Dataset Synthesis for Activity Recognition

Olivier Jansen
4288092

Ronald Poppe

Master Artificial Intelligence

Utrecht University

Utrecht University

Oddity.ai

Table of Contents

1. Introduction

1.1. Motivation

This section will outline the motivation behind the current research. It is twofold as two distinct problems gave rise to the research questions.

1.1.1. Oddity.ai

At Oddity.ai, a deep learning pipeline for violence detection was developed to help video surveillants detect fights more quickly [1], [2]. The system constantly scans live video feeds and notifies the operator if an incident is detected. This pipeline has proven to be functional in commercial, real-life scenarios. It is hypothesised that its performance will improve further if a larger dataset would be available. However, footage of street violence captured by surveillance cameras is not only scarce but also privacy-sensitive, especially with recent European privacy legislation. This means that collecting more surveillance video footage is not a viable option.

Simulation can offer an alternative when the real thing is not available. For example, when a physics experiment is too expensive to carry out, a simulation can deliver an approximation of the results for a much lower price. An additional advantage is that the simulated world is fully under the control of the researcher. Because it is not possible to gather enough videos of violence, simulation might offer a solution for Oddity.ai. This is how the idea of synthetic video footage for training a better violence recognition system came about.

1.1.2. Dutch National Police

The Dutch National Police is interested in the retroactive identification of burglars on surveillance footage. They could then determine whether the criminal explored the neighbourhood in advance. These criminals often have a disguised appearance and the camera circumstances are often rough (for example, bad lighting conditions or low resolution). This makes it difficult to automatically identify criminals from video using common biometrics such as their face. Currently, video footage is inspected by investigators who try to identify the criminal manually.

This is a very labour-intensive process.

Research shows that people can be identified by their way of walking (their *gait*), which is directly visible from surveillance video. Many recent gait recognition proposals are grounded in deep learning and this shows great promise [3], [4]. This raises the question of how we can acquire sufficient data to train these deep learning models. As the Dutch National Police is looking to recognise gaits on very varied video footage, the dataset used for training must be very diverse as well.

A large and robust dataset for gait recognition could be constructed using the same solution as proposed for Oddity.ai. Motion capture footage of people walking would be the source motion of the video generation process. A large number of videos of people walking, with all kinds of environments and appearances, would be the resulting output.

This scenario is simpler than the violence use case, as there is always just a single subject and the source motion does not need to be extracted from monocular video. Evaluation of the pipeline on this distinct task could provide valuable information about the pipeline in general.

1.1.3. Combining the two problems

In this work, both violence detection and gait recognition will be addressed. A pipeline for the generation of synthetic video footage of human motion will be proposed. The pipeline can take videos containing example motion as input. The motion will be extracted from the video and stored in an intermediate format. Motion data recorded with a motion capture suit can also be taken as input directly, skipping the motion extraction. The pipeline generates datasets that can be used to train machine learning models for activity recognition tasks that are invariant to environment and appearance changes.

Research will be done into a pipeline for the two problems, however, the broader context of activity recognition from video will be kept in mind. This means that generic solutions to the arising problems are preferred over solutions that are tailored to gait or violence.

1.2. Background

In this section, some introductory background on the problem will be given to build up to the formalisation that is given in Section 1.3.

1.2.1. The Case for Violence Detection

The automatic detection of rare or anomalous activities (activities of which few examples are available [5]) from video has the potential to be very useful for society. The seconds or minutes that can be saved by automatically and instantly detecting someone suffering from a stroke in a retirement home could save lives, for example. Automatically recognising fights in city streets enables authorities to react earlier and minimise damage suffered by the attacked.

Currently, a growing number of security cameras is being deployed but human surveillants are still often watching the video feeds. Having humans in place to monitor all feeds all the time is expensive and they are prone to making mistakes. They might miss an incident (due to a large number of video streams) or get distracted and they cannot react instantly. Human surveillants are also undesirable for privacy reasons. No matter how well-trained, humans are inherently biased, which hinders objective analysis, and are equipped with a (biological) face-recognition system. Additionally, human personnel is relatively costly compared to a computer-based system. In short, human surveillants are not optimal at all.

A computer-based automated detection system would be able to recognise rare activities quicker and, because of its digital nature, can dispatch alarm signals right into the appropriate communication systems. With these systems in place, human personnel would no longer be required to monitor all video feeds. They could be used to review the fragments for which an alarm was dispatched. And because only the video features needed for activity recognition are processed (instead of the entire video, as human surveillants do), such a system would also be benficial for privacy.

1.2.2. The Case for Gait Recognition

Many surveillance cameras are not actively monitored but are recording so that footage can retroactively be consulted in case an incident took place. Currently, police investigators manually inspect every minute of the footage to solve a crime or build a file on a suspect. The suspect is already known at this stage but often hours of footage needs to be inspected manually to find this specific person. The same arguments against human personnel as mentioned earlier apply here; they have some severe shortcomings regarding this task.

A computer-based system could be beneficial here as well. Firstly, this system takes a video of the suspect walking and calculates a representation of their gait. It then goes over the available surveillance footage, detects all people and calculates their gait representations. For each of the gait representations, its distance to the suspect's gait representation is calculated. Then the footage is cut into portions if it were not already and is sorted on these distances. The police investigators are presented with this sorted list and will find the suspect in the first section of the list, saving a lot of time.

1.2.3. Deep Learning

The past few years, computer vision has been dominated by upcoming machine learning methods. Machine learning methods model a target function between input and output variables that is visible in the data. The estimated function is used to predict the output variable given some unseen input.

The foundation for modern computer vision was laid with the invention of biologically inspired convolutional neural networks (CNNs) [6] and the demonstration that greatly increasing the number of neural network layers can boost performance dramatically [7]. In the computer vision subfield of activity recognition, currently most state-of-the-art models are based on deep learning and convolutional neural networks [8].

DEEP LEARNING

The subfield of machine learning that focusses on many-layered artificial neural networks.

Activity recognition and gait recognition are complex tasks. To accurately model these tasks, a complex model is required. To get an accurate picture of what the tasks look like, a lot of data with representative examples are required to learn from [9]. Also, a large and diverse dataset is required to avoid overfitting when working with complex models.

This is in line with recent computer vision work, most of which is grounded in deep learning. A recent trend is the construction of larger and larger datasets for computer vision tasks [10]–[13].

Instead of this supervised learning manner, for which labelled data is required, alternative learning paradigms are increasingly popular. These include semi-supervised learning, which brings unsupervised learning techniques to cope with scarcely labelled datasets, and self-supervised learning, which does not require any labelled data at all. The supervised learning paradigm is still predominant, however.

1.2.4. Dataset Synthesis

Recently, synthetic dataset construction has gained attention because of its potential answer to the data requirements of deep learning. Dataset synthesis is the construction of entire datasets without obtaining the data points by measurement. This differs from data augmentation, which expands an existing base dataset (e.g. using extrapolation), as every instance in a synthetic dataset is entirely constructed. Computer vision research is a field that is especially interested in the use of synthetic data because large amounts of representative data for real-world tasks is often hard to gather.

SYNTHETIC DATA

Data that is not acquired by direct measurement but that is deliberately constructed.

Dataset synthesis can not only easily increase the amount of available training data but automatic annotation and labeling of data is also among the advantages. Annotations or labels provide additional information about the scenes in images and they are used as target output for many computer vision tasks. Take for example the task of segmentation, in which a two-dimensional image of a scene is given and a partitioning of the distinct objects in the scene needs to be estimated. Segmentations can then be used for reasoning about the object in a picture. The segmentation task is often approached as a supervised machine learning task. This means that the ground truth partitioning is used as guidance during training, and thus needs to be available. Figure 1.1 is an example of a natural image (left) and two annotations for segmentation tasks (middle and right). When constructing training datasets for these tasks, a segmentation has to be created manually for every image. This process is known as image annotating and it is a very time-consuming and expensive exercise. With data synthesis, the ground truth annotations could be generated just as easily as the image itself, as all details about the scene would be known.

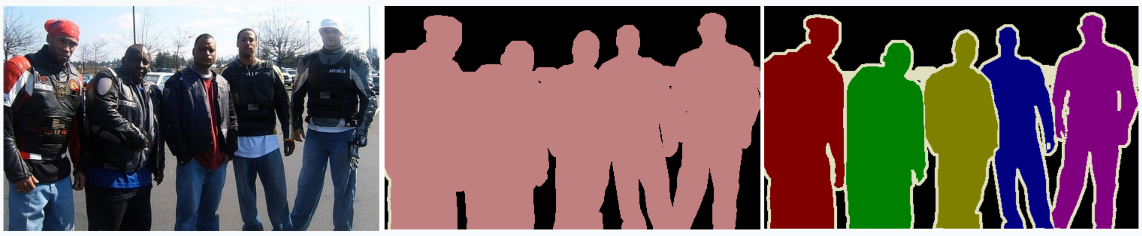


Figure 1.1: Example annotations for segmentation tasks, taken from the Pascal VOC dataset [14]

Dataset synthesis could thus play a role in answering the hunger for data that deep learning methods have, by significantly lowering the cost of collecting large datasets. A lot of research into constructing synthetic datasets for computer vision tasks has already been conducted [15]–[19]. The majority of research proposes an *explicit rendering pipeline* to construct synthetic data. This involves collecting virtual (3D) models, combining models and environments into scenes, animating the scene, applying domain randomisation and rendering this virtual scene into images or video footage. In contrast to this explicit rendering pipeline approach, some research proposes learning a generative model. Synthetic datasets could be constructed by sampling from the generative model. The generative adversarial network architecture is the most promising direction for such generative models but it is still not mature enough to create convincing video footage. More on this approach can be found in subsection 2.5.2 of the Literature Study.

DOMAIN RANDOMISATION

Varying environment or appearance variables during training to stimulate generalisability outside of training.

1.3. Objective

The following research question will guide this research.

MAIN RESEARCH QUESTION

Can training data, that is synthesised using motion data that is captured from alternative sources, improve the classification performance of deep learning-based activity recognition systems?

Alternative sources here means data from a different domain than the test data is from. An example would be to use violence from Hollywood movies to generate synthetic data to train a violence detection model for CCTV footage.

This main research question can be split into the following subquestions:

- **Research Question 1**: With what accuracy can we capture motion from video footage using pose estimation techniques?
- **Research Question 2**: Does training with synthetic data increase performance?
- **Research Question 3**: Do domain randomisation techniques applied to the extracted motion improve classifier performance?

For the answer to Research Question 1, state-of-the-art pose estimation techniques will be investigated and compared. The performance of the motion extractor will be measured using an error metric on the 3D Poses in the Wild Dataset [20], the only real in-the-wild 3D poses dataset with a large amount of video footage. We will also use the Kinetics-400 Dataset as extended by Arnab, Doersch and Zisserman [21] as a second evaluation metric. The poses in this dataset are somewhat less strong as they are automatically generated by the method of the authors but due to the large amount of video data, it could still provide valuable insights. The limitations of using pose estimation to extract motion from video for synthetic data generation will be discussed and addressed.

Research Question 2 will be answered by training and assessing the performance of the systems in question (violence detection and gait recognition). A multitude of datasets will be generated for these training sessions. These datasets will differ in 1) the amount of synthetic data that is added to the base dataset, which consists of only real videos, and 2) the proportion of real videos and synthetic videos. The *area under the curve* of the *receiver operating characteristic* (ROC-AUC) will be used as a comparative classification performance measure as it is a classification threshold-independent measure of discrimination capacity.

For Research Question 3, different sets of synthetic data will be generated using different applications of domain randomisation. The variables that will be varied are the background plane, texture (clothing, hair style and skin colour), body shape and camera angle. For the violence detection task, the number of actors in a scene will also be varied. An ablation study will be done to gain insight into the causal effects of the randomisation of these variables.

1.4. Methods

To answer the main question of this research, an artefact has to be created (namely a data synthesiser). This is a problem in and of itself. Because of its practical nature the *design science paradigm* is adopted. Design science is outcome-focused, the goal is to design a useful artefact through research and development.

This research will follow the Design Science Research Methodologies as outlined by Peffers et al [22], who define the research process as consisting of six distinct steps:

1. **Problem identification and motivation**
   The research problem is outlined, it is motivated why it is a problem and the value of a solution is explained. A short introductory exploration of the problem is offered in Section 1.2. The problem is formalised into research questions in Section 1.3. More elaborate background on the problem is given in Chapter 2 with a literature study.

2. **Definition of the objectives for a solution**
   Domain knowledge and the outcomes of the previous step are used to determine objectives for a solution that should be met for it to be successful. The objectives for a solution, based on the literature study in Chapter 2, are given in Chapter 3.

3. **Design and development**
   The actual designing and building of the artefact is done. The objectives from the previous step are used as guidelines. In Chapter 4 the design and development are described.

4. **Demonstration**
   The artefact is demonstrated and it is shown how it solves the problem. This can be done using e.g. a case study, simulation or proof. This work will not feature a dedicated Demonstration chapter but examples can instead be found in the next chapter. A demonstration will also be given at the defence of this work.

5. **Evaluation**
   The artefact is empirically compared with the objectives. This provides an exact measurement of its effectiveness. In this section, measurements to answer the research questions will also be provided. The evaluation can be found in Chapter 5.

6. **Communication**
   The results from all previous steps are conveyed to an audience. The entirety of this thesis is that communication. The aggregation of the results from previous steps and limitations of this study will have a place in Chapter 6: Discussion.

2. Literature Study

This chapter will shed light upon relevant work in the research areas of interest. First, computer vision, in general, will be covered in Section 2.1. The rise of convolutional neural networks and deep learning for computer vision will then be dealt with (Sections 2.2 and 2.3) and action and activity recognition will be discussed in depth (Section 2.4). After the stage has been set, the jump will be made to literature on data synthesis in Section 2.5. Pose estimation will be explored as it is a component of the theorised solution, in Section 2.6.

2.1. Computer Vision

Computer vision, concerned with giving computers a full understanding of image content, has always held an important place within artificial intelligence research. Traditionally, computer vision has approached image processing as a problem that requires a lot of prior knowledge to be programmed into the solution [23]. Low-level operations, such as edge extraction and specific convolutions, are combined thoughtfully and deliberately to gain a high-level understanding.

The past few decades, computer vision has largely shifted towards a feature-based and data-driven approach. Biologically inspired convolutional neural networks and advancements in hardware were the driving forces behind this shift. The focus moved from designing representations towards automatically learning from examples.

This shift did not only take place within computer vision research but it occured in the whole of the artificial intelligence field. Progress booked with the representation-based approach to AI, also called symbolic or GOFAI ("Good Old-Fashioned Artificial Intelligence"), stagnated and the subsymbolic (or connectionist) approach gained popularity in research. More detailed background and discussion of this dichotomy can be found in AI literature of a more philosophical nature [24].

2.2. Convolutional Neural Networks

In 1980, Kunihiko Fukushima published his work on the Neocognitron [25]. Inspired by earlier research by Hubel and Wiesel on the human visual nervous system, he developed a neural network model that could self-organise such that it recognises stimulus patterns.

Hubel and Wiesel found that the visual primary cortex consisted of *simple cells* and *complex cells* and that these are organised hierarchically [26]. Simple cells respond to oriented edges in a specific receptive field, complex cells integrate (also called "pooling") multiple simple cells and are able to respond to patterns within a larger receptive field. An important aspect of this is that nearby cells in the cortex represent nearby regions in the visual field. Analogous to these findings, Fukushima proposed a neural network model. This model has an initial layer of *S-cells*, responding to local, spatial patterns in the input (modifiable parameter determine how they response exactly), and a subsequent layer with *C-cells* that integrates active *S-cells* from the previous layer. Pairs of *S-cell* layers and *C-cell* layers are stacked to form a complete model. The Neocognitron has been succesfully applied by Fukushima to the task of digit recognition.

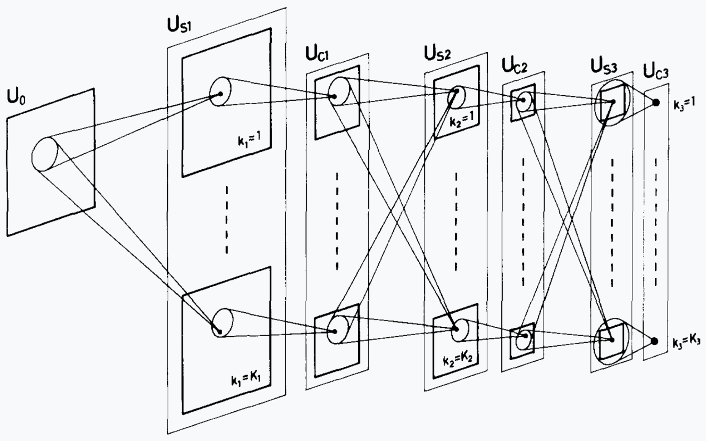


Figure 2.1: A schematic representation of the Neocognitron, where U_0 represents the image input and the following layers contain alternately *S-cells* and *C-cells*. Figure from [25]

Almost a decade later, in 1989, Yann LeCun succesfully applied the backpropagation algorithm to train a network that is very much like the Neocognitron [27] and achieved good classification results on a digit recognition task [6].

BACKPROPAGATION

Widely used algorithm for training neural networks given a loss function. Backpropagation computes the gradient of the model parameters with respect to the loss function. This makes it possible to find the model parameters that minimise the loss function, using gradient optimisation methods.

The kind of neural network described by LeCunn is characterised by the use of convolution and pooling operations, and it was properly dubbed the *convolutional neural network*. Convolution layers iterate over each item in the input and add them to their local neighbours, weighted by a small matrix called the *kernel*. This works analogous to the *S-cells* and effectively applies a filter to the input. Each kernel that the convolution layer applies to the input results in an activation map. The most important benefit of this is that CNNs can take the spatial geometry of the input data into account.

After a convolution layer, a pooling layer often follows. Pooling layers take a feature map and downsample it (e.g. by replacing 2x2 patches of a matrix by their maximum value, halving the size of the matrix). This keeps the computational load within bounds, introduces a non-linearity into the function that the network models and improves the networks invariance to deformation of the input.

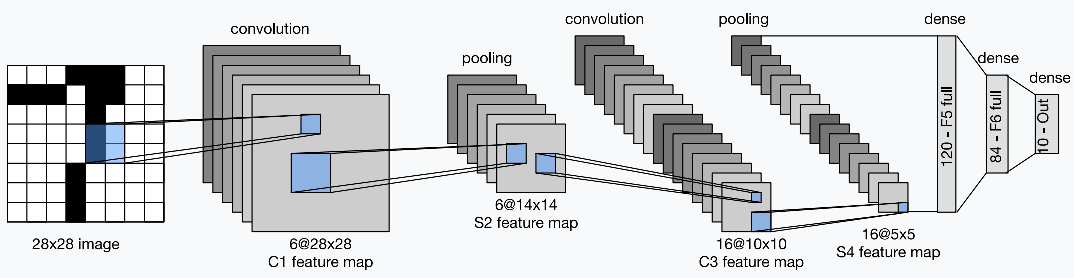


Figure 2.2: The architecture of LeNet-5, the convolution neural network as proposed by LeCun in [6]. Figure from [28]

LeCun applied his convolutional neural network, named LeNet-5, to digit recognition, just like Fukushima, and achieved good results because of the spatial awareness of CNNs. The digit recognition models were widely deployed in postal services to recognise ZIP codes. However, interest in CNNs beyond digit recognition declined as they were relatively computationally costly.

2.3. Deep Learning

Although neural network research was already somewhat revitalised with CNNs, at the beginning of the millennium it was still computationally infeasible to apply the techniques to larger and more demanding problems such as natural image classification. This changed when researchers started using a graphics processing unit (GPU), combined with highly optimized implementations, to more efficiently train models. This allowed for more and more layers to be added to models (hence *deep* learning) while maintaining computational feasibility. The work of Ciresan from 2010 was among the first to show a significant performance gain in an image classification task with the use of a deeper model trained using a GPU [29]. With AlexNet, an 8-layer deep CNN, the biggest blow for deep learning was struck [30]. It achieved state-of-the-art performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 by a very large margin. This shifted the attention of the whole computer vision field towards deep CNNs and also sparked interest for deep learning outside of computer vision.

With this significant increase in the number of layers, these deep neural networks gained more capacity to learn from raw input data, automating the feature extraction process. Deep neural networks can take very large amounts of raw data (such as raw pixel data of images) as input. They perform sufficient operations on them to allow the formation of abstract representations.

2.4. Action and Activity Recognition

The computer vision subfield of action and activity recognition is concerned with the detection and classification of human actions and activities in video footage.

ACTIVITY

Refers to a subsequent number of actions [31], [32], where each action consists of action primitives, i.e. movements that can be described at the limb level. For example, the activity ”110 metres hurdling” consists of ”jumping a hurdle”, which in turn consists of action primitives ”bending right knee”, among others.

2.4.1. Representation Based vs. Deep Networks Based

In a survey on action recognition methods by Herath, Harandi and Porikli [33], a taxonomical breakdown of the solutions found in literature is offered. This breakdown follows the symbolic vs. subsymbolic divide we saw before in Section 2.1.

1. **Representation based solutions**
   Solutions that are based on the handcrafted extraction of information. A further distinction between holistic and local representation based solutions is made. The first extract representations of the global state of the body structure, shape and movement. The latter are concerned with the extraction of a variety of local representations, using e.g. edges, corners and points of interest.

2. **Deep networks based solutions**
   Solutions that use a deep convolutional neural network to recognise actions and activities in a data-driven fashion.

We are only interested in the deep networks based solutions for this work as the research problem arises from the data-driven nature of deep learning, something that the representation based solutions do not suffer from. According to an analysis by Herath, Harandi and Porikli, the state-of-the-art solutions of both types perform equally well. This is somewhat surprising as deep learning-based solutions for many other computer vision tasks do outperform representation based solutions. However, a lack of sufficient training data is among the possible factors limiting the performance of deep learning models for activity recognition, according to Herath et al. Work published after this survey seems to support this thesis, see e.g. [12], [34] for research in which larger datasets are successfully used.

Different approaches to creating deep network architectures for activity recognition are discussed by Herath et al. They have in common that they try to combine the spatial awareness of CNNs with a temporal (i.e. time) aspect. Because activities take place over time, they cannot be recognised from a single image or video frame. Qualified methods for activity recognition should thus properly take the sequence of frames into account, and not just a single frame at a time. Several network architectures that take the temporal aspect of activities into account have been proposed. We follow the categorisation of Herath, Harandi and Porikli in the following summary.

1. **Spatiotemporal networks** introduce convolutional layers that also extract features over the time axis, with convolutional filters (or kernels) that extend to a third dimension, for time. This operation is called a *3D convolution* as opposed to the original 2D convolution that only spans spatial dimensions. The number of frames that the network accepts is hardwired into the network structure as it is one aspect of the filter size.

2. **Multiple stream networks** are inspired by the distinction between appearance and motion found in our brain, expressed as the ventral and dorsal streams. Two distinct networks, one for object recognition and one for motion recognition, are combined into a single output score. The object recognition network is a fairly regular CNN that takes single frames as input. The motion recognition network is also CNN based but gets multiple optical flow fields as input.

3. **Deep generative models** try to learn the underlying distribution of data and can be used in an unsupervised setting. Especially the *auto-encoding network*, that contains a bottleneck to compress the feature representation, is a popular generative model.

4. **Temporal coherency networks** model the coherency between individual frames in a video. The intuition behind this architecture is that sudden and abrupt motions are less likely. The applications of generative models and temporal coherency networks to activity recognition are limited and they are mostly used in conjunction with other architectures.

5. **Long short-term memory networks** have an architecture that allows for information to be persistant over several inputs (i.e. have some sort of short-term memory). What part of the input should be remembered for a while is also learnt and is referred to as the long term memory of the LSTM network. LSTM units are often incorporated in existing network architectures to add a temporal component.

2.4.2. Violence Recognition

Violence recognition is a niche case of activity recognition and it offers some additional challenges to overcome. Because violent behaviour is socially controversial, less training data is readily available than for other behaviours. Even though training data for activity recognition can be found organised in publicly available datasets, these contain no or few examples of violence. An additional difficulty is that violence can be both an individual act as well as interactive behaviour. This is in contrast to e.g. running, which is always individual, or hugging, which is always interactive.

