1.21 | 134 | Boiling Water |
| | Sum of setup number 4 | 43.46 | 4676 | |
| | Sum of all setups | 113 | 28247 | |

Table A.1: Dataset table describing each file created in the data collection process. Data collection was performed with 4 different setups. In the table we show the size of each file, the number of frames recorded, and the action being performed. In total the dataset has 113 GB and 28247 frames equivalent to 4708 seconds.

# AppendixB DATA COLLECTION FORM

# Request for participation in research project

# "Walking Direction Prediction"

**Background and Purpose**

The purpose of this project is to create a model to predict human direction during walk using machine learning and Convolutional Neural Network or Recursive Neural Network. This project is being developed as a master project of a student from UFRGS(Brazil) during an exchange year in UiO as part of the cooperation between the two institutions.

Participants are being asked to be recorded so data of people walking can be used as a training data for a neural network in order to be able to create a good walking prediction model.

**What does participation in the project imply?**

Participation in the project implies being recorded by a regular camera and an IR (Infra Red) camera in an environment where people usually walk with defined goals, such as a kitchen. For instance in the kitchen environment people participating will be recorded walking towards specific kitchen areas performing regular activities in the kitchen, much as people do on a daily basis. The duration of the data collection for each individual will be dependent on how long the individual stays in the kitchen. No record of the individual such as name, age, and other personal information will be collected, only the images from the cameras.

**What will happen to the information about you?**
All personal data will be treated confidentially. Only researchers from UiO related to the project will have access to the data. The data will be stored in a portable hard drive that will never leave the department of informatics of UiO.

It is not expected that participants will be recognized in the publication.

The project is scheduled for completion by september 2019. After that time there are no plans for the usage of data.

**Voluntary participation**
It is voluntary to participate in the project, and you can at any time choose to withdraw your consent without stating any reason. If you decide to withdraw, all your personal data will be made anonymous.

If you would like to participate or if you have any questions concerning the project, please contact MSc student Guilherme Antonio Camelo +47 48671565 or advisors Justinas Miseikis (Ph.D. Candidate), and Professor Jim Tørresen +47-22852454.

The study has been notified to the Data Protection Official for Research, NSD - Norwegian Centre for Research Data.

# Consent for participation in the study

I have received information about the project and am willing to participate, I agree that my personal information may be saved after project completion.

---

(Signed by participant, date)

# REFERENCES

ABADI, M. et al. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. 2015. Software available from tensorflow.org. Available from Internet: <https://www.tensorflow.org/>.

ALAHI, A. et al. Social LSTM: Human Trajectory Prediction in Crowded Spaces. **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, p. 961–971, 2016. ISSN 10636919. Available from Internet: <http://ieeexplore.ieee.org/document/7780479/>.

ALTCHE, F.; FORTELLE, A. de L. An lstm network for highway trajectory prediction. In: IEEE. **2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)**. [S.l.], 2017. p. 353–359.

BROWNLEE, J. How to use the timedistributed layer for lstm. **Machine Learning Mastery. Available at https://machinelearningmastery.com/**, 2018.

BROWNLEE, J. A gentle introduction to the challenge of training deep learning neural network models. **Machine Learning Mastery. Available at https://machinelearningmastery.com/a-gentle-introduction-to-the-challenge-of-training-deep-learning-neural-network-models/**, 2019.

BÜTEPAGE, J.; KJELLSTRÖM, H.; KRAGIC, D. Anticipating many futures: Online human motion prediction and synthesis for human-robot collaboration. 2017.

CAO, Z. et al. Realtime multi-person 2d pose estimation using part affinity fields. In: **CVPR**. [S.l.: s.n.], 2017.

CHAPMAN, P. et al. Crisp-dm 1.0 step-by-step data mining guide. 2000.

CHEN, C.-H.; RAMANAN, D. 3d human pose estimation= 2d pose estimation+ matching. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2017. p. 7035–7043.

CHOLLET, F. et al. **Keras**. [S.l.]: GitHub, 2015. <https://github.com/fchollet/keras>.

CIOS, K. J.; KURGAN, L. A. Trends in data mining and knowledge discovery. In: **Advanced techniques in knowledge discovery and data mining**. [S.l.]: Springer, 2005. p. 1–26.

DOLLAR, P. et al. Pedestrian detection: An evaluation of the state of the art. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 34, n. 4, p. 743–761, 2012.

FANG, H.-S. et al. RMPE: Regional multi-person pose estimation. In: **ICCV**. [S.l.: s.n.], 2017.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. **AI magazine**, v. 17, n. 3, p. 37–37, 1996.

FRANÇOIS, D. Methodology and standards for data analysis with machine learning tools. Citeseer, 2008.

GERS, F. A.; SCHMIDHUBER, J.; CUMMINS, F. Learning to forget: Continual prediction with lstm. IET, 1999.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT press, 2016.

GRAVES, A. Generating sequences with recurrent neural networks. **arXiv preprint arXiv:1308.0850**, 2013.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

INSTITUTE, S. The enterprize miner - semma. 2008.

IONESCU, C. et al. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 36, n. 7, p. 1325–1339, 2014.

IONESCU, C. et al. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. **IEEE Trans. Pattern Anal. Mach. Intell.**, IEEE Computer Society, Washington, DC, USA, v. 36, n. 7, p. 1325–1339, jul. 2014. ISSN 0162-8828. Available from Internet: <https://doi.org/10.1109/TPAMI.2013.248>.