2.5. Data Synthesis

Recent research indicates that limited available data might be the main bottleneck for current deep learning-based computer vision solutions. Sun et al (2017) [9] and Hestness et al (2017) [35] for example, show that an increase in dataset size can lead to significantly increased performance. Synthetic data has been shown to be a very promising way to address this bottleneck [36]. The usage of synthetic data for computer vision problems goes back to the late 1980s [37] and early 1990s [38], according to an extensive survey on the topic by Nikolenko [19]. It started with data synthesis for more low-level problems, such as optical flow estimation or stereo image matching, and progressed to higher-level problems, such as object detection.

Most of the proposed synthetic datasets for computer vision are constructed with low-level computer vision tasks in mind. Examples of such tasks are depth estimation and semantic segmentation. To our knowledge there is no work that looks into constructing synthetic video datasets for violence detection. There is one work on the use of synthetic data for gait recognition, that of Charalambous and Bharath from 2016 [39]. The authors use motion-captured data of people walking to generate large amounts of video footage of simulated humans walking under different kinds of environment and appearance circumstances. However, the pipeline proposed by Charalambous and Bharath 1) does not extend to multiple people in one scene, 2) has only 26 motion sources and 3) does not function from end to end automatically. The current work does address these limitations. Although Charalambous and Bharath call this *data augmentation*, we consider this to be a case of *data synthesis* as the authors do not have a base dataset of videos that could be used as-is for their problem but instead have motion-captured data.

The extension to synthetic people, according to the survey by Nikolenko [19], makes sense for privacy reasons, because of the complexity of manually annotating humans and because of biases in current datasets. However, he notes, synthetic three-dimensional models of humans are especially hard to create and because human-related tasks already play such a large role in computer vision, there already exist good large datasets. This limits the use of synthetic humans in normal scenarios, limiting it to specific edge cases for which no good datasets exist. An alternative is to use simpler methods to extend these existing datasets. For example, Enzweiler and Gavrilla [40] change colours of clothing and the 2D body shape of pedestrians to complement an existing dataset.

2.5.1. Domain Adaptation and the Reality Gap

For this work, we are interested in the use of synthetic data to train a model that will be used for inference in real-world scenarios. Synthetic data and real-world data have different origins so their distributions are not the same. To use synthetic data, we now first have to make the following assumption: whatever is learnt from the synthetic data is also applicable to real-world data. The research of this assumption is known as *transfer learning* (TL) and its researchers are interested in all scenarios where knowledge learnt from solving one problem is applied to another problem. In TL, the problem space where the knowledge is gained is called the *source domain* and the problem space where the knowledge is to be applied is called the *target domain*. In our specific case, the target domain is recognition on real-world surveillance video and the source domain is recognition on synthetic data. Labelled data is not available in the target domain but we can create it for the source domain, as we control the data synthesis process. The availability of labels in only the source domain makes the proposal of this work and instance of *transductive transfer learning* (see Figure 2.3).

Domain adaption is concerned with the transformation of the source domain to increase its similarity with the target domain. The goal is to increase the likelihood that knowledge learnt in the source domain is applicable to the target domain. This process is referred to as *unsupervised* domain adaptation when the labels in the target domain are not known, as is the case with this work's topic of interest.

Usually it is necessary to determine how related a task is in both domains before domain adaptation should be considered. With synthetic data as the source domain, we have a fair amount of control over the source domain. We thus fabricate a source domain that is as related to the target domain as possible, decreasing the amount of work left for domain adaptation to a minimum.

Training on synthetic data to perform some action on real data, is also referred to as a *sim-to-real* transfer in recent literature [42], [43]. Figuring out how to construct synthetic data such that it can be used for a sim-to-real transfer, is *closing the reality gap*. Here the *reality gap* refers to the discrepency between the simulated data and the real data. There is sufficient and diverse work to be found on closing the reality gap for deep learning methods.

A recent survey by Wilson and Cook (2019) [44] is used as a main guide for definitions and directions as we take a closer look at the field of unsupervised domain adaption. Wilson and Cook identify the following distinct approaches:

- **Domain-invariant feature learning**
  Promote the learning of a domain-invariant representation of relevant features from the training data. For example, a dogs versus cows classifier might pick up the environment (e.g. is the animal indoors or in a meadow) as a meaningful indicator because the training data shows cows in the meadow

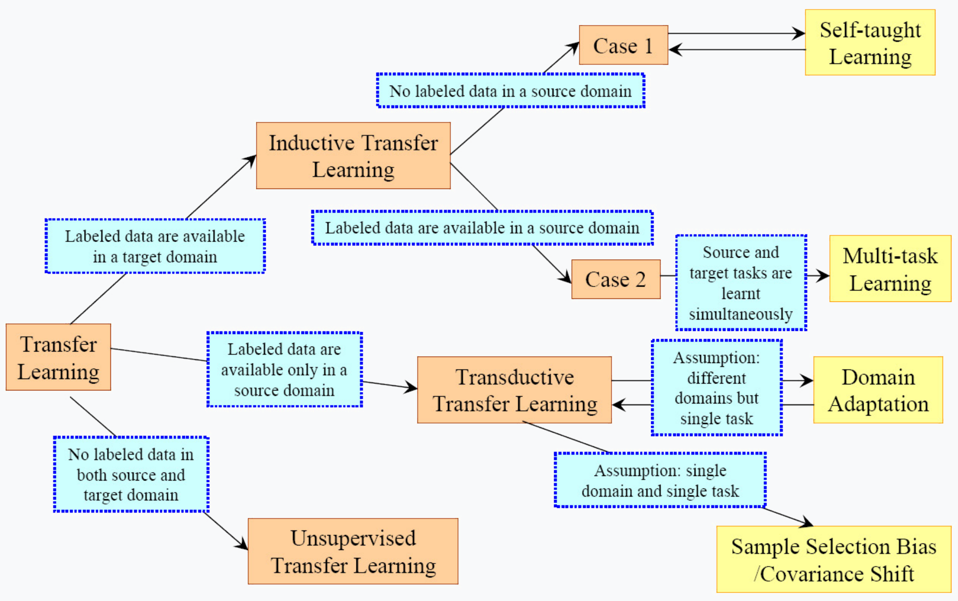

Self-taught Learning
Case 1
No labeled data in a source domain
Inductive Transfer Learning
Labeled data are available in a target domain
Labeled data are available in a source domain
Source and target tasks are learnt simultaneously
Multi-task Learning
Case 2
Transfer Learning
Labeled data are available only in a source domain
Transductive Transfer Learning
Assumption: different domains but single task
Domain Adaptation
No labeled data in both source and target domain
Assumption: single domain and single task
Unsupervised Transfer Learning
Sample Selection Bias /Covariance Shift

Figure 2.3: An overview of different types of transfer learning. Figure from Pan and Yang (2009) [41].

disproportionately often. If the trained classifier was used by a veterinary office to classify animals in the waiting room, it will be wrongly biased towards dogs because of the indoor environment. Domain-invariant feature learning tries to prevent this behaviour and can be divided further into:

- *Divergence-based*
  Define a measure of domain divergence, often some sort of statistical test or correlation measure is picked. In the intermediate layers of the classifier, try to minimise this divergence measure along with the primary classification error.
- *Reconstruction-based*
  Train a single network on two tasks simultaneously: classifying labelled source data and reconstructing target data. The learnt representation captures information about the classification as well as the target domain.
- *Adversarial-based*
  A domain-classifier that tries to determine which domain is behind a given representation is added as an adversarial to the network that performs the main task. This can also be achieved by directly adding a *domain confusion loss* that acts as an adversarial. The Domain-Adversarial Neural Network [45] is a notable example of this.

• **Domain mapping**
Map the input data from the source domain to make it look like it came from the target domain before it is used as input to the classifier. This is often done using a *generative adversarial network* (GAN), in which a generative model is trained against a discriminative opponent in a game-theoretic setting. The GAN architecture is discussed in more detail in Section 2.5.2.

Simpler visual processing is also suggested [42].

- **Normalization statistics**
  Use techniques like batch normalisation to adapt an already-trained network to a different domain by changing the batch norm weights when applying to a different domain. The benefits of using this approach seem limited, according to Wilson and Cook.

- **Ensemble methods**
  Train multiple instances of a model with different initialisations to filter out source domain specific errors.

An alternative to domain adaptation for synthetic data is controlled domain randomisation. This is the intentional application of random permutations to the synthetic source data, in order to force the model to focus on the item of interest. In Tremblay et al (2018) [46] for example, synthetic cars are rendered on top of random backgrounds. Random flying shapes are added, the lighting and viewpoint are randomised and random textures are applied to all objects. They show that this randomisation in training data can improve the test performance of an object detection model that is to detect cars. This makes intuitive sense. A model that works in all of the random environments generated, will likely also work in the environment of the real world. By training on a variety of values for environment variables and keeping the motion constantly clear, the features that will be extracted will become invariant for those environment variables.

2.5.2. Generative Adversarial Networks

More recently, a trend of using generative models to create synthetic data has emerged. A model is generative when it directly estimates the distribution of data from training examples. Especially generative adversarial networks (GANs), introduced by Goodfellow et al. in 2014 [47], have been a popular model architecture in recent literature. GANs build upon the general concept of generative models and the success of deep discriminative models. Instead of training just a generative model, the authors propose training a generative model (the generator) and a discriminative model (the discriminator) simultaneously, in a game-theoretic setting. The discriminator is tasked with deciding whether a given data instance is real or synthesised by the generator. The generator has to generate samples and maximise the probability that the discriminator wrongly classifies these samples.

Recent work by Clark, Donahue and Simonyan presents a state-of-the-art video generating model, DVD-GAN [48]. Their model is able to generate video of unprecedented high resolution (256x256) and length (up to about 2 seconds). The videos do have some sort of temporal consistency but that is mostly on the frame-level. It is hard to identify the contents of most videos and for those videos that do seem to make sense are still absurd (see Figure 2.4 for example frames).

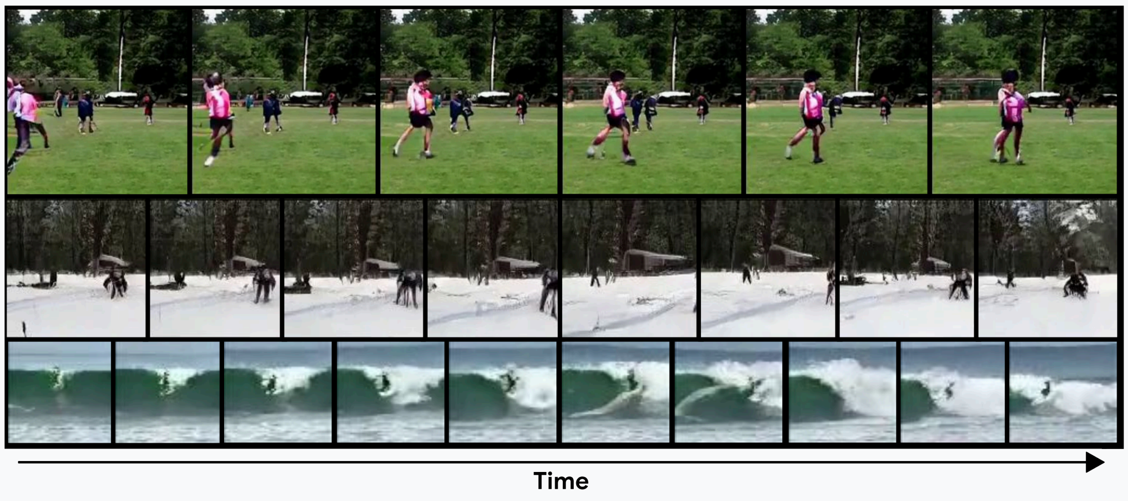


Figure 2.4: Frames from videos generated by DVD-GAN, cherry-picked by the authors. Figure from Clark, Donahue and Simonyan (2019) [48].

2.5.3. Procedural Synthesis

Although the idea of a single generative model that could take care of the entire data synthesis process is compelling, this research will not take such a direction. The current state of video generating GANs, along with the immense amount of training data and computational resources they require, make them unfit for our purposes. Most data synthesis research implements some sort of explicit and composable synthesis process that makes use of video rendering software.

Chen et al (2016) built a simple model for the generation of images with synthetic humans in randomised (but plausible) poses and with randomised clothing textures [49]. Their approach is as follows. First a statistical model was built from 3D poses recorded by a motion capture system. Samples were drawn from this model and then textured using real images of clothing, also in an automatic fashion. The textured human models were finally rendered using Blender and composed onto random background images. In total 5,099,405 images with 10,556 different human models in them were rendered. They demonstrate that the use of synthetic data created this way, aids the performance of CNN-based 3D pose estimators significantly. Without the use of complex physical simulations, but textures instead, their approach is also very scalable.

Varol et al have shown the potential of synthetic data for the recognition of humans from video [15]. The SURREAL (Synthetic hUmans foR REAL tasks) dataset that they constructed, contains videos of synthesised humans composited onto 2D images. The synthesised humans are textured human models that perform actions that were recorded with a motion capture suit before. The 2D background images are stills of various scenes and seem to just be there as some sort of noise, as the synthesised humans are not placed into the scenes in a way that makes spatial sense (see Figure 2.5). It is also shown in the paper by Varol et al that is possible to train models on this synthetic dataset that can accurately estimate human depth and semantic part segmentation.

Doersch and Zisserman, in their work from 2019 [42], show the full potential

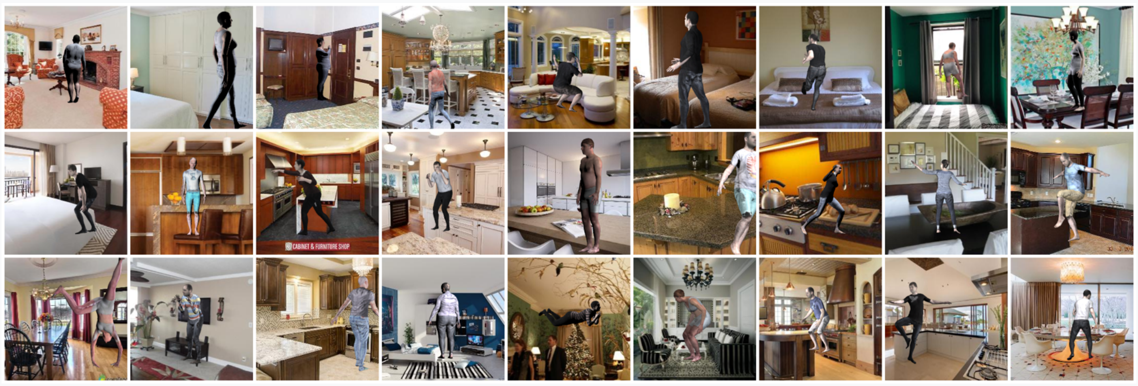


Figure 2.5: Frames from videos in the SURREAL dataset. Figure from [15].

of this dataset. They let their model first extract motion information (optical flow and 2D joint keypoints) from videos before it executes its primary task (e.g. 3D pose estimation). Note that this is a case of domain mapping, as touched upon in the previous subsection. They find that their model is able to bridge the reality gap even better with this additional step in between. The network, trained on just the preprocessed SURREAL dataset, is even able to perform on par with state-of-the-art 3D pose estimation networks, that are all trained on real, manually annotated data.

The work by De Souza et al (2017) aims to solve a problem that is very similar to the problem stated in our work [50]. They propose a procedural generation process for videos ”to train deep action recognition networks” on. The Unity 3D modelling and rendering environment with the Puppet Master plugin by RootMotion are used to render the scenes. This commercial plugin offers some realistic looking human models with preprogrammed behaviours and ragdoll physics. The authors use actions from the CMU MOCAP database [51] and hand-designed actions bought on the Unity Asset Store. A hand-designed, and thus pretty limited, virtual world is used as the environment in which the human models are playing out the actions. The virtual camera levitates behind and above the main character, following the movements of the model. The Procedural Human Action Videos (PHAV) dataset that they produce with this method contains 39,982 videos with over 1000 examples for each of the 35 action categories. A limitation of this work with regard to rare activity recognition is that it uses the CMU MOCAP database as the origin of all motions. This dataset does not contain many rare activities.

Mason, Vejdan and Grijalva (2019) [52] discuss methods to generate synthetic data ”on the fly”, meaning that it gets synthesised when it is needed, at training time. The largest advantage of such an approach is that data does not need to be stored on disk between synthesising and training. The data synthesis parameters (such as the specific viewpoint or action) can also be more easily tweaked to the specific needs of the current problem, as it is an active part of the training process.

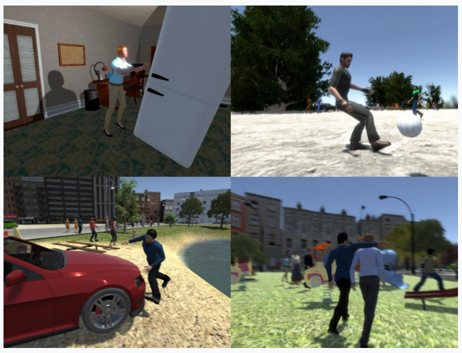
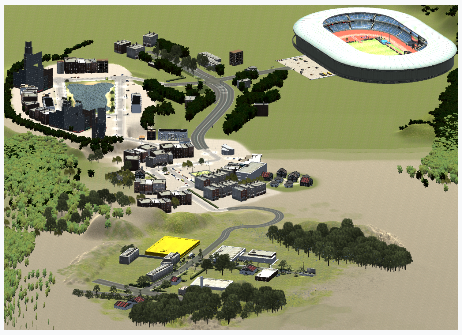



Figure 2.6: Left are frames from the generated videos with actions pushing, kicking a ball, hugging while walking and getting hit by a car. Right is an aerial overview of the entire hand-designed virtual world used by De Souza et al to place the models in. Both figures from De Souza et al [50].

2.5.4. Human Models

To generate photorealistic imagery of humans, renderable, 3D models of human bodies are required. Most research that implements a synthetic human generation solution, uses some sort of parametric human body model, of which many are available [53].

SCAPE (Shape Completion and Animation of People) [54] is a pioneering model proposed by Anguelov et al in 2005. It generates a high-quality mesh of a human body given the position of 16 body keypoints. The model is learnt from real body scans and separately accounts from body shape and pose deformation. SCAPE is used in a lot of research and spawned a variety of models that are based on it.

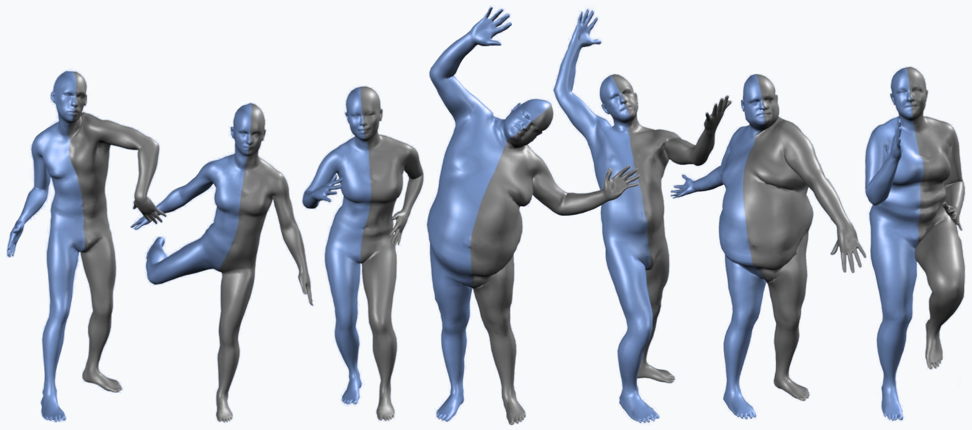


Figure 2.7: Examples of SMPL model fits to a limited number of keypoints (blue) and full ground truth poses (grey). Figure adapted from [55].

SMPL (A Skinned Multi-Person Model) [55] is a more recent (2015) body model that is also learnt from body scans. The model is fit to the desired shape and pose by tweaking a 10 shape parameters and 23 pose parameters. It is significantly faster to run than SCAPE and fits are also closer to ground-truth than the SCAPE fits [53]. SMPL is freely available for research purposes and is the most popular parametric body model in recent literature.

2.6. Pose Estimation

The research area of pose estimation tries to accurately extract the position and stance of one or multiple persons from imagery. This is done by estimating the (relative) position of a number of keypoints or human body joints. Because multiple poses in sequence compose into activities, pose estimation allows human activities to be extracted from existing video material. Pose estimation could thus supply us with a larger amount of motion data than current motion-capture-based datasets. If we curate the source video material for the wanted motion, we could end up with a motion dataset through pose estimation.

The pose estimation technique needs to meet certain requirements to achieve this goal. First of all, it needs to estimate three-dimensional poses from a two-dimensional image. This is an inherently harder problem than 2D pose estimation because a 2D pose can have several possible 3D pose explanations. The 2D image thus contains ambiguity regarding the 3D pose. The estimated poses also need to be temporally coherent, meaning that it must be physically plausible for an estimated pose to follow the pose estimated for the previous frame. Lastly, the technique should work on imagery from a single viewpoint. Performance capture e.g. for virtual character animation is a common application of certain pose estimation techniques and for these purposes a studio setup with multiple camera viewpoints is workable. For the extraction of motion from single viewpoint videos, a monocular pose estimation technique thus is required.

2.6.1. Impact of Deep Learning

Since half a decade, all new and well-performing pose estimation methods are based on deep learning techniques. This began with Google's DeepPose, by Toshev and Szegedy (2014) [56], which first proposed to use deep convolutional neural networks for this task. DeepPose achieved state-of-the-art performance on several pose datasets, including the Leeds Sports Pose (LSP) dataset with 2,000 images [57], the Frames labelled In Cinema (FLIC) dataset with 5,003 images [58] and the MPII Human Pose dataset with around 25,000 images of over 40,000 people [59]. Chronologically, improvements include Stacked Hourglass Networks by Newell, Yang and Deng (2016) [60] and Convolutional Pose Machines by Wei et al (2016) [61], both of which improved upon the state-of-the-art with innovative architectures. DeepCut by Pishchulin et al (2016) [62] also set a new best performance while also able to estimate the poses of multiple people in a single image. OpenPose by Cao et al (2018) [63] is a pose estimator with a very good and performant open-source implementation. This implementation is widely used in academia and industry. DensePose by Güler et al (2018) [64] estimates dense correspondances between 2D input images and a 3D human body. This means that every human body pixel in the image is mapped to a coordinate of a human model. Research on pose estimation in crowds is also popular, see for example Golda et al (2019) [65]. This is challenging because of the huge amount of people to track and the many occlusions.

All of the methods mentioned above estimate pose or pose keypoints in two dimensions.

2.6.2. 3D Pose Estimation

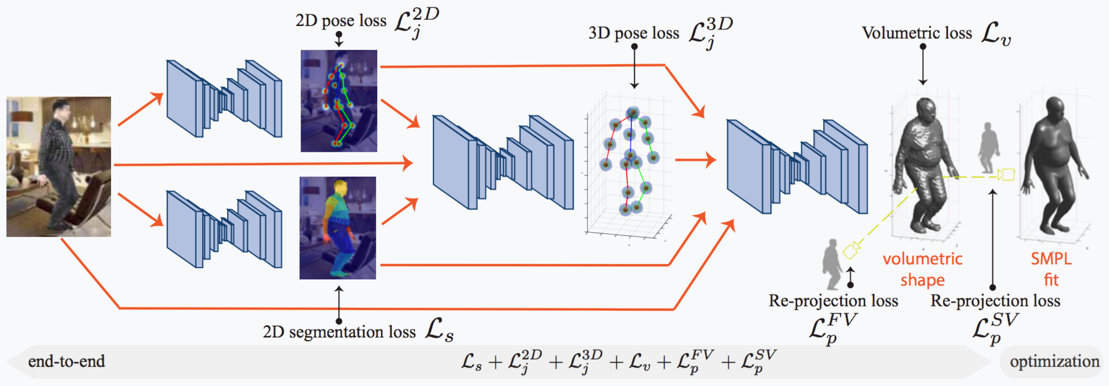
To increase the randomisation options when synthesising data from the extracted motion, 3D pose information needs to be available. With this information available, accurate 3D human models that allow for randomisation of body shape can be used. Three-dimensional information also allows for a modification of camera perspective, rendering the activity from different viewing angles. However, extracting 3D poses from 2D images is significantly harder than extracting 2D poses. This is mainly because of the larger size of 3D pose space and the possible 3D ambiguity of a 2D pose.

There are several common evaluation datasets for 3D pose estimation. The HumanEva-I dataset [66] contains 7 videos of 4 subjects performing 6 actions with ground-truth 3D poses. The Human3.6M dataset [17] aims to significantly extend existing datasets and contains lots of pose-annotated videos (3.6M refers to the number of video frames). However, these videos are all still in a laboratory setting, limiting its utility for estimation in-the-wild. The authors also extend the dataset to *mixed reality* by rendering 3D models on video backgrounds, to address this limitation. However, the resulting dataset is limited in size and the movements are still limited by the laboratory setting in which they were recorded. The Max Planck Institut Informatik 3D Human Pose dataset (MPI-INF-3DHP) [67] offers more diversity in poses, human appearance, clothing, occlusion, viewpoints and environments. This makes it a better dataset for in-the-wild 3D pose estimation (i.e. pose estimation outside of a controlled environment). The 3D Poses in the Wild dataset (3DPW) [68] contains 60 annotated video sequences recorded with a moving camera in various environments. This makes it the biggest and most challenging dataset for the evaluation of in-the-wild pose estimation methods.

The first fully deep learning-based approach to 3D human pose estimation was proposed by Li and Chan in 2014 [69]. Their model regressed from the input image to the 3D pose directly.

Instead of direct estimation by regression, others found that there is a lot of information about 3D pose embedded in the 2D pose. Bogo et al (2016) [71] used DeepCut [62] to estimate 2D joints and then fitted the 3D SMPL body model [55] to match the 2D keypoints. They called this framework SMPLify and showed that it achieves state-of-the-art performance. Chen and Ramanan (2017) [72] take a different but similar approach. Using Convolutional Pose Machines [61] they first estimated 2D pose keypoints from the image. From a predefined collection of 3D poses, a large number of 2D poses were rendered (i.e. the 3D poses were 'flattened', viewed from a variety of angles). Using a Nearest Neighbour model, the estimated 2D pose could then be associated with a 3D pose by matching. This approach led to state-of-the-art results again. Estimating 3D pose via a 2D pose is also referred to as *lifting* the 2D pose to 3D.

Zhou et al (2017) [73] went back to direct regression again. They state that a lot of depth information is lost by only using an estimated 2D pose for 3D pose esti-



2D pose loss $\mathcal{L}_j^{2D}$
3D pose loss $\mathcal{L}_j^{3D}$
Volumetric loss $\mathcal{L}_v$
volumetric shape
SMPL fit
Re-projection loss $\mathcal{L}_p^{FV}$
Re-projection loss $\mathcal{L}_p^{SV}$
2D segmentation loss $\mathcal{L}_s$
end-to-end
$\mathcal{L}_s + \mathcal{L}_j^{2D} + \mathcal{L}_j^{3D} + \mathcal{L}_v + \mathcal{L}_p^{FV} + \mathcal{L}_p^{SV}$
optimization

Figure 2.8: Overview of the BodyNet architecture. Note the intermediate 2D pose, 2D segmentation and 3D pose losses that allow for tight supervision. The volumetric shape that is estimated is used to fit a SMPL model. Figure from [70].

mation. However, available datasets in 2017 were either annotated with 2D poses or were annotated with 3D poses but situated in a lab environment, making them unsuitable for *in the wild* tasks. Zhou et al introduced a weakly-supervised transfer learning solution that partially overcomes the limitations of current datasets.

Kanazawa et al (2018) [74] and others have tried to estimate 3D pose and body shape at the same time, by estimating SMPL parameters from it, with HMR (Human Mesh Recovery). This is not a direct pose regression technique but the pose can be inferred from the fitted SMPL model. Varol et al [70] have noted that the mapping between human shape and pose and the body model parameters is currently too hard to learn well. With BodyNet, they propose an alternative: a fully trainable neural network to directly estimate 3D human body shape, not just the parameters of a body model. The network is supervised intermediately by 2D pose, 2D body part segmentation and 3D pose. The final result of the network is a volumetric estimate of the human body shape and pose, see Figure 2.9 for their architecture. Varol et al use this estimate to fit a SMPL model instance, outside of the network structure.

Kolotouros et al [75] proposed SPIN, which combines the two main paradigms: lifting 2D poses and directly regressing 3D poses. SPIN first regresses a 3D fit and subsequently uses this as a starting point for the iterative process of fitting a 3D pose to 2D joints.

2.6.3. Temporal Pose Estimation

All methods discussed so far focus on pose estimation on single images. To extract actions and activities from video, however, the estimated poses need to make sense from frame to frame. As it turns out, simply applying 3D pose estimation methods to videos causes jitter in the output poses, making the extracted activities less physically plausible. There is also additional information present in a sequence of poses. Previous and next frames can remove pose ambiguity from a certain frame because the transitions must be physically plausible. There thus is a need for taking the temporal coherency between frames of a video into account in terms of motion and body shape.

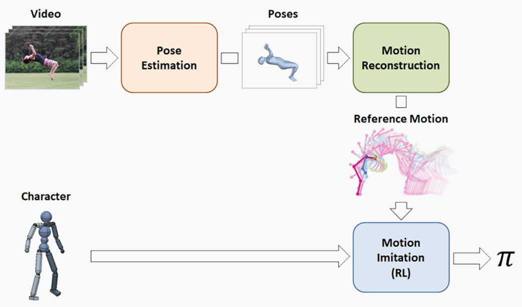
Figure 2.9: Demo of the MonoPerfCap system. Figure from [76].

A lot of work in this area has focused on a controlled setting where multiple cameras are available, for example Fusion4D by Dou et al (2016) [77]. The authors have properly dubbed this *performance capture* as this seems to be the primary goal of much research in this direction: offering a motion performance capture solution e.g. for the movie industry. Xu et al [76] and Habermann et al [78] were the first to take the leap to monocular video but still focused on performance capture very much. This means that their approaches require a 3D body scan of the actor in the video to use during the process.

Peng et al (2019) [79] have added a reinforcement learning component to solve temporal incoherencies in captured motion. They use a standard 3D pose estimation pipeline to extract raw motion from video. Reinforcement learning is used to make a simulated virtual character imitate the raw motion while being subject to a physics system. This not only smoothens the motion, as the physics do not allow the character to stutter but also allows for the environment or the character to be modified and the motion to be adapted to it in a physically plausible manner. Accuracy of the fit is traded for plausibility within the physical simulation, however. This approach is also quite computationally intensive, as is requires an entire motion policy to be learnt using reinforcement learning for every motion, on top of the usual pose estimation.

Wang et al (2019) [80] introduced *3D human pose machines* that are able to self-supervise. *Self-supervision* is a feedback signal that does not rely on manually created labels for the data. The self-supervision by Wang et al is realised by first making an initial 3D pose guess along with a direct 2D pose estimate. It is assumed that this estimated 2D pose is accurate. The 3D pose guess is then projected to a 2D pose, which is compared to the direct 2D pose estimate. Feedback from this comparison allows the 3D pose estimate to be further refined. Here the projected 2D pose is the signal and the results from the comparison is the self-supervision. The 2D-to-3D module also contains *long short-term memory* (LSTM) layers that capture the temporal dependency of the frames. A major benefit of the 3D human pose machines is their lightweight architecture, allowing them to run near real-time. See Figure 2.11 for an overview of their architecture.

In research by Pavvlo et al (2019) [81], a one-dimensional dilated convolution over 2D pose estimations for all frames is used to catch the temporal aspect. This means that first for each frame a 2D pose is estimated and then 2D poses for multiple frames are combined by the dilated convolutional layer to make a 3D pose estimation. See Figure 2.12 for a schematic view of the architecture. The authors report competitive performance both in terms of accuracy (state-of-the-



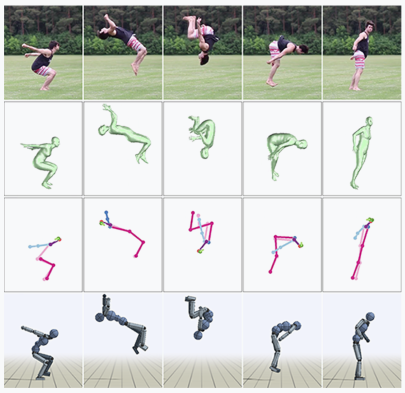


Figure 2.10: Left: overview of the SFV framework by Peng et al (2019) [79]. A pose estimator extracts motion from video and a simulated character is learnt to imitate the motion using reinforcement learning. Right: comparison between motions generated by different stages of their pipeline. From top to bottom: the input footage, estimated 3D poses, estimated 2D poses and imitated motion. Both figures from Peng et al.

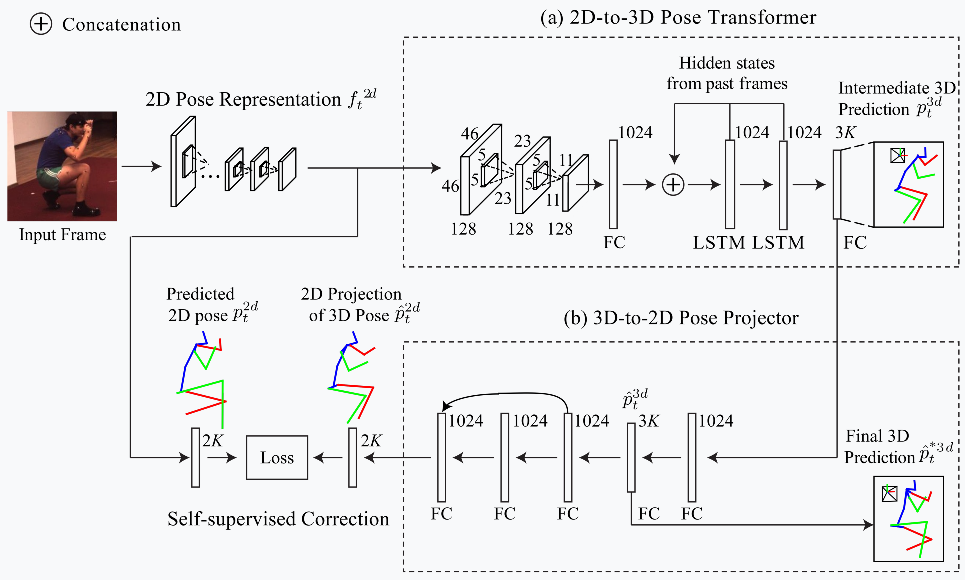
art on several tasks) and cost-efficiency. All video frames do need to be available beforehand, which is a limitation for real-time purposes but not ours.

A similar method is proposed by Arnab, Doersch and Zisserman (2019) and is based on *bundle-adjustment*, the problem of reconstructing a 3D scene from multiple 2D views of that scene [21]. The authors argue that the overall body shape of a person does not change throughout the video and multiple views throughout the video can provide useful information about this shape. Their method initially also estimates the 2D pose for each frame. After this, all estimates are taken together, at once, by the bundle-adjustment component and an optimised estimation is formed.

Human Mesh and Motion Recovery (HMMR), as proposed by Kanazawa et al (2019) [82], directly estimates a temporally coherent 3D pose sequence from 2D images. A window of frames is compressed into a representation of human dynamics, which is used to predict the current poses as well as past and future changes to the pose. As usual, 3D ground truth is used to calculate loss during training. HMMR can also train using a 2D reprojection error if 3D ground truth is not available (but 2D is).

The VIBE method by Kocabas, Athanasiou and Black [83] uses a simple CNN plus recurrent neural network architecture to generate 3D meshes. They trained this generator network with an adversarial *motion discriminator*, in a GAN-like manner. This discriminator has access to a dataset of motion capture data to determine whether a motion is genuine. The large dataset of motion capture data is an attempt to capture the space of possible human motions. This seems to be an essential part as the rest of the VIBE architecture is very simple.

All beforementioned temporally coherent methods take just a single pose into account. This can be partially solved by cropping the video around a single person and recombining the pose estimates afterwards. Kocabas, Athanasiou and Black [83] have taken this approach for their demo of VIBE and are thus able to estimate the 3D pose of multiple people in a single video. However, they do not estimate the 3D spatial relation between each of these people, just the scale and translation as seen from the original camera viewpoint. This makes it impossible to construct a



Concatenation
(a) 2D-to-3D Pose Transformer
2D Pose Representation f_t^{2d}
Input Frame
Hidden states from past frames
Intermediate 3D Prediction p_t^{3d}
46
46
5
5
23
23
11
11
128
128
128
1024
1024
1024
3K
FC
LSTM LSTM
FC
Predicted 2D pose p_t^{2d}
2D Projection of 3D Pose $\hat{p}_t^{2d}$
(b) 3D-to-2D Pose Projector
$\hat{p}_t^{3d}$
2K
Loss
2K
1024
1024
1024
3K
1024
Final 3D Prediction $\hat{p}_t^{*3d}$
FC
FC
FC
FC
FC
Self-supervised Correction

Figure 2.11: Overview of the 3D human pose machine architecture. Note the comparison of 2D poses for self-supervision and the two LSTM layers for temporal dependencies. Figure from Wang et al [80].

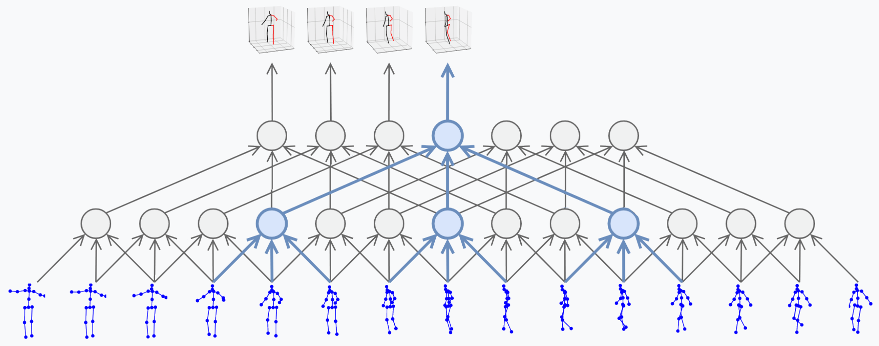


Figure 2.12: Schematic view of the dilated temporal convolution of Pavllo et al. By taking past, current and future 2D pose estimates into account, long term dependencies are taken into account. Figure from Pavllo et al [81].

correct and complete 3D scene from the estimated poses.

All beforementioned methods also cannot estimate the spatial relations between people. Most methods estimate scale and translation parameters for a *weak-perspective camera*. This allows a rendered 3D mesh to be scaled and translated in such a way that it overlays the person in the original video but does not allow for accurate 3D scene reconstruction.

Mehta et al (2019) [84] propose a method that is able to estimate multiple temporally coherent 3D poses from a video and localise these estimates in 3D space. They do so by showing a checkerboard at the beginning of the video to calibrate the camera parameters, exploiting any visible ground plane in the video and estimating a fixed height for each person when they first appear in frame. This does require the camera to be stationary. At the time of writing, the paper by Mehta et al was not yet published and was still undergoing changes. The authors

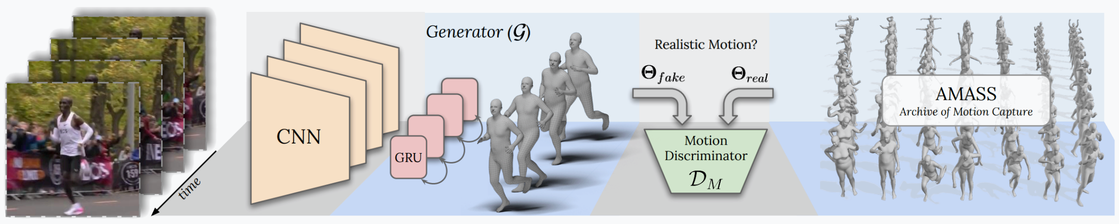


Figure 2.13: Schematic view of the VIBE architecture. SMPL parameters are estimated per frame of the input video using a network with a temporal compontent. The generator is trained in an adversarial manner against a motion discriminator that has access to a motion capture archive. Figure from Kocabas, Athanasiou and Black [83].

did not want to discuss further details or share code before publication so an exact reproduction of their method was not possible.

3. Solution Objectives

With the exploration of existing literature taken into account, the following objectives to guide the design process and to evaluate its result can now be defined. These objectives are to be met by the data synthesis solution to be evaluated on the research questions.

3.1. Input/Output

The pipeline should be able to take monocular video files as well as motion capture files as input. This objective finds its origin in the different use cases of the pipeline. For Oddity.ai, the input should be videos of violence. Copyright-free videos of violence are abundantly present on the internet, while motion-captured violence data is rare. The pipeline should be able to leverage the motion present in such videos. For the Dutch National Police, however, the input could be direct motion capture data of people walking, to train a gait recognition model on. These are easy to find in existing motion capture databases and easy to manipulate. This eliminates the need to the extract motion from monocular video, a process that introduces uncertainty about the motion data. Performance-wise this is also beneficial, of course, as no motion extraction needs to take place.

The input files should be accompanied by their respective target labels. For the violence detection use case, this denotes for each video whether it contained violence or not. For the gait recognition use case, each person should be tagged with a unique label. The pipeline should take this annotation into account and should label synthetic videos with the right class. It should be possible to create synthetic videos in which just a single class of motion is present or in which motions from multiple classes are combined.

The pipeline output should be a set of videos. The resolution of these videos should be a parameter of the pipeline. References must exist from the synthetic output videos to the source motion and the corresponding target label. With these references in place, an annotated dataset is formed of which each video can be traced back to its origin motions.

3.2. Performance

Since the pipeline will be based on deep learning techniques, a graphics processing unit is required to be present in the host system. To keep the barrier to entry as low as possible and to promote usability and reproducibility, it should be feasible to generate a dataset using the pipeline on a consumer-grade GPU.

The pipeline should be able to generate at least several hours of novel synthetic video at a minimum resolution of 640x360 pixels in less than 48 hours. This is based on the fact that most popular video datasets for activity recognition contain at least several hours worth of footage. Take for example the UCF101 dataset [85] with about 27 hours in total, Hollywood2 [86] with 20 hours, KTH Human Activity Recognition [87] with almost 3 hours. More recent, commonly used datasets, such as Kinetics-700 [12], Sports1M [88] and HACS [89] contain in the order of hundreds of hours of video.

3.3. Accessibility and Reproducibility

To ensure accessibility and reproducibility, the pipeline should work with as much publicly available research results and software as possible. This means that the reference implementation as offered by this research should use off-the-shelf components with licences that allow anyone to use them for research purposes. These components include a machine learning framework, a rendering engine and a 3D human body model.

4. Design and Development

This chapter describes the overall design of the pipeline, goes into detail on all components and motivates design choices.

4.1. Overview

At the highest abstraction level, the pipeline consists of two phases. These are the *motion extractor* phase and the *video synthesiser* phase. During the motion extractor phase videos are processed by a pose estimator to estimate human motions that will be used in the synthetic videos. During the video synthesiser phase extracted motions are combined into scenes that are rendered to videos. These phases can be ran through standalone, to just extract motion or just synthesise videos from motion (pre-extracted or pre-existing, such as motion-captured data), or in sequence, to first extract motion from video and then synthesise new video containing those motions. See Figure 4.1 for a schematic overview.

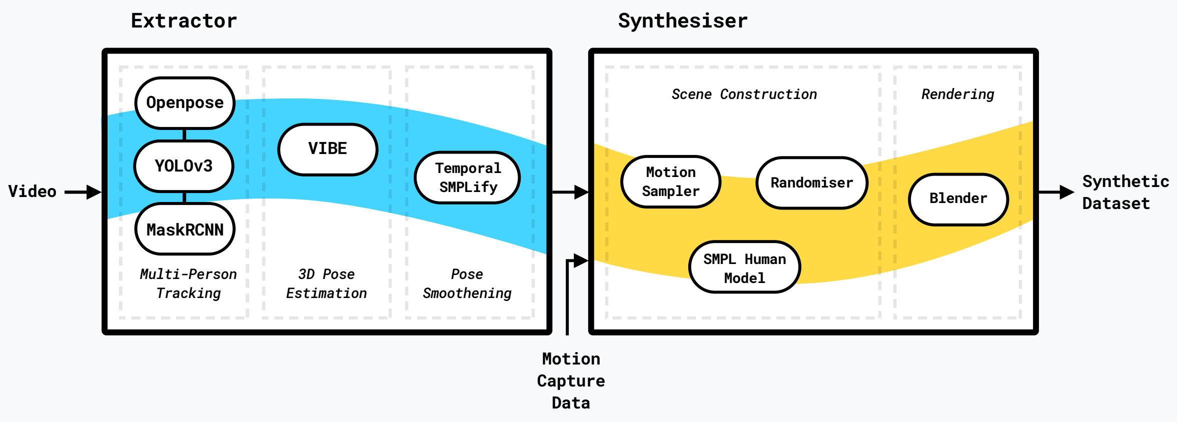


Figure 4.1: Schematic overview of the synthetic data pipeline. The *motion extractor* and *video synthesiser* phases are visible in respectively blue and yellow. Within each phase, subphases and important components are visible.

The Python implementation of the pipeline can be found in a private repo[1] to which access is granted upon request.

[1] https://github.com/oddity-ai/SYDAMO

4.2. Motion Extractor

The selection of a pose estimation technique is guided by the literature study in Section 2.6. A major requirement is for the cost-effectiveness of the technique to be reasonable, i.e. finding the right balance between motion plausibility and estimation speed. Plausible motions are of course preferred but the performance objective (in terms of speed) as stated in the previous chapter, should also be met.

Reinforcement learning-based techniques, such as those of Peng et al (2019) [79] are at the motion plausibility end of this tradeoff but are too slow for our speed objective. Peng et al use an ensemble of pose estimators and train a reinforcement learning policy per motion, making their method orders of magnitude slower. Of the other methods we report on, VIBE, being the most recent one, is chosen as its authors report improvements over some earlier methods. It is quick, accurate and accounts for temporal consistency. An additional benefit of VIBE is that it directly regresses towards pose and shape parameters for the SMPL human body model (see Subsection 2.5.4 for more details on SMPL). This means that no additional step is required to get from estimated 3D pose to a human body mesh.

As mentioned in the literature study, a limitation of all 3D pose estimation methods that are described, is that they fit the pose of just a single person. The authors of VIBE implemented a demo in which they use a multi-person tracker to crop the video into parts that all contain just a single person. These video parts are then fed to VIBE and 3D poses are estimated. We adopt this method in our pipeline too. Specifically, YOLOv3 [90], MaskR-CNN [91] and Openpose [63] are supported to find crops in the video. YOLOv3 and MaskR-CNN both are widely used object detectors. Both detect multiple classes of objects but our pipeline only uses the *human* class detections as the basis for crops. Openpose is an open-source pose estimation library that can estimate 2D poses of multiple people at once. VIBE optionally uses the extracted 2D poses to further optimise the estimated SMPL parameters. This is done by extending the SMPLify method of Bogo et al (2016) [71], as touched upon in Section 2.6. *Temporal SMPLify*, as the authors call it, also optimises the fitted SMPL model using constraints for a smooth pose and consistent shape over time. However, insignificant improvement of pose plausibility when using Temporal SMPLify is reported.

VIBE's demo implementation also provides a renderer that shows the estimated poses over the original video. However, the poses are first rendered individually and then translated and scaled back onto the original video using the crop dimensions as found using YOLOv3, MaskR-CNN or Openpose. This leads to a convincing render of multiple poses on the 2D plane of the original video. But actually, no real spatial information on how different poses relate to each other is known. This cannot be expected since the input to VIBE is just the single person crop, so no real understanding of the entire scene can be formed.