IVANOV, S. **37 Reasons why your Neural Network is not working**. 2017. Accessed: 2018-03-10. Available from Internet: <https://blog.slavv.com/37-reasons-why-your-neural-network-is-not-working-4020854bd607>.

JIANG, H. 3d human pose reconstruction using millions of exemplars. In: IEEE. **2010 20th International Conference on Pattern Recognition**. [S.l.], 2010. p. 1674–1677.

JIAO, L. et al. Anatomy of a multicamera video surveillance system. **Multimedia systems**, Springer, v. 10, n. 2, p. 144–163, 2004.

JUNG, Y.-K.; HO, Y.-S. Traffic parameter extraction using video-based vehicle tracking. In: IEEE. **Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (Cat. No. 99TH8383)**. [S.l.], 1999. p. 764–769.

KAMIJO, S. et al. Traffic monitoring and accident detection at intersections. **IEEE transactions on Intelligent transportation systems**, IEEE, v. 1, n. 2, p. 108–118, 2000.

LEE, H.-J.; CHEN, Z. Determination of 3d human body postures from a single view. **Computer Vision, Graphics, and Image Processing**, Elsevier, v. 30, n. 2, p. 148–168, 1985.

LERNER, A.; CHRYSANTHOU, Y.; LISCHINSKI, D. Crowds by example. **Computer Graphics Forum**, v. 26, n. 3, p. 655–664. Available from Internet: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2007.01089.x>.

MAJUMDAR, A. **Mouse Tracking Predictor**. 2017. Available from Internet: <https://github.com/abhijitmajumdar/Mouse_tracking_predictor>.

MARTINEZ, J.; BLACK, M. J.; ROMERO, J. On human motion prediction using recurrent neural networks. 2017. Available from Internet: <http://arxiv.org/abs/1705.02445>.

MATLAB. **version 7.10.0 (R2010a)**. Natick, Massachusetts: The MathWorks Inc., 2010.

MIŠEIKIS, J. **Joint human detection from on-board and off-board cameras**. Dissertation (Master) — ETH-Zürich, 2012.

MORRIS, B. T.; TRIVEDI, M. M. Learning and classification of trajectories in dynamic scenes: A general framework for live video analysis. In: IEEE. **2008 IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance**. [S.l.], 2008. p. 154–161.

MORRIS, B. T.; TRIVEDI, M. M. A survey of vision-based trajectory learning and analysis for surveillance. **IEEE transactions on circuits and systems for video technology**, IEEE, v. 18, n. 8, p. 1114–1127, 2008.

OPENCV. **Camera calibration With OpenCV**. 2017. Available from Internet: <https://docs.opencv.org/master/d4/d94/tutorial_camera_calibration.html>.

PELLEGRINI, S. et al. You'll never walk alone: Modeling social behavior for multi-target tracking. In: **2009 IEEE 12th International Conference on Computer Vision**. [S.l.: s.n.], 2009. p. 261–268. ISSN 2380-7504.

Phillips, D. J.; Wheeler, T. A.; Kochenderfer, M. J. Generalizable intention prediction of human drivers at intersections. **IEEE Intelligent Vehicles Symposium (IV)**, 2017.

POLLACK, M. E. et al. Pearl: A mobile robotic assistant for the elderly. In: **AAAI workshop on automation as eldercare**. [S.l.: s.n.], 2002. v. 2002, p. 85–91.

PYCHARM, I. For professional developers. **Available at: https://www.jetbrains.com/pycharm**, 2016.

Python Core Team . **Python: A dynamic, open source programming language.** [S.l.], 2019. Available from Internet: <https://www.python.org/>.

QUIGLEY, M. et al. Ros: an open-source robot operating system. In: **ICRA Workshop on Open Source Software**. [S.l.: s.n.], 2009.

ROBERTSON, N.; REID, I. Behaviour understanding in video: a combined method. In: IEEE. **Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1**. [S.l.], 2005. v. 1, p. 808–815.

SIMON, T. et al. Hand keypoint detection in single images using multiview bootstrapping. In: **CVPR**. [S.l.: s.n.], 2017.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to sequence learning with neural networks. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2014. p. 3104–3112.

VENUGOPALAN, S. et al. Sequence to sequence-video to text. In: **Proceedings of the IEEE international conference on computer vision**. [S.l.: s.n.], 2015. p. 4534–4542.

VINYALS, O. et al. Show and tell: A neural image caption generator. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2015. p. 3156–3164.

WEI, S.-E. et al. Convolutional pose machines. In: **CVPR**. [S.l.: s.n.], 2016.

WIEDEMEYER, T. **IAI Kinect2**. 2017. Available from Internet: <https://github.com/code-iai/iai_kinect2>.

XIU, Y. et al. Pose Flow: Efficient online pose tracking. In: **BMVC**. [S.l.: s.n.], 2018.

XU, K. et al. MultiMedia Modeling. v. 10704, p. 106–116, 2018. ISSN 00200255. Available from Internet: <http://link.springer.com/10.1007/978-3-319-73603-7>.

ZHAO, E. Explaining the terms ai, ml, dl, ds. **Data Science Student Society . Available at https://medium.com/ds3ucsd/explaining-the-terms-ai-ml-dl-ds-b0ac43e99f55**, 2018.