To guarantee a constant framerate for the extracted motions from different source videos, each source video is first converted to have a framerate of 24 frames per second. All motions that are directly obtained from motion capture suits through AMASS are also converted to 24 frames per second by subsampling. Typical motion capture framerates are high enough to allow for accurate interpolation to 24 frames per second. For example, all recordings in the CMU MOCAP

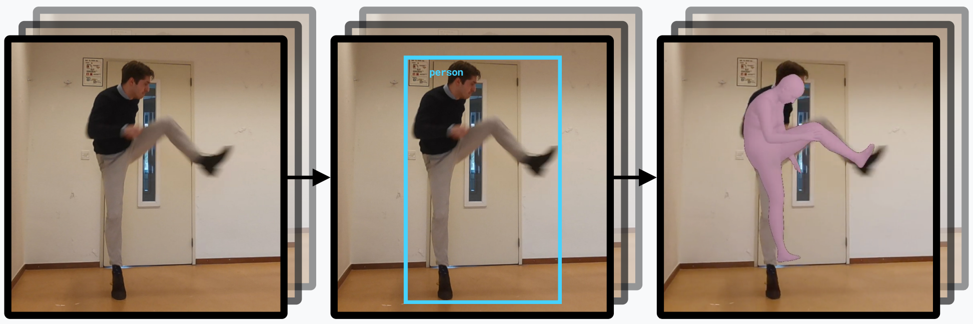


Figure 4.2: An example of the two first steps in the motion extraction process with YOLOv3 or MaskR-CNN. First, all people in the video are detected and crops are made. Then, for each of these crops a 3D mesh is fitted using VIBE.

database are captured at 120 Hz, which subsamples perfectly to 24 frames per second.

4.3. Video Synthesiser

4.3.1. Modelling and Rendering Environment

As seen in Subsection 2.5.3, procedural synthesis approaches currently have major benefits over alternative methods such as the use of GANs to generate video (see Subsection 2.5.2). To implement such explicit and procedural video synthesis, 3D modelling and video rendering software is required. The most important capabilities are the composition of scenes in 3D, including the manipulation of a 3D human model, the animation of the human models in those scenes and the rendering of a video of the scene. To ensure the accessibility objective of the pipeline, all software must also be free for anyone to use.

After looking into Cinema4D [92], Unity [93], Unreal Engine [94], 3ds Max [95] and Blender [96], the latter turned out to be the best option for this research. First of all, Blender integrates 3D modelling and video rendering into a single piece of software. Blender also lends itself very well to scripting. Its user interface is built on top of a Python API that is entirely available to the user [2]. All Blender functionality can be used via this API and a Python script. Another major advantage of Blender is its open-source nature, which lowers the barrier to entry enormously. This could also be a main reason why Blender is the most widely used 3D software in related research, judging from our literature study (see Subsection 2.5.3).

[2]https://docs.blender.org/api/current

4.3.2. Motion Sampling

Each synthetic video that is to be generated, is in essence a subset of the (extracted) motions. How these subsets should be obtained from the set of all available motions depends on the use case of the synthetic dataset.

For the use case of gait recognition, for example, only one motion should be present at a time. For the use case of violence detection, multiple motions could be present and the subset size does not always have to be the same. The maximum size of the motion subset thus is made configurable. For each synthetic video, a subset size is sampled from a uniform distribution $U(1, m)$ where m is the configurable maximum number of motions per video.

Each motion file is annotated with a class. When $m > 1$, motions of different classes could be present in the synthetic video's motion subset. It depends on the use case of the synthetic dataset whether this is desired or not. How the synthetic video should be annotated, can also differ from use case to use case. This motivates a generalisation to three *class combination strategies* of which one can be used at a time:

1. **Pure**
   Only motions of the same class are selected and combined into a scene. The synthetic video naturally is annotated with that class.

2. **Mixed**
   Motions are combined regardless of their class. The synthetic video is annotated with all classes that are present in the scene.

3. **Dominant**
   Motions are combined regardless of their class. The synthetic video is annotated with the class name of the dominant class, if it is present. Otherwise it gets annotated with all classes that are present in the scene. Note that this can also be achieved with the mix strategy and postprocessing of the annotations.

Consider our use cases again. For gait recognition $m = 1$ so these strategies do not cause different output. For violence detection the *dominant* strategy will be used for the *violence* class: when a violent motion is included, the entire video should be marked violent. These strategies cover alternative use cases as well since the *mixed* strategy can easily be combined with a mapping from present classes to the desired class annotation. E.g. classes *throw punch* and *being punched* might both be present in a given video and this could be annotated with *punch* using a simple association mapping.

4.3.3. Body Model

Based on the literature study on body models in Subsection 2.5.4, the SMPL body model is the most obvious choice because of its speed, physical plausibility and popularity in literature. With VIBE as the preferred backbone of our motion extractor, this choice is confirmed, as VIBE directly estimates a SMPL model fit.

The SMPL model is parametrised by Θ, which consists of θ for the pose and β for the shape. Specifically, $\theta \in \mathbb{R}^{72}$ and each triplet is a rotation vector in the axis-angle representation. The first rotation vector indicates the global rotation of the body. The remaining 23 rotation vectors denote the rotation of each of the 23 joints. These do not include joints in hands and feet, which are thus always in the same pose. $\beta \in \mathbb{R}^{10}$ for the 10 shape coefficients that were obtained from body scan data. Unlike the 23 joints, which are manually determined and anatomically grounded, the shape coefficients do not have an intuitive meaning because of their origins from data.

AXIS-ANGLE REPRESENTATION

A common representation of rotation in 3D Euclidian space. Given a unit vector (vector of length 1) e that indicates the direction of the axis of rotation and θ that indicates the amount of rotation, the *axis-angle representation* equals θe. Sometimes this is also written as a vector of length 4 where the first 3 items are e and the last item is θ.

On the project website of SMPL[3] the 3D model is provided in the FBX format, which is a widely used file format for storing 3D models, and which is supported by Blender [97]. The 3D model includes all SMPL joints organised in a *kinematic tree*. This means that the rotation of joints higher up the tree cause lower joints to also move along. For example, rotating the right shoulder joint causes the elbow joint to change location.

With Blender, this body model can be painted using texture image maps and a shader. The texture image map is an image whose pixels mapped to certain regions of the body model. We build upon SURREAL [15] and use their texture maps to paint our body models. These include maps that are extracted from the CAESAR body scans as well as new body scans. The faces in these textures have all been anonymised by replacing them with the average CAESAR face and then correcting the colour for the skin colour according to the rest of the body. In total there are 930 body textures that our pipeline will uniformly sample from.

4.3.4. Animation

The extracted motions contain SMPL pose and shape parameters for each time step. This determines the maximum length of the animation. The SMPL FBX model has function B_S that maps shape coefficients β to a deformation of the model mesh built-in. This makes it straightforward to apply body shapes from the (extracted) motion files. The pose parameters are converted from axis-angle representation to full rotation matrices using Rodrigues' formula. These rotation

[3] `https://smpl.is.tue.mpg.de/downloads`

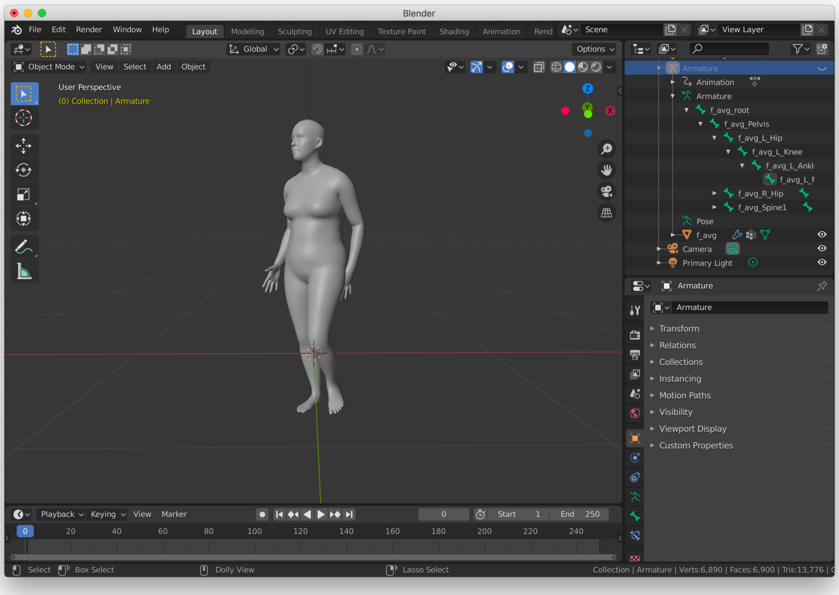

Blender
File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rend
Scene
View Layer
Object Mode View Select Add Object
User Perspective
(0) Collection | Armature
Armature
Animation
Armature
f_avg_root
f_avg_Pelvis
f_avg_L_Hip
f_avg_L_Knee
f_avg_L_Ankl
f_avg_L_F
f_avg_R_Hip
f_avg_Spine1
Pose
f_avg
Camera
Primary Light
Transform
Relations
Collections
Instancing
Motion Paths
Visibility
Viewport Display
Custom Properties
Playback Keying View Marker
Start 1 End 250
Select Box Select Dolly View Lasso Select
Collection | Armature | Verts:6,890 | Faces:6,900 | Tris:13,776

Figure 4.3: The SMPL body model imported in Blender. The top-right pane shows the kinematic tree of model joints.

matrices are then applied to each joint in the SMPL model and keyframes are set in Blender to create an animation.

RODRIGUES' FORMULA

Formula (named after Olinde Rodrigues) to rotate any vector $\boldsymbol{v}$ using a rotation in axis-angle representation.

$$\boldsymbol{v}_{\text{rot}} = \boldsymbol{v}\cos\theta + (\boldsymbol{e}\times\boldsymbol{v})\sin\theta + \boldsymbol{e}\,(\boldsymbol{e}\cdot\boldsymbol{v})(1-\cos\theta)\,,$$

where $\boldsymbol{v}$ is a vector in $\mathbb{R}^3$, $\boldsymbol{e}$ is a unit vector that indicates the direction of the axis of rotation and θ is the amount of rotation.

In this phase, it might also be useful to apply a smoothing algorithm to the pose data. Motion capture data can be noisy and the extracted motions from VIBE also contain data distortions. We resort to the findings of Poppe, Van Der Zee, Heylen and Taylor (2014) [98] and optionally apply a moving median filter to the rotation matrices. As suggested by Poppe et al, we implemented this filter with a modest window size of 0.25 seconds.

Applying the right translation, along with the animation of the joint rotations as mentioned above, would make the animation convincingly realistic. This makes intuitive sense. For example, 'jumping' expresses not only in explosively stretching the knee joints but also in moving through space vertically.

A major limitation of the motion extraction phase, however, is the lack of information about the location of the persons in the scene's 3D space. When multiple motions are extracted from a single video, it is not known how these relate to each other spatially because there is no real 3D understanding of the scene in the source video. Per individual person a 2D translation and scale, in terms of the dimensions of the source video from which the motion is extracted, is estimated by VIBE. This can be used to project the estimated 3D mesh back onto that original source video such that is overlaps the source person. Though the 2D translation and scale do convey some limited amount of information about the 3D translation (the scale could be interpreted as the missing depth axis), more information is required for a real and accurate 3D translation animation. This means that the translational motion information is not captured correctly. The earlier example of 'jumping' thus cannot be accurately implemented as there is no access to the vertical translation details.

However, a complete lack of translation is undesirable as it is unnatural for all actors in a scene to not move around. For this reason we made a few assumptions about movement and placement of the models to allow the algorithmic generation of *random walks*, which will be used for the translation animations.

Movement and Placement Assumption 1: We assume all models can be permanently placed on the same ground plane without invalidating the motions (i.e. without the essence of the motions as described by joint rotation being lost). We think this is reasonable because most people spend by far most of the time with at least one foot on the ground in real life.

Movement and Placement Assumption 2: We assume the models can move around the scene semi-randomly without invalidating the motions. Under certain constraints for physical plausibility, such as momentum and a range of possible speeds, we think this is also reasonable.

Movement and Placement Assumption 3: We assume a certain collision avoidance tendency of the models to prevent them from walking through each other.

The random walk generation procedure is as follows. Each model in the scene is first assigned coordinates x_0 and z_0 of an initial location on the ground plane, where $z_0 \sim U(0, 5)$ and $x_0 \sim U(-x', x')$, where x' is an approximation of the field of view of the $45°$ perspective camera as in Equation 4.1 to ensure that the models are always in view. No random jumps or falls are generated.

$$x'(z) = 0.375z + 1.6 \tag{4.1}$$

Initial horizontal speeds are sampled: $v_0^x, v_0^z \sim \mathcal{N}(0, 0.005)$. Then for each animation time step t after $t = 0$ the coordinates x_t and z_t are updated with a delta value, see Equations 4.2 and 4.3.

$$x_t = x_{t-1} + \Delta x_t \tag{4.2}$$

$$z_t = z_{t-1} + \Delta z_t \tag{4.3}$$

If a collision with another model happens after the coordinate update, it is tried again. The delta values are sampled from a normal distribution centred around the speed values with a very small standard deviation that increases with the number of tries, see Equations 4.4 and 4.5 where n is the number of tries thus far.

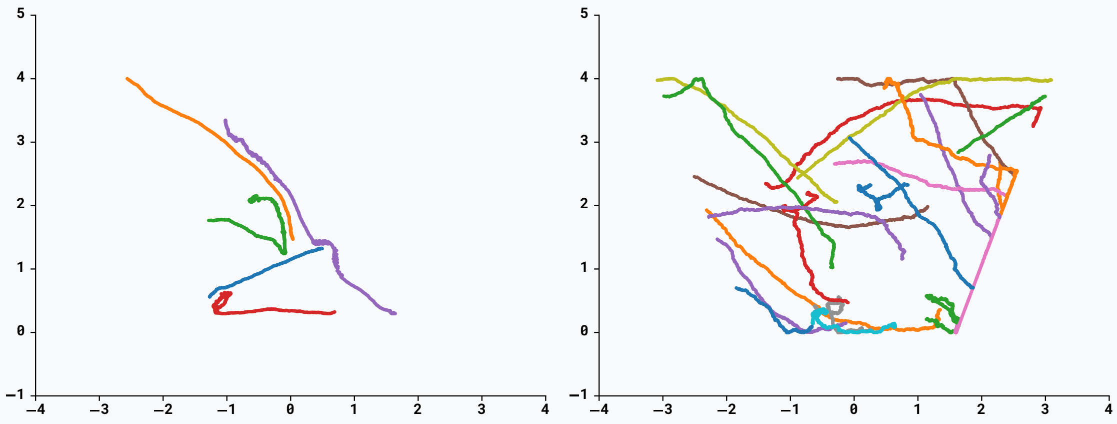


Figure 4.4: Random walks generated by our algorithm. Left: 5 random walks with collision avoidance. Right: 15 random walks without collision avoidance. Note how the camera field of view becomes visible by the limits of the random walks.

$$\Delta x_t \sim \mathcal{N}(\mu = v_{t-1}^x,\, \sigma = 0.0025 * (1 + (n \mod 10))) \tag{4.4}$$

$$\Delta z_t \sim \mathcal{N}(\mu = v_{t-1}^z,\, \sigma = 0.0025 * (1 + (n \mod 10))) \tag{4.5}$$

This way models either speed up or slow down to avoid collisions. At each time step, the speed values are also updated.

$$v_t^x \sim \mathcal{N}(v_{t-1}^x,\, 0.0005) \tag{4.6}$$

$$v_t^z \sim \mathcal{N}(v_{t-1}^z,\, 0.0005) \tag{4.7}$$

The speed values also approach 0 when the model approaches the borders of the visible part of the ground plane in order to prevent the from disappearing off-screen.

Figure 4.4 depicts some examples of random walks generated by the described algorithm.

4.3.5. Background Images

In Subsection 2.5.3 it became clear that it is not necessary to place synthetic humans in convincing 3D scenes if the purpose of the data is to train a deep learning model with a human-centred task such as activity recognition or pose estimation. This is apparent in SURREAL [15] in particular. In this work synthetic humans are rendered on top of images and the results achieved with human recognising CNNs trained on the synthetic dataset are very competitive. The point made is that realistic and diverse background images increase robustness of the trained models and that these models do not develop a spatial understanding of scenes they are presented with anyway. In SURREAL it is theorised that any effort to create convincing 3D scenes and to realistically place the human models in there, is not needed to achieve competitive performing human-centred models.

The LSUN dataset [99] is used by Varol et al for SURREAL. This dataset contains roughly 10 million images of 10 scene categories. Only the kitchen, living room, bedroom and dining room categories are used and all images with humans are removed, which leaves around 400,000 images remaining.

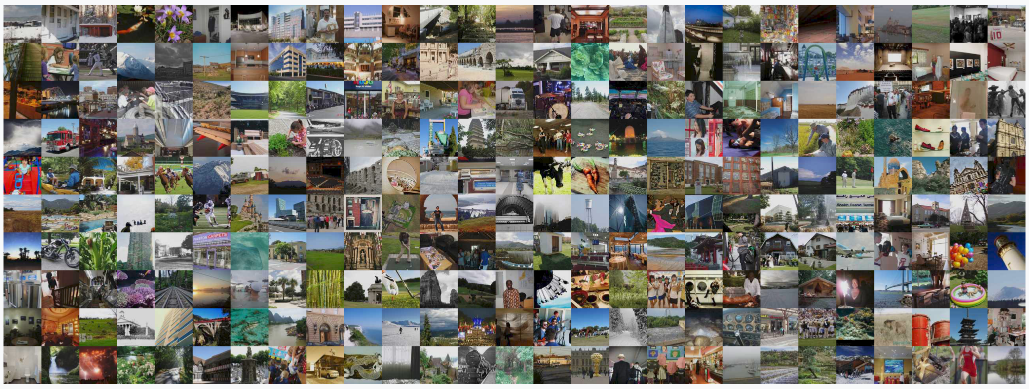


Figure 4.5: Collage of some scenes found in the Places dataset. Figure from [100].

Our pipeline is built to use any dataset of background images. For our evaluation experiments, a subset of the Places database [100] is used. This database also consists of roughly 10 million images but is more diverse with a total of 434 scene categories. We took the evaluation subset of Places, which contains 22 thousand images, to generate our evaluation videos with. All backgrounds were processed by a multi-person detector and those with people depicted were excluded from our synthesis pipeline.

4.3.6. Virtual Camera

To create video footage of the 3D scenes, a virtual perspective camera is placed in the scene. The camera is equipped with a lens that has a field of view between $45°$ and $55°$, which rougly equals a focal length of $30 - 38$ mm. This ranges from a slightly wide to a standard lens, which is common for surveillance cameras.

The camera is tilted $0°$ to $15°$ downwards, the exact angle is determined randomly for each synthetic video. This is motivated by the fact that most surveillance cameras are located on poles or are attached to buildings, pointing slightly down. When the tilt angle increases, the vertical position of the camera also increases slightly, such that all body models are always in view. The amount of variation in the camera tilt angle can be set using a pipeline parameter.

4.3.7. Scene Lighting

In real life, lighting conditions are endlessly different, depending on time and place. There are many light sources with a large variety of strengths and at a large variety of distances of the subjects in a scene. The sun is often responsible for a majority of available light but reflections of sunlight via objects in a scene also play a big part in lighting subjects. Indoors artificial sources of light also take a part in lighting. This results in an extremely complex and interconnected collection of variables that determine the exact lighting.

In our modelling environment simplifications have to be made for the sake of technical feasibility and interpretability. The first variation-introducing component is a brightness adaptation mechanism take partly takes the background image into account, as an analogy for the environmental influences in the real world. The second component is a mechanism to change the position or direction of the lighting. Two options, or *light modes*, that implement these mechanisms but differ in the number of lights, the range in which they are positioned and the range of their brightness are implemented.

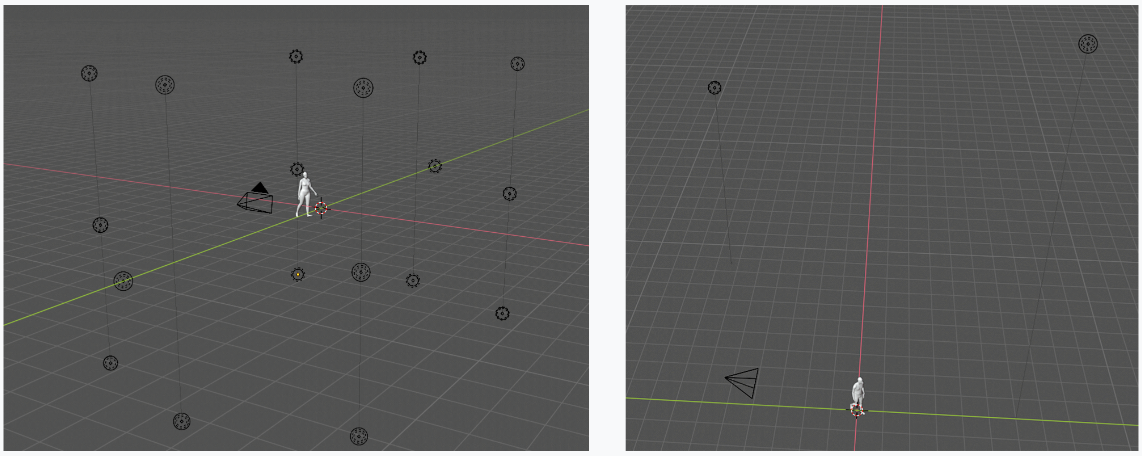


Figure 4.6: Visualisations of the two light modes. Left is the ambient light mode. Right is the direct light mode. In both the camera and body model are also visible.

The first mode, referred to as the *direct light* mode, two lights are placed high up in the air. This is motivated by the fact that most scenes are lit by a primary light source that is high up in the air, such as a lamp or the sun. The primary light in this mode has a light-emitting power of $12500 * l$ watt, where l is the average pixel value of the background image and $0 \leq l \leq 1$. This light is positioned on the same Y-axis as the camera, with $x = 0$, always just behind the human model with $z = 6$ and in the air with $y = 15$. The secondary light in this mode has a light-emitting power of $(0.75*l+0.2*r+0.05)*20000$ watt, where r is a uniformly sampled random value between 0 and 1. This can be thought of as consisting of three components: $0.75 * l$ makes sure that this light adapts to the background, $0.2 * r$ is a random variation and 0.05 makes sure that it is never totally dark. The coordinates for this light are determined as follows: $x \sim U(-12, 12)$, $z \sim U(-12, -2)$, $y \sim U(9, 19)$.

The second mode, referred to as the *ambient light* mode, is motivated by the fact that most modern surveillance cameras are equipped with *high dynamic range* (HDR) [101], [102]. HDR is a technique that combines a slightly underexposed shot and a slightly overexposed shot into a single frame to combat loss of detail that can occur when there is too much contrast (such as hard shadows caused by bright lights) in a scene. By smartly combining the most detailed regions of differently exposed shots, as much detail as possible is combined into the final image.

This process balances out the sharp light conditions and effectively lowers the contrast.

To simulate HDR in our modelling environment, two grids of 3 by 3 lights are placed before and after the human model (as seen from the camera). The grids are deformed a bit such that a semi-spherical formation of lights around the subject is formed. See Figure 4.6 for a visualisation of the lights in 3D space. All lights have a light-emitting power of $(0.5 * l + 0.5) * 500$ watt, where l again is the average pixel-brightness of the background image. These lights thus have a more constant light-emitting power than the lights in the direct light mode. Light is also coming from all directions because of their numbers and placement.

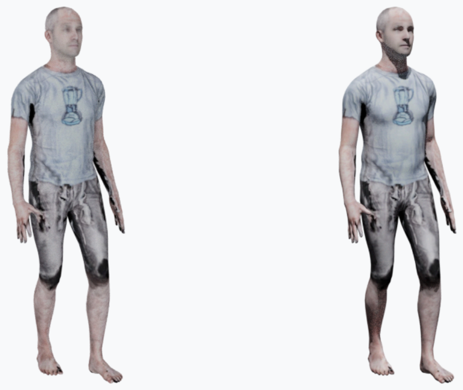


Figure 4.7: Renders of the same human model under the two different light conditions. Left is the ambient light mode, right is the direct light mode.

In Figure 4.7 the two light modes as seen from the camera's perspective can be compared. Note the increase in contrast and sharper shadows in the direct mode compared to the ambient mode. The is especially visible on the neck, legs and the area around the chest and armpit.

All lights in both modes are omnidirectional lights as offered by Blender, meaning that they shine equally bright in all directions. The difference in performance that these two light modes result in will be evaluated in Chapter 5.

5. Evaluation

In this chapter the experiments that are performed to evaluate the data synthesis pipeline are introduced and discussed. These experiments are guided by Research Questions 1-3. The importance of variance in all available parameters of the pipeline is investigated using an ablation study. In this study variance in the value of one pipeline variable at a time is brought to a minimum. The resulting synthetic data is then used to train a model and the trained models are compared to draw causal conclusions about values of the pipeline parameters. The impact that synthetic data can have on activity recognition performance is also experimentally assessed. This is done by training a model several times on datasets that differ in their composition of real and synthetic data.

As introduced in Chapter 1, gait recognition and violence detection are the two experimental cases chosen to evaluate on. Gait recognition is considered first because it is simpler than violence detection in several key aspects: just a single person is involved and motion capture data is readily available. It will function as the subject of our ablation study. Violence detection will be considered as a real-life and production-ready test of the benefits of using synthetic data for training purposes. Here we will evaluate the impact synthetic data can have on an already high-performance deep learning system.

The results and discussions of both experiments serve to answer the main research question in Chapter 6.

5.1. Gait Recognition

This section provides more details on the task of gait recognition in preparation for its use in the ablation study.

5.1.1. Objective

At the Dutch National Police, a common search task is to find a suspect in a large set of surveillance footage in order to make a reconstruction of the suspected crime. The objective of this work is to create a deep learning model that assists with this search task. The model should find a unique *gait representation* of each person visible in the footage. This should be invariant of clothing that the person wears, the lighting condition of the scene, the camera angle or the backgrounds of the scene itself. By comparing the unique gait representation of the suspect with those of the people visible on the surveillance footage, the list of all footage can be sorted. This work will be focussed on training the model using synthetic data and not the search task as a whole.

5.1.2. Model architecture

The model we propose is a hybrid between an action recognition and pose detection architecture. We refer to this model as *GaitNet*. Video clips are passed through both an action recognition component and a pose detection component. The resulting features are combined and passed through two fully-connected layers. The last of these layers is the classifying layer, which can be removed in order to get a feature representation that could be used as the gait representation. See Figure 5.1 for a conceptual overview of the proposed GaitNet architecture.

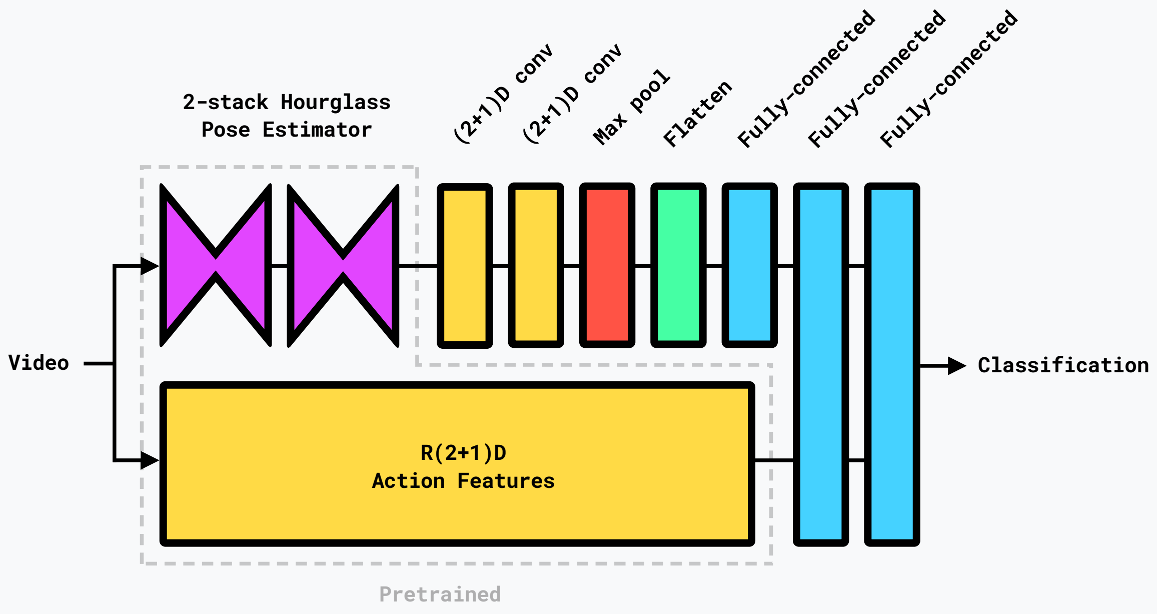


Figure 5.1: Conceptual overview of the GaitNet architecture.

As theoretic background on the architecture is not relevent for the synthetic data generation pipeline itself, it has not been discussed in Chapter 2. Therefore this will be done shortly here.

The 18-layer deep, ResNet-inspired R(2+1)D network proposed by Tran et al (2018) [34] takes place as the action recognition component in our network. This network makes use of (2+1)D convolutional units, which serve as drop-in replacements of 3D convolutional units (see Section 2.4) but decompose them into separate spatial and temporal steps. This decomposition has two advantages. Firstly, an extra non-linearity can be added between the two convolutions, increasing the complexity of the functions that can be estimated by the network. Secondly, it has been shown experimentally by Tran et al that the decomposition is beneficial for optimisation during training, leading to faster convergence. Further reasons to choose R(2+1)D are its excellent performance as reported by the authors and that fact that it takes RGB frames as input, not requiring heavy preprocessing. The instance of R(2+1)D that is used in our model has been pretrained on Kinetics-400.

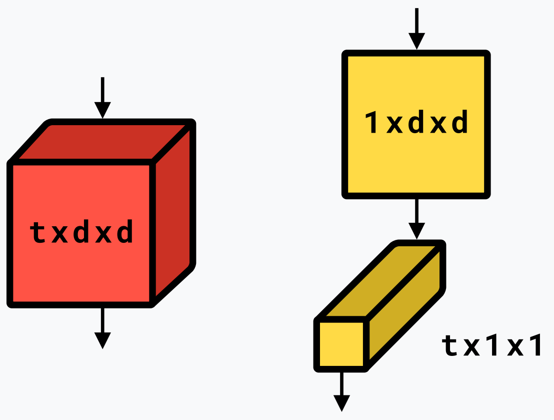


Figure 5.2: Conventional 3D convolution (a) compared to (2+1)D convolution (b). The filter size component *t* refers to the temporal dimension and *d* refers to the spatial dimensions. Figure adapted from Tran et al (2018) [34]

(2+1)D CONVOLUTION

A spatiotemporal convolutional unit that consists of a spatial convolution and a temporal convolution. The spatial convolution is implemented as a 3D convolution where the time dimension of the filter has size 1. The temporal convolution is implemented as a 3D convolution where the spatial dimensions of the filter are 1. See Figure 5.2 for a schematic comparison with the conventional 3D convolution.

The second component of our model, the pose estimator, is made up of a two-stack hourglass model as proposed by Newell, Yang and Deng (2016) [60], which is also touched upon in Section 2.6. This is a simple and performant 2D pose estimator based on convolutional layers. The convolutions map the input image down to a low dimensional space and then up again to estimation joint locations, forming a pose. Useful features still need to be extracted from theses poses and, on top of that, the stacked hourglass model estimates poses on a frame-by-frame basis. This motivates the two (2+1)D convolutions our model applies on the estimated poses over time. After this a max pooling operation and a fully-connected layer are applied.

5.1.3. Source Motion

Motion capture recordings from the CMU MOCAP database [51] are used as the source for the synthetic video generation. Each recording in the database is annotated with the action performed and the subject that performed the action. We collected recordings of the ’walking’ action for 13 subjects that had more than 1 of these recordings and 2 additional subjects with only one such recording. This results in 43 motion capture recordings of 15 distinct subjects that function as the base motion for synthetic data generation. The first 30 recordings were used to generate training data and the last 13 recordings were used to generate test data. To provide at least one recording of all 15 different subjects in both sets, the test set was complemented with two recordings from the train set. Unfortunately, the CMU database does not contain enough ’walk’ data to keep these two sets strictly disjunctive. There is thus slight overlap between the source motion data of the train set and test set, which should be taken into account when analysing model performance. See which exact recordings were used in Table 5.1.

Subject	Recordings (train)	Recordings (test)
2	1	2
5	1	1*
6	1	1*
7	1,2,3	6
8	1,2,3	6
12	1,2	3
15	1,2,9	14
32	1	2
35	1,2,3	4
38	1	2
39	1,2,3	4
74	1,2	20
83	27,28,29	30
90	22	23
114	13,14	15

Table 5.1: Overview of all CMU MOCAP recordings used and the division between recordings that were used for the train and test set. Note that recordings with an asterisk are in both the train and test set.

As the motion capture data from CMU is marker-based and our data synthesis pipeline requires SMPL pose parameters, a conversion step is required. Loper, Mahmood and Black (2014) [103] introduced MoSh, the go-to method for finding SMPL pose and shape parameters from body markers. AMASS, which stands for Archive of Motion Capture as Surface Shapes, by Mahmood et al (2019) [104] provides a unified database of many motion capture datasets which have already been converted to SMPL parameters using MoSh. We made grateful use of AMASS to get the 43 CMU MOCAP motions as SMPL pose and shape parameters.

5.1.4. Video Synthesis

The retrieved motion capture data was put through our synthetic data pipeline with base configuration settings as can be found in Table 5.2. Generating 750 videos of maximum 150 frames (less if the motion data is shorter) at a resolution of 640x360 pixels took about 5 hours using an NVIDIA Tesla K80 GPU. For the ablation study, this base configuration was adjusted slightly several times to create more datasets. Details on the tweaked parameters can be found in the next section.

Parameter name	Value
Tracking method	**YOLOv3**
Max. number of frames per video	**150**
Max. number of persons per video	**1**
Dataset size	**750**
Render height	**360**
Render width	**640**
Class combine strategy	**Pure**
Moving median smoothing	**False**
Original body shape	**True**

Table 5.2: Base configuration parameter values used for the gait recognition data generation.

5.1.5. Data Preprocessing

The synthetic videos need to go through preprocessing to make them fit as input for the GaitNet model.

The two pretrained components, the pose estimator and the R(2+1)D model, require the videos to be cropped to respectively 256x256 and 224x224 pixels. The human models move through the scene (caused by the random walk algorithm) but for the gait recognition to work they should always be visible in the square crop. To realise this, the YOLOv3 multi-object tracker is employed to find crops around on the human model. Additionally, we ran a pose estimator to find the location of the pelvis joint and centred the crops on it. This way all human models are always positioned in the center of the crops, which we think is beneficial for

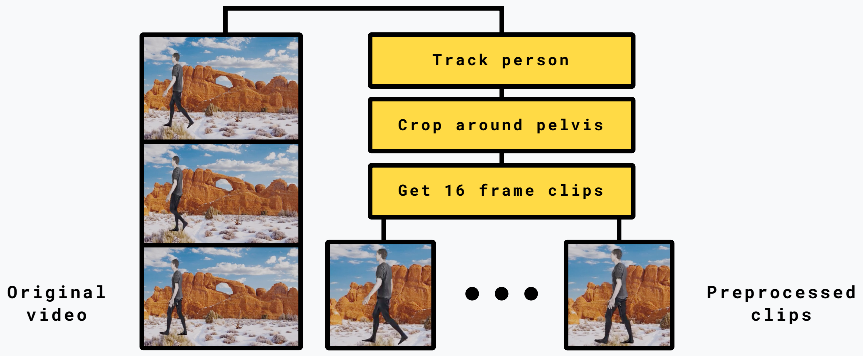


Figure 5.3: Conceptual overview of the data preprocessing done for GaitNet. Original videos as generated by our pipeline are inputted and clips of 16 frames long, tightly cropped around the subject and centred on the pelvis are the output.

GaitNet performance. The scale of the crop as estimated by YOLOv3 is smoothed using a Savitzky–Golay filter with a large window size. We save all crops with a resolution of 224x224 and upsample them again to 256x256 during the forward pass of the pose estimator component to prevent duplicate saves.

The R(2+1)D model is trained on videos of 16 frames long. Accordingly, with the last preprocessing step 16 frame clips from the full-length cropped video are extracted. Each clip overlaps the previous clip with 2 frames.

The size of generated datasets is measured in number of videos, which results in some datasets containing slightly more frames than others. Also, during the preprocessing phase the multi-object tracker does not always detect a human in the synthetic videos. These two combined result in small differences in preprocessed dataset sizes. In total the amount of 16 frame clips in the different datasets is: 5130 for the full dataset, 5191 for the background sample limit dataset, 5204 for the clothing sample limit dataset, 4912 for the direct light dataset, 5187 for the static camera angle dataset, 4922 for the random body shape dataset and 1695 for the test dataset. These deviations are relatively limited.

5.2. Ablation Study

To investigate the causal effects of single components on the results of the pipeline as a whole, we perform an ablation study. Removing just one component of our pipeline at a time and comparing the results with the baseline, where all components are present, allows for causal statements to be made about that single component. This method is especially useful when dealing with unwieldy systems such as deep learning models, which generally are expensive to train.

At several moments in the synthesis pipeline some sort of variety is introduced in the synthetic videos. It is theorised that this variety grounds the videos in reality by closing the reality gap (see Subsection 2.5.1). The amount of variety at all of these moments is controlled using pipeline parameters. These parameters, 5 in total, will be subject of the ablation study. In the following subsection, the parameters are introduced. For each parameter, it is made clear what its default value is and what value the parameter takes when it is *removed*, in terms of the ablation study on gait recognition.

A dataset is generated for each of the 5 parameter removals and one dataset is generated in which all parameters are present. The latter dataset is referred to as the *full* dataset and it acts as the baseline to compare ablations with. A test dataset with 250 videos is also generated using the novel test set of mocap data, as introduced in Subsection 5.1.3. This test dataset is generated with the same pipeline parameters as the full dataset. It is used to assess the ability of the trained model instances to generalise to novel situations.

Two model instances are also trained on subsets of the full dataset (250 videos and 500 videos). The performance of these models is compared with each other and baseline to make statements about the impact of changing the amount of synthetic training data.

This results in a total of 8 model instances that are trained and compared.

5.2.1. Parameters

Backgrounds As seen in Subsection 4.3.5, our pipeline renders synthetic humans on static background images, just as related literature does. Our hypothesis is that a large variety of diverse backgrounds increases the robustness of trained models in real-life scenarios where the backgrounds are very diverse and might sometimes be challenging. To test this hypothesis, a synthetic dataset is generated in which only 5 unique background images are used to sample from. All other datasets for gait recognition use the entire collection of backgrounds to sample from.

Clothing Consistent with other literature, our pipeline applies textures to the synthetic humans to make them appear clothed. We hypothesise that a large variety of diverse textures increases the robustness of trained models in real-life scenarios where people wear all kinds of different clothing and have a variety of skin tones. A dataset in which just a single body model texture is used and thus all human models seem to wear the same clothing and have the same skin tone, is generated to test this hypothesis. All other datasets are generated using the entire collection of textures to sample from.

Lighting Direct light conditions are common in real life, in case of bright sunshine or a single ceiling light. However, the ambient light mode looks more like the footage produced by most modern surveillance cameras and has more detail. We therefore theorise that the ambient light mode will lead to better performance. A dataset with the direct light mode is generated to assess this theory. All other datasets use the ambient light mode.

Camera tilt We think a varying camera tilt improves performance as the human models are now viewed from a variety of angles. This reflects real-life scenarios where surveillance cameras are places under different angles. To analyse this, a dataset is generated where the camera tilt angle is always $0°$. All other datasets use the range between $0°$ and $15°$ to sample camera tilt from.

Body shape By default, our pipeline randomly samples a body shape from the CEASAR dataset of body scans for each human model in the scene. The original body shape of the actor that performed the motion is thus not used. We think that this would increase performance for many use cases, such as violence detection, as models trained on this synthetic data get more invariant for body shape.

However, this is different for gait recognition. Because one does not lose or gain weight as easily as one changes clothes, body shape could still be an important indicator for identification. This is especially true for our use case at the Dutch National Police, where the investigation timeline is short and rules out any significant weight change of the persons of interest.

We generate one dataset in which body shape is randomly sampled for each human model to test this. All other datasets apply the original body shape of the actor in the CMU MOCAP recording, as estimated by MoSh during the conversion to SMPL parameters.

5.2.2. Experiment Results

All instances are trained using 5-fold cross-validation to obtain a reasonable approximation of the prediction accuracy of the model instance on the data it is trained on. With cross-validation, each fold is trained on a different $\frac{k-1}{k}$ of the data and tested against the other $\frac{1}{k}$, where k is the number of folds. By taking the mean prediction accuracy over all folds, the prediction accuracy over the entire dataset is approached while still being able to use all data for training. After cross-validation, the model is trained on the entire dataset from scratch to obtain the trained model instance that is used for testing on the full set and the test set. Training of a single GaitNet instance took about 24 hours on an NVIDIA Tesla K80 GPU, including training the 5 folds.

Prediction accuracy is measured as top 1 and top 5, denoted as T1 and T5 respectively. Top 1 represents the accuracy with which the true identity is correctly classified as most probable, and top 5 represents the accuracy with which the true identity is in top 5 most probable classifications. The top 5 accuracy is reported because it gives a coarser but more forgiving picture of the classification ability.

In Table 5.3 the prediction accuracies of all instances on the various datasets can be found and compared.

Instance	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Mean**	**Full** T1	**Full** T5	**Test** T1	**Test** T5
Full	37,52%	42,11%	35,87%	37,91%	39,77%	38,64%	-	-	7,26%	36,62%
Body	37,87%	32,99%	35,16%	41,06%	35,06%	36,43%	5,48%	31,28%	4,90%	34,93%
Camera	39,40%	36,22%	38,57%	36,93%	36,84%	37,59%	4,99%	35,25%	13,98%	46,16%
Background	28,49%	29,96%	26,59%	31,02%	26,49%	28,52%	10,16%	38,60%	9,85%	36,96%
Clothing	44,48%	43,61%	40,06%	47,74%	45,96%	44,37%	6,02%	34,95%	10,03%	37,55%
Light	34,59%	33,67%	33,10%	32,28%	34,83%	33,69%	9,61%	39,64%	3,89%	30,72%
Full-250	11,92%	17,73%	20,06%	22,45%	10,79%	16,59%	4,35%	26,38%	2,18%	28,73%
Full-500	18,51%	16,47%	12,97%	25,40%	17,52%	18,17%	2,51%	22,90%	4,19%	24,86%

Table 5.3: Prediction accuracy of all trained model instances. Columns **Fold 1-5** contain the prediction accuracy for each fold during 5-fold cross-validation. Column **Mean** contains the mean of these fold accuracies, which approaches the prediction accuracy of the instance trained on the entire dataset. Column **Full** contains the prediction accuracy of the trained model instances on the full dataset, which functions as baseline. This is omitted for the model instance trained on this same dataset, the cross-validation mean is to be used. Column **Test** contains the prediction accuracy of the trained model instances on the test dataset, which contains videos with of novel motions.

5.2.3. Experiment Discussion

Training accuracy The mean prediction accuracy of all cross-validation folds approaches the performance of an instance on the dataset it is trained on. However, because we are interested in the effect of ablations in the synthetsis process on the performance of trained models on the baseline data, comparing the cross-validation mean accuracy does not lead to very valuable insights as it concerns different datasets. Still, differences between the different instances are clearly visible.

Randomising body shape leads to a decrease in accuracy. This is expected as body shape no longer carries identification information in this dataset, thus making the identification task harder. Removing the camera tilt also leads to a slight decrease in accuracy. This is not expected as we hypothesised that removing variation here would make the task easier. However, with a difference of just 1%, this is negligible. The accuracy of the instance with less background images has a significantly lower accuracy and this is also not expected. It could be explained by the fact that there are still 5 backgrounds in the dataset, which could lead to the model falsely associating background elements with person identities. We deem this very likely. We theorise that completely removing variety in the background images would instead increase training accuracy as the background images no longer serve as distractors. The clothing instance sticks out as the only instance with a higher prediction accuracy than the full instance. Intuitively, it does make sense that taking away the clothing and skin tone distractors makes the identification task easier, as the gait features stand out more with other variables remaining constant. The instance trained on videos generated with the direct light mode has a harder time fitting and this is also expected, as the direct light mode causes visual obstacles such as hard shadows that make the task harder. On the other hand it could be theorised that the higher contrast on the humans created by the direct light mode might make it easier to separate the models from the background, making the task easier. However, the results seem to indicate the opposite.

The Full-250 and Full-500 instances both are significantly less accurate than the Full instance with 750 videos. This is expected as an increase in the amount of training data of good quality generally leads to performance improvements, especially when dealing with complex tasks such as gait recognition.

Test on full dataset Prediction accuracy on the full dataset indicates how well a trained instance performs despite the ablation in its training data. For the instance that is also trained on the full dataset, we take the cross-validation mean accuracy as the prediction accuracy on the full dataset.

Many instances have such low accuracy scores that they should be compared to the expected accuracy when guessing randomly. Take a naive model that knows the number of classes in the data. It would obtain a T1 test accuracy score of $\frac{1}{15} = 6.66$% and a T5 test accuracy score of $\frac{5}{15} = 33.33$%. This is very comparable with some of the reported scores. Only the light and background instances perform better than this T1 and T5 accuracy. The camera and clothing instances perform better than the baseline on just the T5 accuracy.

The prediction accuracy of every ablation instance against the full dataset is significantly lower than the prediction accuracies found using cross-validation. The variance in prediction accuracies is also quite high, with a standard deviation of 11,01% for the T1 accuracy. We can conclude from this that the model does not perform well with any of these ablations and that there is overfitting going on. Judging from the performance on the training, overfitting seems likely.

Comparing the different ablations shows that camera tilt, body shape and clothing limit perform worst. We could conclude that it is essential to randomise camera tilt and clothing texture in order to train models that are invariant to this. The body shape ablation instance was theorised to learn a representation that was more body shape invariant. It might as well have done this but the other instances still score higher, possibly just by using the identity information present in body shape as expected. The ablations with which the model performs best are the background limit and the direct light mode. The background instance might perform better because it is distracted less by all the different background images during training and could find better gait features that way. The data preprocessing used for GaitNet might play a role here, as the crop moves over the video, creating the illusion of a moving background. We can conclude from this that the light mode and background ablations have the least negative impact on performance. The prediction accuracies of the Full-250 and Full-500 instances surprise with the instance trained on less data actually performing better.

The differences of these instances with guessing accuracy are at most 7,32% (T1) or 12,83% (T5). These are not very large so any conclusions should be taken with a grain of salt as all results could still fall in the margin of error.

Test on test dataset Performance on the test dataset represents the ability of a trained model instance to extend to novel motions, as the videos are generated with new mocap data.

When comparing these results with the accuracies in the full column, many instances trade ranks. Some instances perform better on the full dataset than on the test dataset. These instances can be explained to be overfit on the underlying mocap data used for the training datasets and thus also for the full dataset. However, some instances perform better on test than on full. This cannot be explained as intuitively.

The background instance is the only instance that remains constant with only a slight decrease in accuracy, as would be expected when testing on novel data. For the instances trained on less training data there also is no surprise in the accuracies.

Just as with the test on the full dataset, not all accuracies are above the score a naive random guesser would obtain. Because the mocap data behind the test dataset is perfectly balanced, this score is 6,66%. Only the full, camera, background and clothing instances are higher than this. One could argue that this baseline should compensate for the mocap data overlap between the test data and train data. This would result in a baseline accuracy score of 7,69%, if the model could always recognise the two known underlying mocap motions through the synthetic video. All beforementioned instances still perform better, except for the full instance that is comparable to this baseline.

See Figure 5.4 for the confusion matrices of all 6 ablation instances tested on the testset. Seeing the unbalanced predictions, it is immediately clear that the models are overfit. The models all predict just a few subjects and ignore all others. Which subjects they predict, does differ from model to model.

5.3. Violence Detection

Violence detection is taken as a real-life and production-ready test of the benefits of using synthetic data for training purposes. This section provides details on the task of violence detection and the approach to add synthetic data to the pipeline.

5.3.1. Objective

The main objective is to improve the detection performance of the violence detection pipeline by complementing the dataset with synthetic data.

We conduct several additional experiments to gain further insight in the influence of injecting the dataset with synthetic data. We are interested in the change in performance when adding synthetic data compared to the change in performance the same amount of real data would make. Understanding this relation is key when generating synthetic data instead of collecting real data is considered.

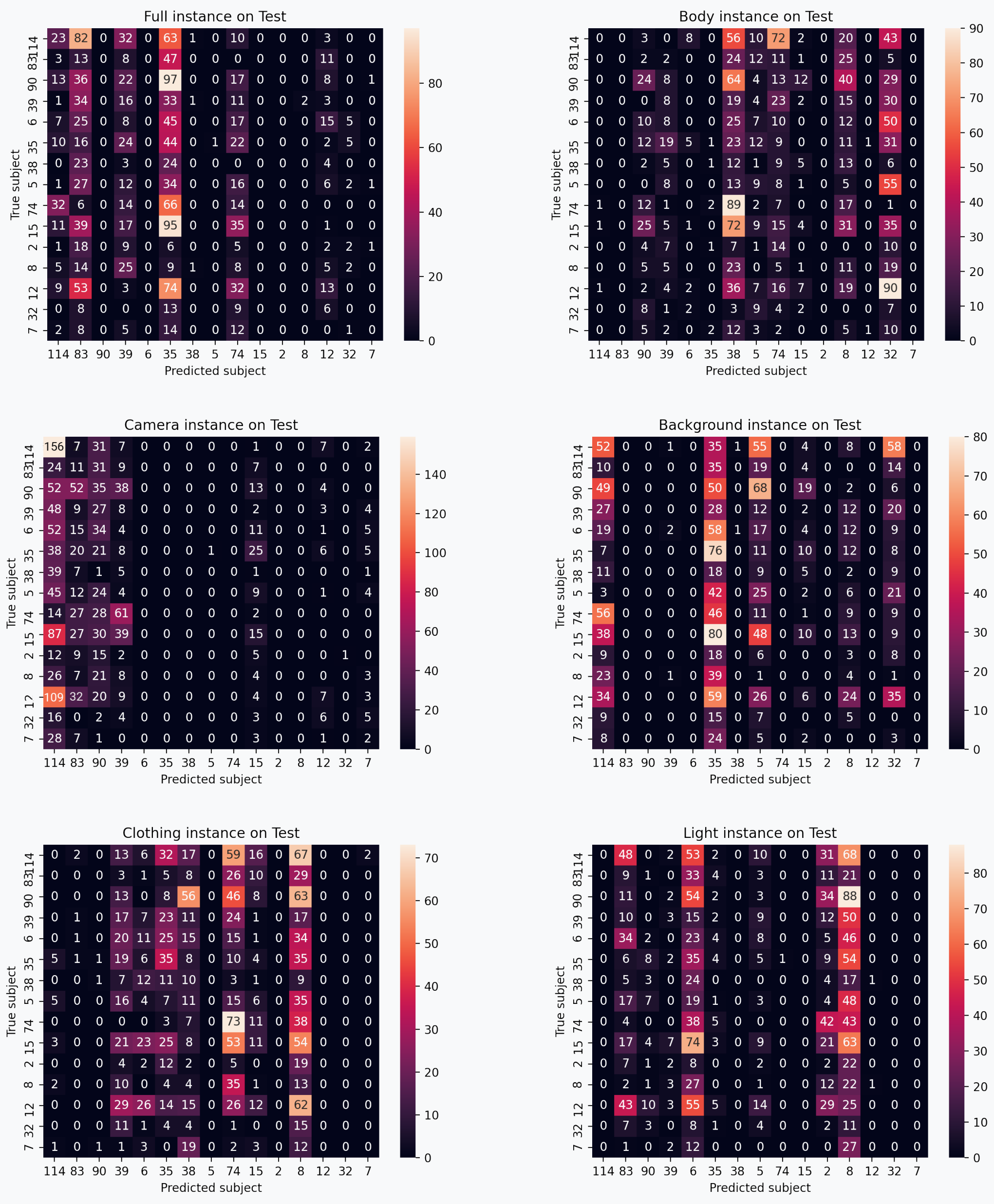


Figure 5.4: Confusion matrices of all 6 ablation instances tested on the testset.

5.3.2. Pipeline Architecture

This subsection described the violence detection pipeline as proposed by Van der Lugt (2019) [2] and as being used in production at Oddity.ai.

The violence detection pipeline consists of a *region proposal algorithm* (RPA), a base model that functions as feature extractor and a temporal module that results in a classification. See Figure 5.5 for a conceptual overview of the pipeline.

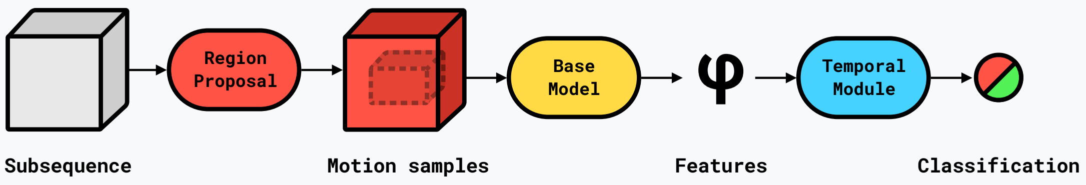


Figure 5.5: Conceptual overview of the Oddity.ai machine learning pipeline for violence detection.

Region Proposal Algorithm The first component of the pipeline is the region proposal algorithm. This algorithm is used to find *regions of interest* in the input video, i.e. spatial crops around violence, in order to increase the detection efficiency of the network. Without an RPA suboptimal situations could arise when violence happens only in a small region of the input image. The main assumption behind the RPA is that violence always expresses itself as substantial movement. The algorithm expects sequences of a fixed number of frames as input, meaning that longer videos are sliced into subsequences before applying the RPA. In the RPA, a *motion volume* is computed over the input based on difference in pixel brightness. See Figure 5.6 for an example visualisation of the calculation of a motion-volume. The motion volume is treated with various simple image transforms to compute a motion intensity threshold and to get a clear segmentation of the motion-volume from the background. These steps are all basic image operations with fast implementations, which is essential to keep the entire pipeline running in real-time. The minimum bounding box around this segmentation is used to crop the subsequence into a motion sample, which is the input for the next stage. More details on the RPA can be found in the work by Van der Lugt (2019) [2].

Base Model The base model is built upon the *Two-Stream Inflated 3D ConvNet* (I3D) by Carreira and Zisserman (2017) [12]. This is a very deep CNN for video classification trained on the Kinetics dataset. By stripping the classifying layer away and using the layer before as output, I3D is turned into an automatic video feature extractor. These features, denoted as ϕ in Figure 5.5, are inferred for every motion sample and then passed on to the temporal module.

Temporal Module The temporal module effectively smoothes the feature-activations of the base model over time and captures temporal patterns in these activations.

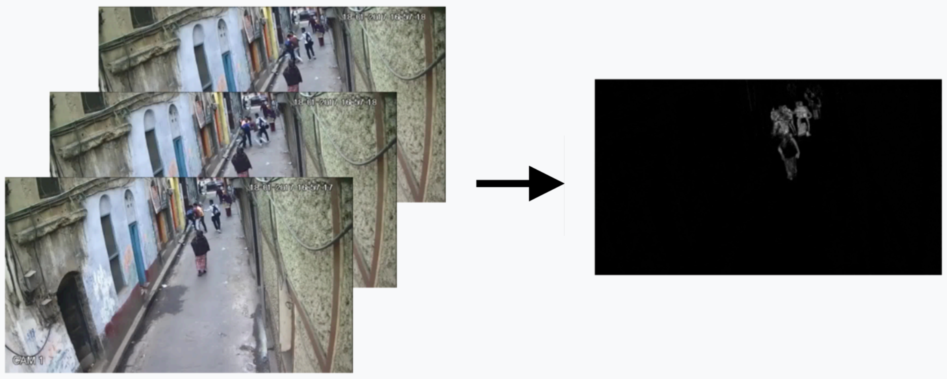


Figure 5.6: Example application of the maximum mean difference operation on a set of frames, resulting in a motion-volume. Figure taken from Van der Lugt (2019) [2]

The ability to capture these temporal patterns improves the violence-detection accuracy. This temporal modelling is achieved by using an LSTM layer, which is the main component of the temporal module. The LSTM activations are passed on to a fully-connected layer for the final violence or non-violence classification.

5.3.3. Source Motion from Video

The source motions used to generate synthetic videos of violence can be extracted from a large variety of videos. The following section walks through all sources we tried and treated for motion extraction.

Combat sports Violence is omnipresent in many combat sports. Most of these sports are centred around one-on-one fights in either a cage or on a (delimited) soft mat. Examples are boxing, kickboxing, karate, wrestling, Judo and Mixed Martial Arts. Many of these sports put a limit on the allowed movements or only allow a set of techniques. This naturally also puts a limit on the usability of these motions for our purposes because these techniques are most likely not representative of violence as seen on the streets.

Mixed martial arts (MMA) is known for its combination of elements from many other disciplines and its freestyle-like appearance. It is also very popular and large events such as the *Ultimate Fighting Championship* attract millions of viewers. This diversity in fighting technique and the availability of video footage makes MMA a suitable candidate for motion extraction.

Computer games Fighting, combat and crime are among the most popular computer game genres. For example, *Grand Threft Auto V* has a player count of 71 million [105], *Call of Duty: Warzone* has 50 million [106] and *Counter-Strike: Global Offensive* has 46 million [107]. Video game footage is generated on-the-fly by game engines that can nowadays produce almost photorealistic imagery. The potential amount of data available for motion extraction thus is enormous. However, many games have limitations that prevent us from extracting representative

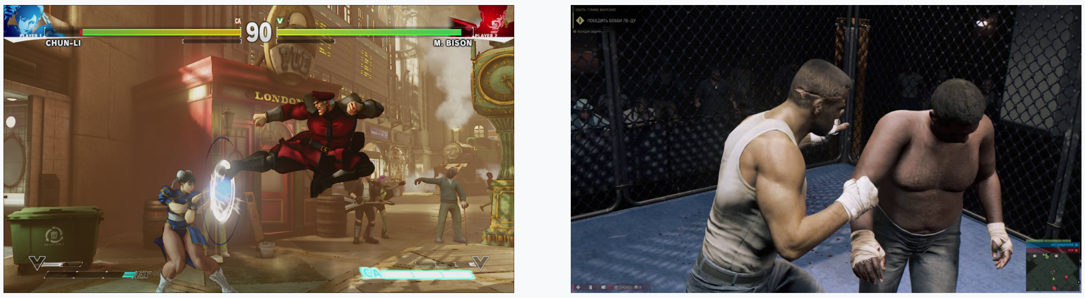


Figure 5.7: Left: screenshot of *Street Fighter V* gameplay. Figure from Ars Technica [108]. Right: screenshot of *Mafia 3* gameplay. Figure from YouTube [109].

motion from them.

First of all, the characters need to look and behave realistically for the motion extractor to work properly. The movements they make need to be physically plausible and the characters should not shrink, grow or distort. Games such as *Street Fighter V* contain characters that are out of proportion and that can move in unrealistic ways and the game contains exaggerated visual effects such as explosions on impact of a punch. See Figure 5.7 for a screenshot of the gameplay of this game.

Secondly, the fighting characters in the video game footage must be unoccluded for the motion extractor to function properly. This means the game user-interface should not overlap and the characters must not be partially out of view. See Figure 5.7 for an example of the game *Mafia 3*, where the main character is always out of view from the middle down.

Action movies Fights are common in movies and the *Action film* genre centres entirely around this. However, most movies depict fight scenes very stylistically. Quick camera movements and large amounts of motion blur are deliberately used to give scenes a fast appearance. Major occlusions and quick cuts between different takes are common, and are sometimes even used deliberately to obscure the fact that the fight is not real. The bodies of the actors are often not completely visible. We found that these factors combined make it very hard to find fight scenes that do meet our requirements for good motion extraction. An additional limitation of gathering material from movies is the relatively high cost of buying movies, certainly when only the violent parts of the whole movie are of interest, and the fact that finding all violent segments is time-consuming.

Violence datasets There exist precompiled sets of violent videos readily available to download. These are interesting candidates for our case.

The Violent Scenes Dataset by Demarty, Penet, Soleymani and Gravier (2015) [110] contains segments of violence from 32 movies and 86 short web videos such as *Saving Private Ryan* and *Pulp Fiction*. The Movie Fight dataset by Nievas, Suarez, Garcia and Sukthankar (2011) [111] is a similar dataset with violent segments from movies. The videos in these datasets suffer from the same visual limitations as mentioned before as they are extracted from movies. See Figure 5.8 for an example of an incorrectly estimated pose when the entire bodies of people are not visible.

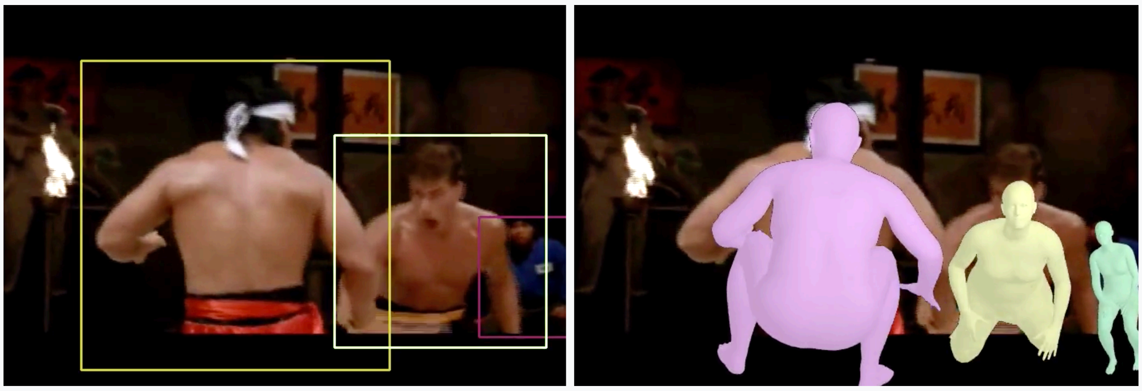


Figure 5.8: Frame of video *newfi83* from the Movie Fight dataset with estimated bounding boxes and 3D poses overlapped. The video shows the actors from the pelvis up, resulting in incorrect pose estimation.

Violent Flows by Hassner, Itcher, and Kliper-Gross (2012) [112] contains 246 videos from YouTube. The dataset is meant to reflect real-world scenarios of violence erupting in crowded places. However, the videos are so crowded that neither of the multi-person trackers in our motion extractor is able to confidently find people in them. See Figure 5.9 for screenshots of a few videos in Violent Flows.

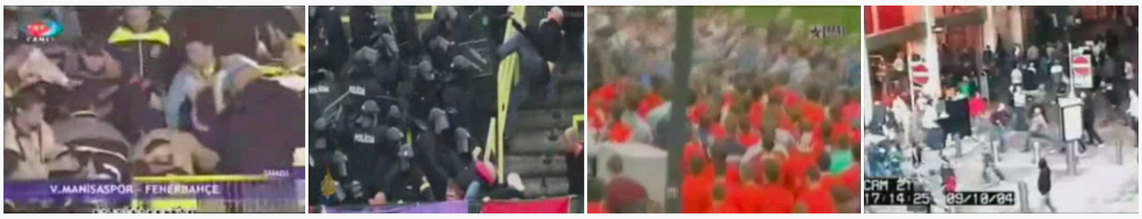


Figure 5.9: Screenshots of 4 videos from the Violent Flows dataset of violence in crowds.

The Hockey Fight dataset, also by Nievas, Suarez, Garcia and Sukthankar (2011) [111], contains 1000 videos of violence outbreaks in hockey matches of the National Hockey League. Many videos do not show the full bodies of the subjects, which makes motion extraction impossible. When the full body is visible, the multi-person tracker often locates only the people with least occlusions. See the first row of Figure 5.10 for an example where the bounding boxes found by YOLOv3 are projected onto the original frame. This error propagates through to the 3D pose estimator component. Additionally, this component seems to suffer from the low resolution of the videos. It is unable to consistently determine a persons height and the location of joints, resulting in jitter. See the last row of Figure 5.10 for a number of frames (in order but with gaps of 5 frames in between) with the estimated 3D pose as a SMPL mesh projected onto them. The jitter in height, pose and position of joints is clearly visible here.

The UCF-Crime dataset is created for anomaly detection in surveillance videos and contains 1900 videos of criminal behaviour, divided into several categories. All videos have a resolution of 320×240 pixels. The *Fighting* category contains 50 videos, which are processed by our motion extractor. However, the accuracy

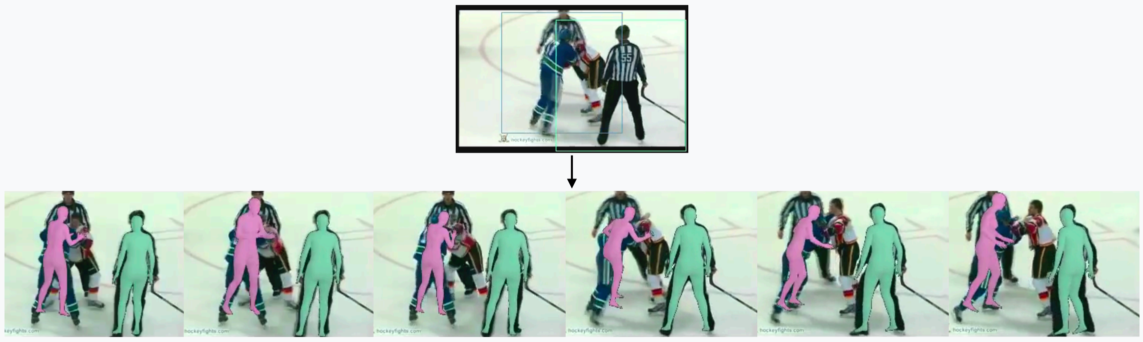


Figure 5.10: Frames of a video from the Hockey Fight dataset in which the error modes of the multi-person tracker (first row) and the 3D pose estimator (last row) are visible.

and overall quality of the motions extracted from the dataset is lower than we expected. Presumably the low resolution of the videos is at the root of this problem. Many videos also contain heavy occlusions, increasing the number of errors in the extracted motions. See Figure 5.11 for example stills of extracted motions projected onto the original videos. Many of these examples are not accurate enough for the violence to even be recognisable by the human eye or contain clear errors due to occlusions.

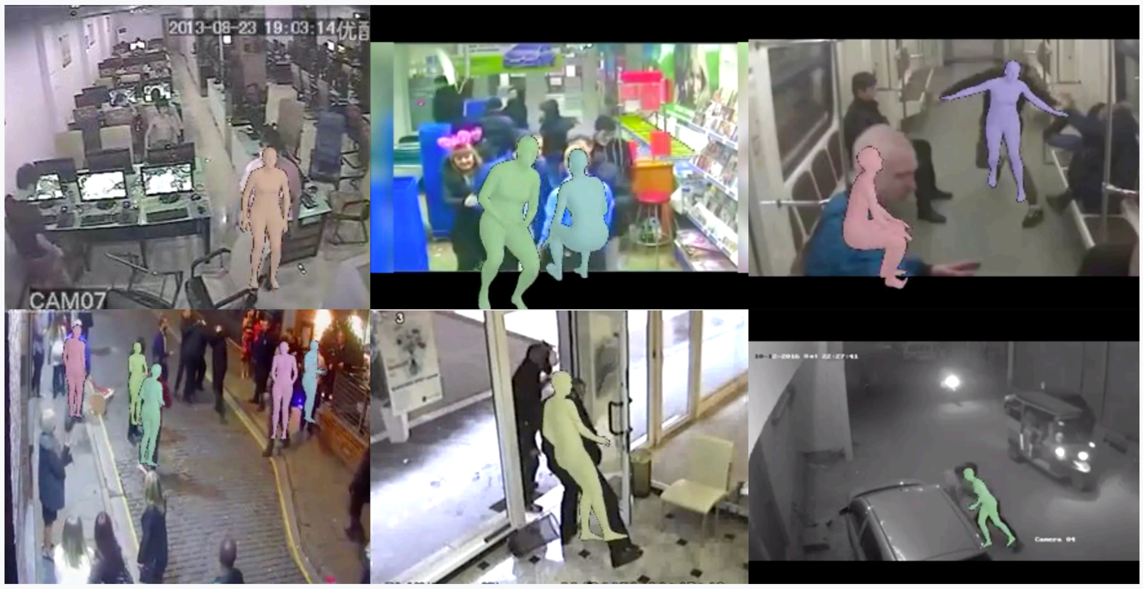


Figure 5.11: Stills of 3D pose estimates for 6 videos from the *Fighting* category of the UCF-Crime dataset projected onto the original videos.

Collecting videos Violence is often recorded and uploaded to popular video sharing platforms such as LiveLeak or Reddit, where the terms of usage do not prohibit such content. We scraped videos with titles and descriptions containing at least one term from (*fight, violence, violent, aggression, aggressive, assault, attack, rage*) and one term from (cctv, cctv footage, security camera, security, surveillance camera, surveillance, caught on camera). The terms in the first list should find videos with violence and the terms in the second list should ensure the videos are recorded with a static camera.

Using this strategy a large number of videos can be collected. However, most of them suffer from the same problems as mentioned before: many (large) occlusions and low resolution. This direction was not pursued any further than a shallow analysis.

Homemade videos Nowadays recording video at high resolution is easier than ever with the built-in cameras of mobile devices. This opens the door for a *do-it-yourself* approach to solving the problems met above. By acting out violent actions ourselves and recording them we can guarantee that there are no occlusions and the resolution is high enough. The Movement and Placement Assumption 1 (introduced in Subsection 4.3.4) allows us to record these actions individually as they will be mixed anyway. This is a major benefit for the prevention of occlusions and also a minimum requirement, as in practice the motion extractor is not able to function sufficiently on video footage in which multiple people are visible. It also allows us to record violent actions without hurting each other.

We recorded 58 videos of various violent and neutral (non-violent) actions performed by 5 actors. These actions include a variety of punches, hits, slaps, kicks, pushes and smashes. The videos duration mean is 8.28 seconds with a standard deviation of 5.08 seconds. See Figure 5.12 for still frames of 12 of the videos.

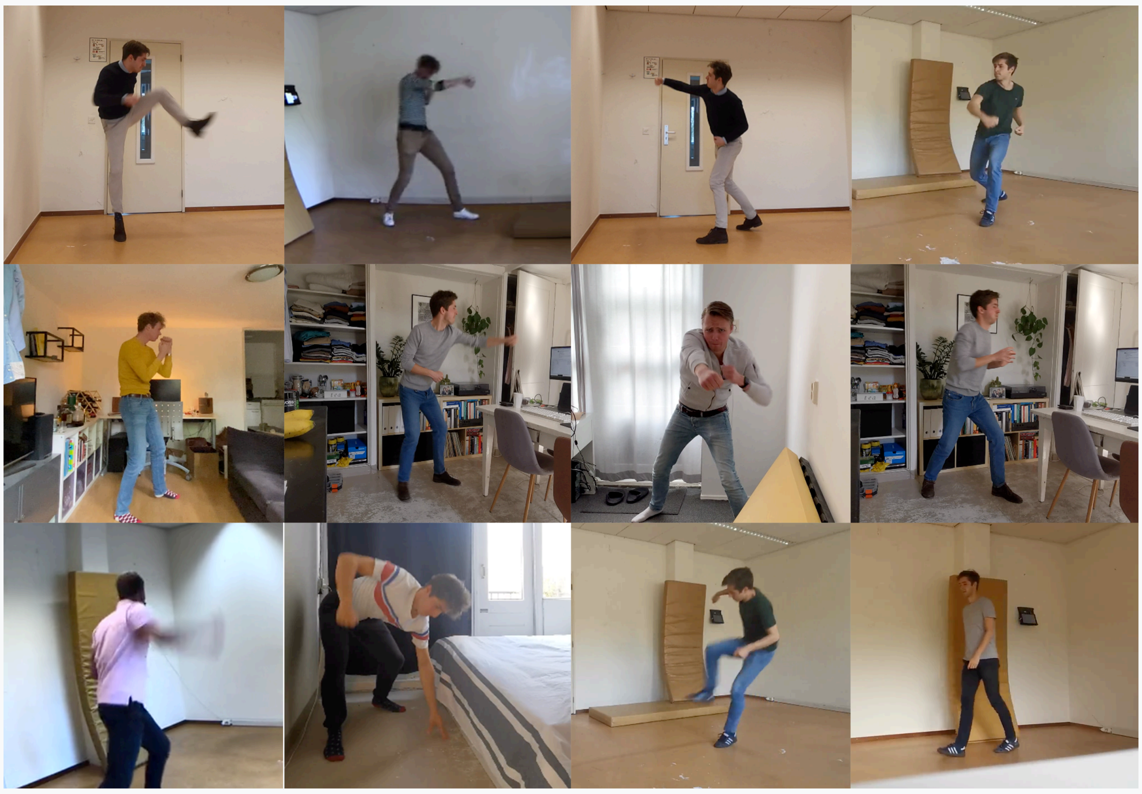


Figure 5.12: Stills of the videos we recorded for motion extraction.

Applying the motion extractor to these videos and projecting the estimated 3D poses back onto the original videos allows for a qualitative analysis of the estimations. See Figure 5.13 for still frames of 6 of the videos with the estimated

3D poses. Compared to the estimations done on different sources as discussed above, the errors are noticeably smaller. The estimation of global body pose and orientation is practically always correct. Errors only seem to occur when quick motions cause motion blur and at the hands and feet.

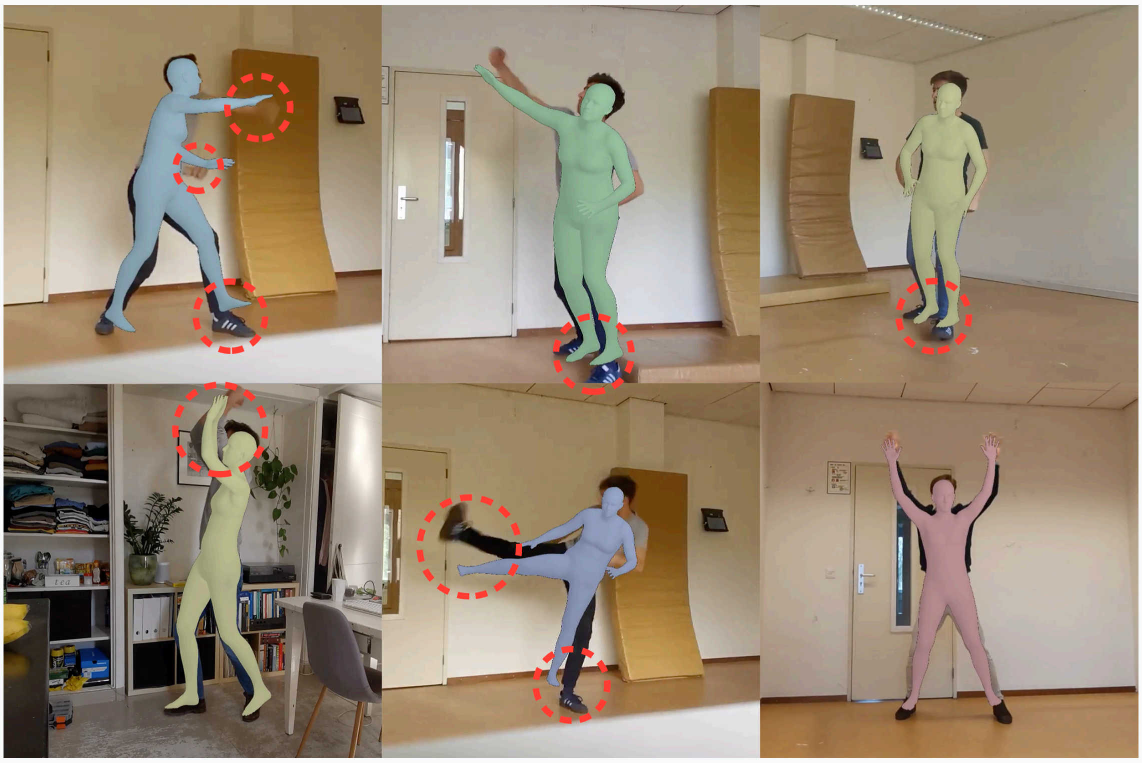


Figure 5.13: Stills of 3D poses estimated by our pipeline projected onto the original videos. Errors are circled with a dotted red line.

5.3.4. Motion Capture Data

Additional motion capture data is used to complement the motion set extracted from videos and make it more diverse. Just as with gait recognition, the large CMU MOCAP dataset is used to collect mocap data from. The actions that are used include a large amount of walking, running and jogging, but also bending, stretching, jumping and climbing. The violent actions are all labelled as boxing. The CMU set also includes actions such as striking or punching but these are all single strikes at a very low speed. As these are very unrealistic they are not included. See Table 5.4 for which CMU recordings were used.

The extracted motion and mocap data are mixed and treated exactly the same except for the smoothing, which is not applied to mocap data.

Subject	Recordings
1	1,2,3,5,6,7,8,9,11
2	1,2,3,4
5	1,2,3,4,5,6,7,9,10,11, 12,13,14,15,16,17,18, 19, 20
6	1
7	1,2,3,4,5,6,7,8,9,10, 11,12
8	1,2,3,4,5,6,7,8,9,11
9	1,2,3,4,5,6,7,8,9,10 11,12
10	4
12	1,2
13	17,18
15	1,3,9,13
17	10
32	1
35	1,2,3
38	1
39	1,2,3
74	1,2
83	27,28,29
90	22
114	13,14

Table 5.4: Overview of all CMU MOCAP recordings used for the generation of synthetic videos for violence detection.

5.3.5. Video Synthesis

The motions extracted from video and motion capture recordings were put through our synthetic data pipeline with the configuration settings as can be found in Table 5.5. We generated 6000 videos of maximum 250 frames at a resolution of 640x360 pixels in 4 runs of 1500 videos each. In total this took about 40 hours using an NVIDIA Tesla K80 GPU. This is significantly quicker than collecting this amount of data manually. See Table 5.5 for the exact parameter values used.

Figure 5.14 shows example frames of the generated videos.

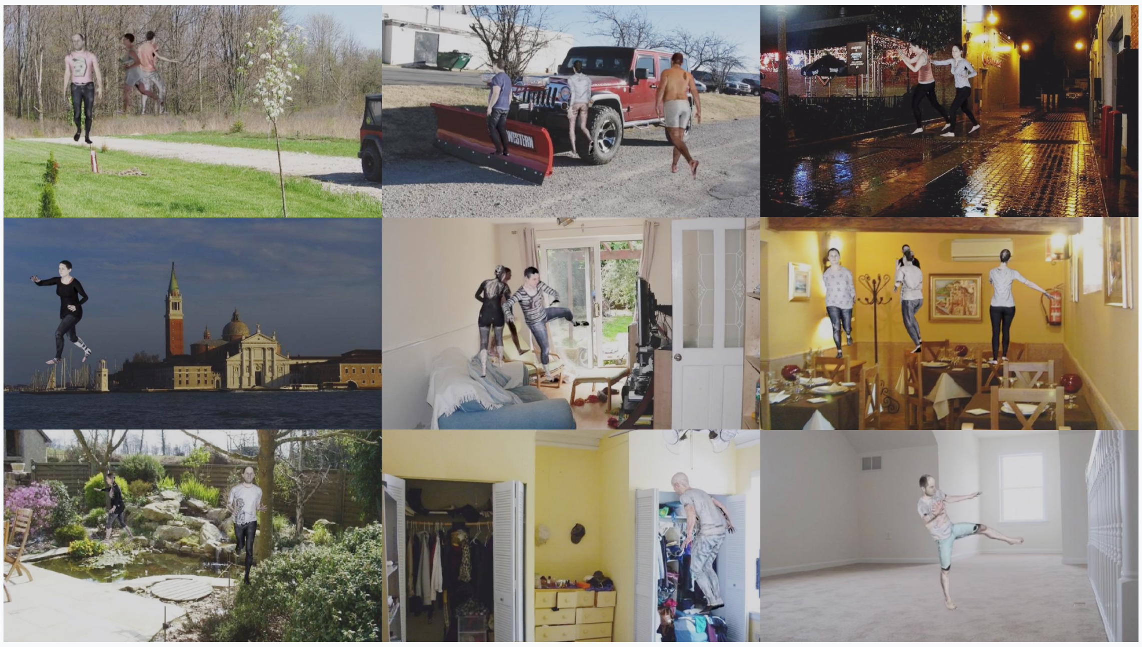


Figure 5.14: Still frames of 9 of the 6000 generated videos.

Parameter name	Value
Tracking method	YOLOv3
Max. number of frames per video	250
Max. number of persons per video	3
Dataset size	1500
Render height	360
Render width	640
Class combine strategy	Dominant ("violence")
Moving median smoothing	True
Original body shape	False
Light mode	Ambient
Camera angle variation	15°

Table 5.5: Configuration parameter values used for the violence detection data generation. This configuration was used 4 times to generate a total of 6000 videos.

5.3.6. Data Preprocessing

The first component of the pipeline, the region proposal algorithm, is not trainable so all data (real and synthetic videos) is passed through the RPA only once and the motion samples are saved to be used for all training sessions. A total of 44,704 motion samples are generated in this phase.

Following this, all motion samples need to be classified as being an example of either violence or neutral. The following steps are taken to achieve this.

1. *Before generation:* The source videos for the motion extractor and motion capture data are annotated on a per-frame basis with violence or neutral.

2. *During generation:* Minumum bounding boxes around every human model are calculated for every frame, these are mapped from 3D object space to 2D camera space and saved along with the generated video.

3. *After RPA:* For every motion sample the bounding boxes (from 2.) are used to determine which motions are visible and the annotations (from 1.) are used to determine whether they demonstrate violent actions. If there is no overlap, a motion sample is classified as neutral. Otherwise, it is staged for manual review.

The manual reviewal of these motion samples is necessary because the RPA might select a region that does not show the movement that is *essential* for the violence classification. For example, the RPA might select a region showing only the legs of a punching model but from the legs it might not be visible that person is indeed throwing punches. Further automation is possible but the (more philosophical) questions of *what constitutes violence* and *how small a crop can get before violence is no longer recognisable as violence* have to be answered for it to be feasible. The need for this manual reviewal is a major limitation as it is labour and time-consuming.

Next, the two classes are balanced to the same proportions as the dataset containing real examples as obtained by Oddity.ai by removing 5,731 randomly selected neutral motion samples. The resulting set contains 32,846 neutral and 6,127 violence synthetic motion samples. See Figure 5.15 for examples of the generated motion samples.

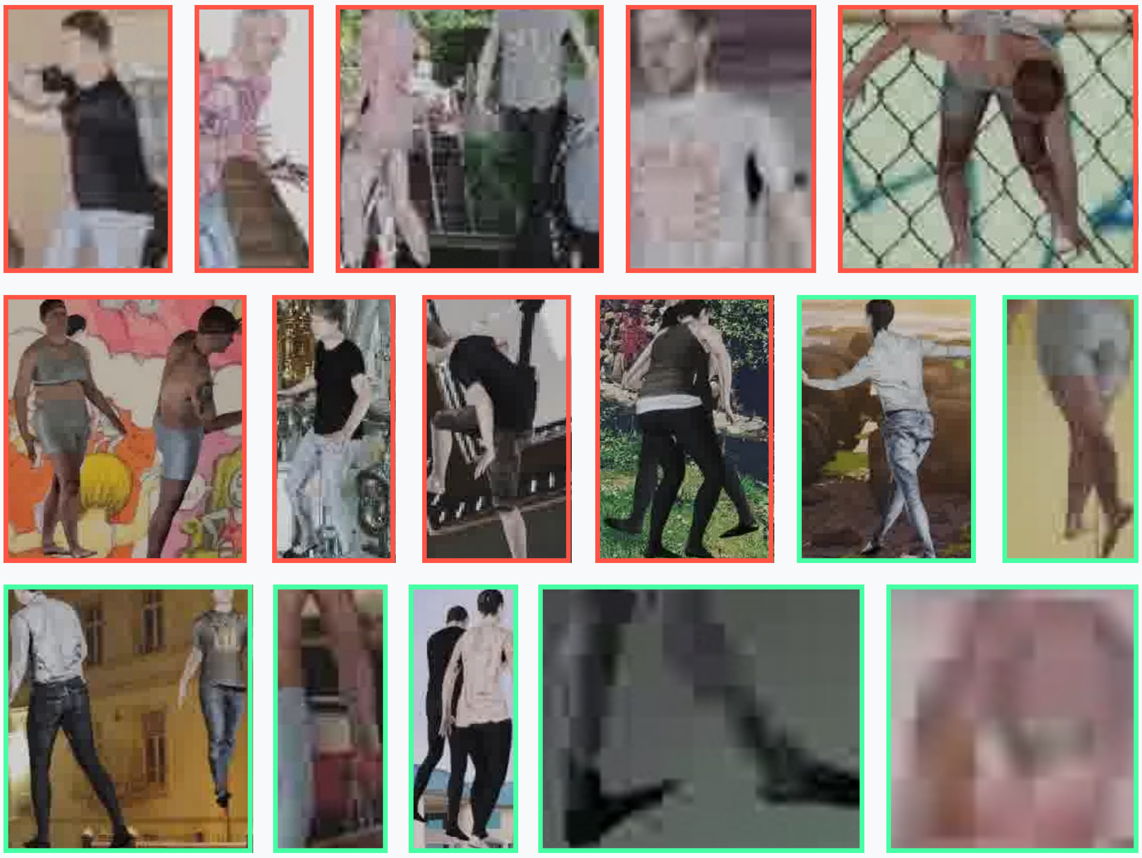


Figure 5.15: Still frames of 16 arbitrary fully processed motion samples. Motion sample classification is colour coded with red for violence and green for neutral.

5.3.7. Training and Testing

Parameters and duration The training parameters of the entire model are taken directly from practice at Oddity. This consists of finetuning the pretrained I3D weights of the base model independently and then training solely the temporal module. These work well in their experience and are not the subject of study for this thesis. For more information on the training of the Oddity.ai model see Van Der Lugt (2019) [2]. The time needed to train a single instance varies between 2 and 4 days on a Tesla K80 GPU, depending on the size of the training set that is used.

Datasets We use the default training set used at Oddity.ai as the baseline for our experiments. This dataset contains 51,517 neutral and 9,770 violence motion samples and is collected manually from real videos over the course of 2 years.

The synthetic dataset resulting from the data preprocessing phase, as described above, contains 32,846 neutral and 6,127 violence motion samples This is 63% of the number of motion samples that are in the real dataset.

Testing the trained models happens against the default testing set that is in use at Oddity. This testing set contains 4,815 neutral and 666 violence motion samples, which is roughly 9% of the size of the default training set. All motion samples come from surveillance footage that is not in the training set.

5.3.8. Instance naming

The different datasets and the instances trained on these datasets are labelled *R{real}-S{synthetic}*, where *{real}* is a number between 0 and 100 that refers to the proportion of real data that is in this dataset and *{synthetic}* is a number between 0 and 63 that refers to the amount of synthetic data in this dataset as a proportion of the real dataset size. For example, *R50-S50* is a dataset that contains 50% of the real dataset and an equal amount of synthetic data.

When extra clarity is needed, the icons 🟨 , ◩ and 🟦 are used to indicate datasets with respectively only real data, a mix of real and synthetic data and only synthetic data.

5.3.9. Experiment Results

Table 5.6 reports the amount of real data, synthetic data and the ROC-AUC score of the trained instance on the real testing set for all instances that were created. All instances maintain the proportion between violence and neutral motion samples of the complete real dataset of Oddity.ai (i.e. the proportion found in R100-S0). Figure 5.16 visualises the changing dataset size, the changing proportions of synthetic and real data and the maintained proportion of violence and neutral motion samples for three example instances.

	Instance	Real samples	Synthetic samples	ROC
🟨	**R50-S0**	**30,644** (50%)	**0**	**92,02%**
🟨	**R70-S0**	**42,901** (70%)	**0**	**93,79%**
🟨	**R100-S0**	**61,287** (100%)	**0**	**95,18%**
◩	**R50-S50**	**30,644** (50%)	**30,644** (50%)	**93,30%**
◩	**R70-S30**	**42,901** (70%)	**18,386** (30%)	**93,93%**
◩	**R100-S63**	**61,287** (100%)	**38,973** (63%)	**93,53%**
🟦	**R0-S63**	**0**	**38,973** (63%)	**88,16%**

Table 5.6: Amount of real motion samples and amount of synthetic motion samples for every dataset instance and the ROC-AUC score achieved by the model instances trained on these datasets.

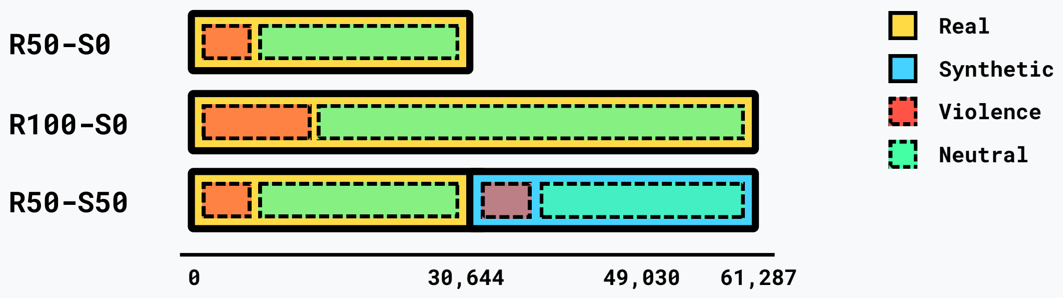


Figure 5.16: Visualisation of three example instances. Note that the dataset size changes, the proportion of synthetic and real data changes but the proportion of neutral and violence motion samples remains the same for all instances.

5.3.10. Experiment Discussion

Establishing a baseline The first step in discussing the different instances is establishing the baseline. Instances *R50-S0*, *R70-S0* and *R100-S0* serve this purpose.

	Instance	AUC
🟨	R50-S0	92,02%
🟨	R70-S0	93,79% (+1,77%)
🟨	R100-S0	95,18% (+3,16%)

Table 5.7: Baseline instances and their ROC-AUC scores.

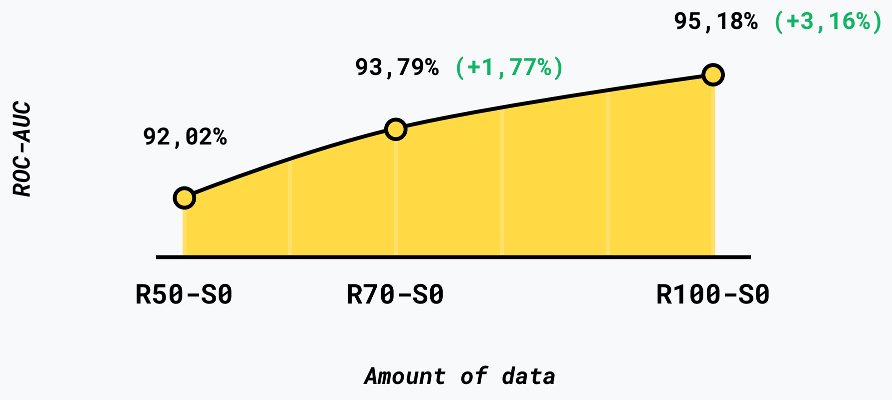


Figure 5.17: ROC-AUC scores of baseline instances plotted against their training set size.

In Table 5.7 the ROC-AUC scores of these instances are repeated. Figure 5.17

plots the scores against the amount of data in the instances. From these scores and the graph we can see that:

1. *R70-S0* performs better than *R50-S0*,
2. *R100-S0* performs better than *R70-S0*, and
3. the performance increase decreases with more data.

This is entirely as expected and establishes the baseline.

Synthetic data adds value Next we complement the baseline instances with synthetic data. Instances *R50-S50*, *R70-S30* and *R100-S63* are the result of this.

	Instance	AUC
🟨	R50-S0	92,02%
◩	R50-S50	93,30% (+1,28%)
🟨	R70-S0	93,79%
◩	R70-S30	93,93% (+0,14%)
🟨	R100-S0	95,18%
◩	R100-S63	93,53% (-1,65%)

Table 5.8: Synthetic complement instances and their ROC-AUC scores.

The ROC-AUC scores of these instances can be found in Table 5.8. From these scores we can see that:

1. *R50-S50* performs better than *R50-S0*,
2. *R70-S30* performs better than *R70-S0*, and
3. *R100-S63* performs worse than *R100-S0*.

Points 1. and 2. indicate that adding synthetic data increases performance. This is not the case for 3, where performance decreases.

Between *R100-S63* and other instances no changes to any (hyper)parameters have been made. The cause for this decrease thus surely lies in the addition of more synthetic data. This means that the synthetic data is biased in a way that is not completely representative of real data, or at least of the testing set, and that fitting the model on a lot of this biased data impairs its performance on real data.

Synthetic data is thus a partial substitute for ”missing” real data, but eventually it will impede performance.

Compared to real data Instances with equally sized training sets but different proportions between real/synthetic data are compared. This leads to insight in the value of synthetic data relative to real data.

	Instance	AUC
🟨	**R100-S0**	**95,18%**
◪	**R70-S30**	**93,93% (-1,25%)**
◪	**R50-S50**	**93,30% (-1,88%)**

Table 5.9: Instances with equally sized training sets but different proportions between real/synthetic data and their ROC-AUC scores.

The ROC-AUC scores of instances *R50-S50*, *R70-S30* and *R100-S0* are repeated in Table 5.9. From these scores we can see that:

1. *R70-S30* performs better than *R50-S50*, and
2. *R100-S0* performs better than *R70-S30*.

Points 1. and 2. indicate that using real data instead of synthetic data leads to better performance. This is as expected. Real data contains more predictive information than synthetic data.

Only synthetic data Instance *R0-S63* is trained solely on synthetic data, it thus has never seen a real example of violence. Its ROC-AUC score of 88,16%, which can be found in Table 5.10, is significantly worse than any other instance. This is in line with our previous finding that synthetic data is not as valuable as real data.

	Instance	AUC
🟨	**R50-S0**	**92,02%**
🟦	**R0-S63**	**88,16% (-3,86%)**

Table 5.10: ROC-AUC scores of the R0-S63 instance, which is trained on only synthetic data, and the R50-S0 instance for comparison.

However, it is significantly better than the ROC-AUC score a random guesser would obtain: 50%.

This indicates that there is indeed information to be learnt from the synthetic data that can be used to solve the real-world task.

Testing on synthetic data The same experiment can also be conducted in reverse. Instance *R100-S0* is trained solely on real data. In Table 5.11 its ROC-AUC score when tested on the entire synthetic dataset of *R0-S63* can be found.

Instance	AUC on synthetic data
🟨 **R100-S0**	**81,15%**

Table 5.11: ROC-AUC score of the R100-S0 instance tested on the synthetic dataset (R0-S63).

The result is in line with the test of *R0-S63* on the real testset: the synthetic data and real data overlap but also differ.

Finetuning R0-S63 We further investigated *R0-S63* as a prior for further training by fine-tuning it with real data. During this fine-tuning all layers but the last fully-connected classifier are being frozen. All hyperparameters are kept the same, just as with other training sessions. This way representations learnt on the synthetic data are preserved but the classification layer is adapted to real data. This is a form of *supervised domain adaptation*, as seen in Chapter 2.

We found that fine-tuning *R0-S63* with 10% of real data (*R10-S0*) improved classification performance with 1,28% to 89,44%. Fine-tuning with 30% of real data (*R30-S0*) improved performance with 2,19% to 90,35% and fine-tuning with 50% of real data (*R50-S0*) improved performance further to 90,95% (+2,79%). See Table 5.12 for all ROC-AUC scores.

Instance	AUC
🟦 **R0-S63**	**88,16%**
🟦 **R0-S63 +** 🟨 **R10-S0**	**89,44%** (+1,29%)
🟦 **R0-S63 +** 🟨 **R30-S0**	**90,35%** (+2,19%)
🟦 **R0-S63 +** 🟨 **R50-S0**	**90,95%** (+2,79%)

Table 5.12: ROC-AUC score of the R0-S63 instance and the instance when it is fine-tuned with real data.

This shows that *R0-S63* has learnt solid representations, both in the spatial, I3D-based base model as well as in the temporal LSTM layer, that can be put to good use on real data by only tweaking the classification layer.

Analysis of misclassifications Misclassification of testing set motion samples was inspected for every instance. This is especially interesting for *R100-S0*, *R100-S63* and *R0-S63*. See Figure 5.18 for an overview of still frames of the top-10 misclassifications (loss-based) of the mentioned instances.

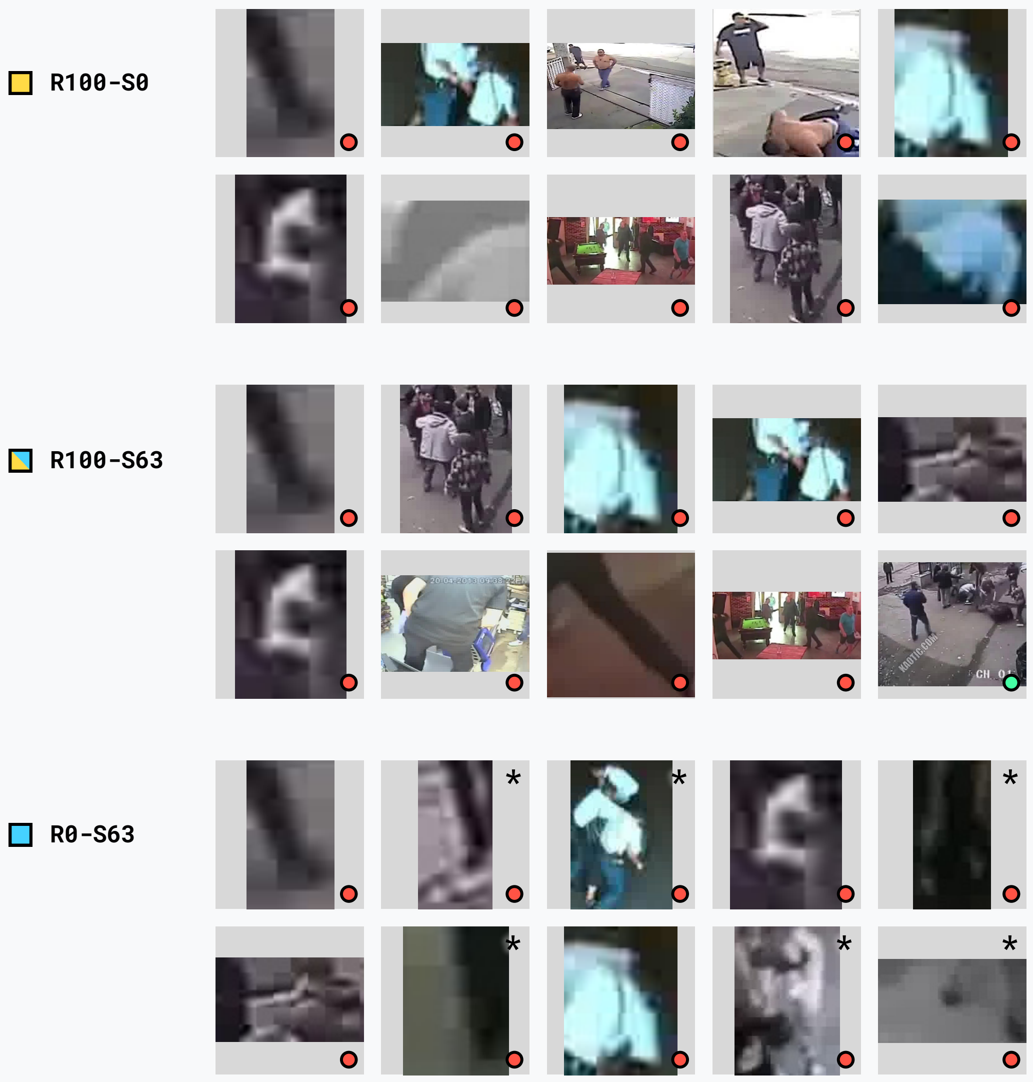


Figure 5.18: Single frames of the top-10 misclassifications for R100-S0, R100-S63 and R0-S63. Order from left-to-right, top-to-bottom. Red dots indicate a ground truth of violence, green dots neutral.

Note how the top-10 lists of *R100-S0* and *R100-S63* share 6 motion samples. The top-10 of *R0-S63* only shares 4 motion samples with that of *R100-S0* and *R100-S63* combined. When extending to top-50, this trend still holds.

Instances *R100-S0* and *R100-S63* thus consistently make different mistakes than *R0-S63*, where the base of real data lacks. This can be interpreted as an expression of the *reality gap* (see Subsection 2.5.1). The synthetic data does not contain sufficient information to seamlessly apply transfer learning.

Analysis of probability estimates The distributions of estimated probabilities on the testing set are compared for different instances. See Figure 5.19 for different plots of these distributions.

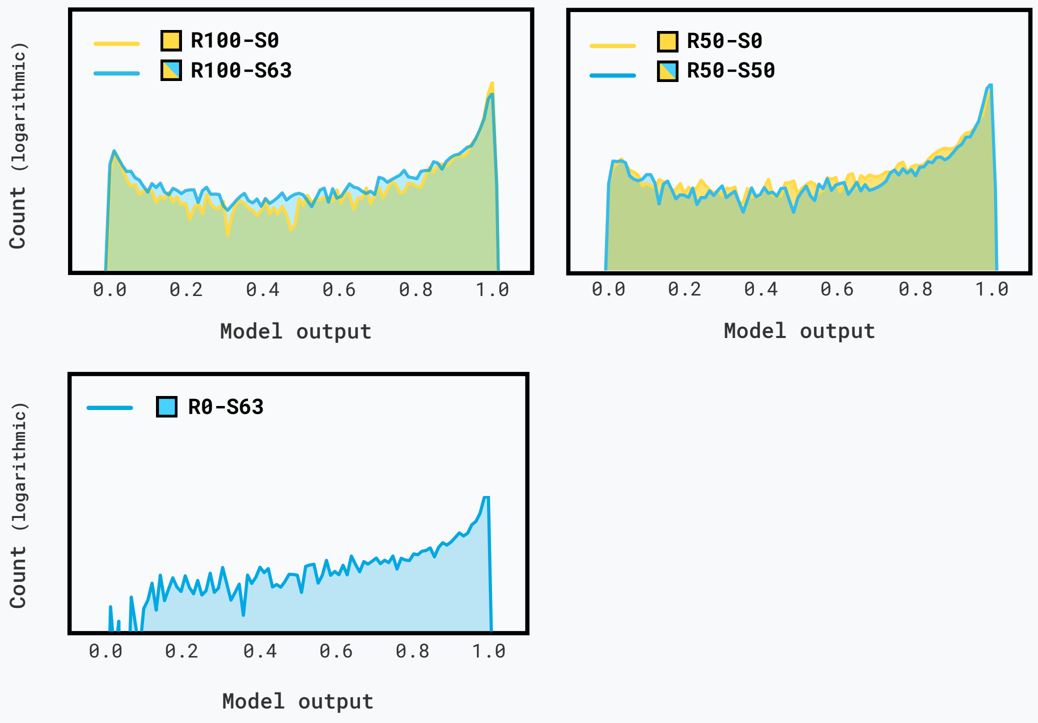


Figure 5.19: Estimated probability distributions for instances *R100-S0* vs. *R100-S63*, *R50-S0* vs. *R50-S50* and *R0-S63*. Kernel density estimation with a bandwidth of 0,015 is used to smoothen the counts. Note that distributions of estimated probability of *neutral* are reported. Distributions of estimated probability of *violence* can be inferred from these.

First, we treat the comparison of the distributions of *R100-S0* and *R100-S63* as can be found at the top left of Figure 5.19. It is apparent that *R100-S63* has more weight towards the center and *R100-S0* is more extreme in the estimated probabilities. This is interpreted as *R100-S0* being able to separate *neutral* and *violence* samples better and more confidently than *R100-S63*. This corresponds to the performance analysis above.

In the distributions of *R50-S0* and *R50-S50* the same pattern is visible but here it is in favour of the latter. This also corresponds with the performance measurements in terms of the ROC-AUC score.

Finally, we consider the distribution of *R0-S63* at the bottom of Figure 5.19. The rough distribution and the lack of weight towards the lower end are remarkable. It seems this model is less able to confidently predict *violence*.

6. Discussion

This chapter recaps the research problems, interprets the achieved results, discusses their implications and limitations and sheds light onto possible future research directions. Finally, a conclusion is drawn from the whole work.

6.1. Research Problem

The data synthesis pipeline as proposed by this work is a solution to answer the Main Research Question as defined in Chapter 1:

MAIN RESEARCH QUESTION

Can training data, that is synthesised using motion data that is captured from alternative sources, improve the classification performance of deep learning-based activity recognition systems?

In Chapters 2, 3 and 4 the following subproblems of answering the Main Research Question emerged, each with their own challenges:

1. Multi-person tracking and pose estimation to reliably extract spatiotemporal human motion data from video.
2. Combining motions into scenes using random variables and heuristics grounded in reality.
3. A powerful modelling and rendering environment.

These subproblems have all be addressed with this work, resulting in a powerful and versatile pipeline to generate synthetic video data for activity recognition related tasks.

Research Questions 1-3, that underlie the Main Research Question, have guided the evaluation experiments in Chapter 5. The results of these experiments and their implications are discussed in the next section. The Research Questions are answered in Section 6.4.

6.2. Discussion of Results

6.2.1. Design and Development

Solution Objectives The design and development of the synthetic data generation pipeline has culminated in a working implementation that meets all Solution Objectives as defined in Chapter 3. First of all, the pipeline takes videos as well as motion capture data as input and outputs synthetic videos. It is also able to run on consumer-grade hardware, even though our tests were run on professional data center GPUs. Lastly, the pipeline is also able to generate at least several hours of synthetic videos at a resolution of 640x360 within 48 hours. The design and development can thus genuinely be called a success.

Limitations Outside of the solution objectives, however, some limitations can be found.

First of all, the motion extractor is able to locate persons in 2D pixels space but unable to estimate a persons location in 3D space. This is caused by the one-person-at-a-time nature of current 3D pose estimation models that does not take environment into account at all. Inherently this lack of spatial information directly causes a major limitation of our pipeline: interactions between people cannot be replayed, as location and orientation of those people are key.

In the video synthesiser, this led to the three *Movement and Placement Assumptions* (see Subsection 4.3.4). These do not solve the *interaction limitation* from above but rather accept it and work with it. The assumptions serve as a guide for the generation of random movement and placement, as a complete lack of movement is even less realistic than generated movement. With this approach, all extracted motions can be combined with all others, instead of just motions from the same sources.

A major limitation of the random walk generation algorithm, in turn, is that it does not take pose motion into account at all. Because of this, awkward situations where pose motion contradicts the generated random walk can arise. For example, a model can have pose motion extracted from a person standing still applies while it is moved around by the generated random walk or the other way around.

The scene lighting modes are another limitation of the pipeline implementation. Two light modes were implemented, one inspired by direct sunlight and one by ambient studio light. The lighting conditions in the real world are endlessly varied and ideally, this is captured by our implementation. Compared to the other limitations of this work, we theorise this to be of minor importance.

Lastly, all limitations of the 3D pose estimation model in use, VIBE, seep through to our pipeline. VIBE achieves state-of-the-art results on multiple benchmarks but subjective analysis of in-the-wild application shows multiple common errors. There is much temporal stutter and errors when the video resolution is low or there is an occlusion happening. More details on this can be found in Subsection 5.3.3. The implication of this limitation is that inaccurate motions end up in our synthetic videos.

These limitations all decrease the realism of the synthetic data, thus potentially preventing the closing of the reality gap.

6.2.2. Gait Recognition

The Dutch National Police is interested in recognising gait from surveillance footage. As an early investigation to that goal, this work implemented a deep learning classifier, GaitNet, and used synthetic data to train it. Building such a system is a challenge in and of itself but our main focus is on the use of this task for an ablation study on pipeline parameters. The next subsection discussed that ablation study.

The synthetic data used for training and testing of GaitNet was generated using motion capture recordings of multiple subjects walking. This testing set used was generated with motion capture data that was not used in the training set, to reliably assess the generalisability to never-seen-before motions.

The trained models did not perform well on the subject classification task. The model that performed best on the test set achieved a classification accuracy of 13,98% and a top 5 accuracy of 46,16%, compared to the baseline accuracies of 6,66% and 33,33%. We suspect all models have been overfitted.

However, the main system task envisioned by the Police is ordering a list of surveillance footage clips on the likelihood that it contains a given person, such that an investigator that works from top to bottom finds relevant clips more quickly. This is very much more forgiving than classification, as any improvement in making distinctions between people adds value to the end-user. That is why, even though the classification accuracy in itself is not exceptional, we consider this proof-of-concept successful.

6.2.3. Ablation Study

The ablation study as described in Section 5.2 investigates causal effects of single components of the pipeline. Variation of background, clothing, lighting, camera tilt and body shape were all brought to a minimum one by one and a dataset was generated for each of these ablations using the motion capture data collected for GaitNet. A GaitNet instance was then trained on this dataset and tested against two testing sets. The gait recognition task served as the working example for the ablation study but other tasks would also have sufficed.

Testing showed that all ablations had a negative impact on prediction accuracy. This means that all 5 tested areas of variation in synthetic data are beneficial for the performance of models trained on that data. This is a valuable insight as it confirms the effectiveness of domain randomisation.

6.2.4. Violence Detection

To assess the impact of synthetic data on a real-life and production-ready deep learning activity recognition system, the task of violence detection was considered. This was done at Oddity.ai, where a state-of-the-art violence detection model is developed and brought into production.

After the ablation study, where only motion capture recordings were used, the entire pipeline is now deployed. The synthetic videos were based on both motion capture recordings as well as videos of violence taken from different domains, from which the motion was extracted. However, the motion extractor posed severe difficulties when processing real-life surveillance videos with complex interactions, occlusions or low resolution. To proceed, we recorded videos of actors pretending to engage in violent interaction in-house. This gave us complete control and this allowed us to prevent occlusions and ensure high video resolution. The use of these videos is a major limitation, though, as the violence is acted out and not real. Its usage thus compromises the realism of the synthetic data.

Using the generated synthetic data and the existing Oddity.ai training set, we constructed several datasets with different amounts of real and/or synthetic data. These datasets include:

- Datasets of different sizes without any synthetic data at all, to establish a baseline.
- Datasets equal in size but with different real/synthetic proportions, to compare the value of synthetic data with that of real data.
- Datasets with the same amount of real data but different amounts of synthetic data, to assess the absolute difference in performance when complementing a dataset with additional synthetic data.

Model instances were trained on these datasets and their performance was measured against a testing set containing only real videos. The experiments showed the following:

1. Complementing real data with synthetic data does improve model performance up to a point,
2. The same amount of real data is more valuable than synthetic data.
3. The synthetic data used for violence detection is biased but relevant respresentation can be learnt from it.

It is to be expected that real data is more valuable, performance-wise, but this should, of course, be seen in light of the performance-cost tradeoff. Synthetic data is significantly cheaper to acquire than real data. Depending on the exact cost of real data collection, which differs for each use case, this performance-cost tradeoff is worth it. This is certainly the case for violence detection, where data collection is not only costly but also legally and ethically questionable.

The bias in the synthetic data that led to performance decrease when adding a large amount of synthetic data can be explained by the limitations found Subsection 6.2.1 above. These are all limitations that could prevent the reality grap from being closed. We deem the interaction limitation and errors through VIBE as the most contributing causes. However, these are both beyond our scope and reach to solve. Improvements to the random walk generation algorithm and light modes remain.

Our fine-tuning experiment showed that representations learnt on the synthetic data are solid though and can be put to good use by adapting the classification layer using small amounts of real data. This does also indicate that there is room for improvement on the domain randomisation side as fine-tuning is a domain *adaptation* technique.

Overall, the results achieved with this experiment are very satisfactory. The theorised positive value synthetic data can have, is now empirically backed up and our pipeline has proven itself.

6.3. Future Research

The synthetic data pipeline as proposed in this work has achieved very promising results in experimental and production environments. Still, there is ample room for additional research. A few key areas to explore further rise up from this work.

First, further research to a good random walk generation algorithm is required when trying to close the reality gap further. It is not evident how a realistic position and movement distribution can be found to sample random walks from. Ideally this would be grounded in real-world position and movement data. What this real-world data looks like differs for each use case of the pipeline, so this will compromise versatility. Research into the heuristic use of pose motion for more realistic random walk generation is also promising. Here the main challenge again is keeping the pipeline versatile and applicable to a wide range of problems.

Second, research into spatiotemporal scene understanding is required. The 3D pose estimation techniques reviewed in this work estimate a single person at a time, disregarding all spatial relations between people. This has as implication that interactions between people cannot accurately be reconstructed as their relative position and orientation is unknown. Insights from scene understanding achieved in e.g. self-driving car research [113], [114] or the field of depth estimation [115] might be combined with the known location of people in 2D pixel space to estimate their depth. Novel research might also be conducted into a 3D pose estimation model that directly regresses to multiple spatial-interdependent poses.

Third, we found annotation of synthetic data samples to be a major bottleneck when generating big datasets. Due to random (spatial) cropping and (temporal) splitting, it is not always evident if a cropped clip should still be classified as an example of the activity of interest. This was especially the case in our violence detection experiment. Further research into leveraging the synthetic nature of the generated data for the automatic annotation is desirable. A starting point could be Snorkel by Ratner et al (2017) [116], who propose a framework to programmatically label training data.

Lastly, broader research into the bias in the synthetic data and how to further close the reality gap is desirable. There is a wide variety of factors that can influence this and investigating their impact requires even more extensive research than our ablation study. Existing domain randomisation factors might have introduced bias but the absence of a yet unknown factor might also be the root cause. A possible direction thus is the theorisation and implementation of more *visual realism*, such as using moving backgrounds instead of static images or a higher fidelity body model.

6.4. Conclusions

Innovation of deep learning models for activity recognition has rapidly driven the required amount of training data up due to increasingly complex models. Many difficulties can arise when collecting such large amounts of data, especially with the recent tightening of privacy legislation. For human activity recognition, this certainly poses a problem.

In academia, the automatic generation of synthetic data for training purposes has been put forward as a solution for data scarcity but research into synthetic video data is limited. With this work we propose a powerful and versatile pipeline for the generation of synthetic data for a multitude of activity recognition tasks. It is able to process motion capture recordings but also videos containing activities of interest, from which the spatiotemporal motion data will be extracted using pose estimation techniques. However, motion extraction from video turned out to be limited because of the state of current research in that field. A multitude of scenes containing the collected motions are algorithmically generated and using domain randomisation techniques the reality gap with real videos is closed. Domain randomisation adds variance to a number of parameters, which in our case include: actor gender, actor body shape, actor clothing style, scene lighting conditions and scene background. Finally, Blender is used to build the scenes in 3D space and render them into videos.

As subjects of evaluation experiments, we considered the cases of gait recognition and violence detection, where data collection has proven difficult. Our ablation study on the gait recognition task showed that all proposed domain randomisation techniques used at data generation time led to increased classifier performance. Using synthetic data we were also moderately capable of identifying people by their gait. Experimentation with the violence detection model also showed that synthetic data can successfully be used to improve model performance. We achieved this by complementing real datasets with synthetic data. This is an important result as synthetic data is significantly cheaper than real data and is completely privacy-preserving. However, we did find the synthetic data to be biased, leading to a performance decrease when increasing the amount of synthetic data.

Concluding, we have proposed a powerful and versatile synthetic video data generation pipeline in this work and have been able to affirm our research questions with it.

Bibliography

[1] *Website of oddity.ai*, `https://oddity.ai/`, Accessed: 2020-02-12.

[2] G. v. d. Lugt, *A deep learning pipeline for real-time violence recognition*, 2019.

[3] A. Sokolova and A. Konushin, "Pose-based deep gait recognition," *IET Biometrics*, vol. 8, no. 2, pp. 134–143, 2018.

[4] C. Wan, L. Wang, and V. V. Phoha, "A survey on gait recognition," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, p. 89, 2019.

[5] T. M. Hospedales, J. Li, S. Gong, and T. Xiang, "Identifying rare and subtle behaviors: A weakly supervised joint topic model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 12, pp. 2451–2464, 2011.

[6] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[7] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[8] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[9] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 843–852.

[10] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 7, pp. 1325–1339, 2013.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[12] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.

[13] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, *et al.*, "The" something something" video database for learning and evaluating visual common sense.," in *ICCV*, vol. 1, 2017, p. 3.

[14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[15] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid, “Learning from synthetic humans,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 109–117.

[16] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian, “Building generalizable agents with a realistic and rich 3d environment,” *arXiv preprint arXiv:1801.02209*, 2018.

[17] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, “Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1325–1339, Jul. 2014.

[18] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, “Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?” *arXiv preprint arXiv:1610.01983*, 2016.

[19] S. I. Nikolenko, “Synthetic data for deep learning,” *arXiv preprint arXiv:1909.11512*, 2019.

[20] T. von Marcard, R. Henschel, M. J. Black, B. Rosenhahn, and G. Pons-Moll, “Recovering accurate 3d human pose in the wild using imus and a moving camera,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 601–617.

[21] A. Arnab, C. Doersch, and A. Zisserman, “Exploiting temporal context for 3d human pose estimation in the wild,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3395–3404.

[22] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A design science research methodology for information systems research,” *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.

[23] D. A. Forsyth and J. Ponce, *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[24] J. Copeland, *Artificial intelligence: A philosophical introduction*. John Wiley & Sons, 2015.

[25] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[26] D. H. Hubel and T. N. Wiesel, “Receptive fields of single neurones in the cat’s striate cortex,” *The Journal of physiology*, vol. 148, no. 3, pp. 574–591, 1959.

[27] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[28] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, “Dive into deep learning,” *May*, vol. 19, p. 2019, 2019.

[29] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural computation*, vol. 22, no. 12, pp. 3207–3220, 2010.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[31] R. Poppe, "A survey on vision-based human action recognition," *Image and vision computing*, vol. 28, no. 6, pp. 976–990, 2010.

[32] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Computer vision and image understanding*, vol. 104, no. 2-3, pp. 90–126, 2006.

[33] S. Herath, M. Harandi, and F. Porikli, "Going deeper into action recognition: A survey," *Image and vision computing*, vol. 60, pp. 4–21, 2017.

[34] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 6450–6459.

[35] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. Patwary, M. Ali, Y. Yang, and Y. Zhou, "Deep learning scaling is predictable, empirically," *arXiv preprint arXiv:1712.00409*, 2017.

[36] A. Gaidon, A. Lopez, and F. Perronnin, "The reasonable effectiveness of synthetic visual data," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 899–901, Sep. 2018, ISSN: 1573-1405. DOI: 10.1007/s11263-018-1108-0. [Online]. Available: https://doi.org/10.1007/s11263-018-1108-0.

[37] J. J. Little and A. Verri, "Analysis of differential and matching methods for optical flow," in *[1989] Proceedings. Workshop on Visual Motion*, IEEE, 1989, pp. 173–180.

[38] J. L. Barron, D. J. Fleet, S. S. Beauchemin, and T. Burkitt, "Performance of optical flow techniques," in *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, 1992, pp. 236–242.

[39] C. C. Charalambous and A. A. Bharath, "A data augmentation methodology for training machine/deep learning gait recognition algorithms," *arXiv preprint arXiv:1610.07570*, 2016.

[40] M. Enzweiler and D. M. Gavrila, "A mixed generative-discriminative framework for pedestrian classification," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2008, pp. 1–8.

[41] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[42] C. Doersch and A. Zisserman, "Sim2real transfer learning for 3d human pose estimation: Motion to the rescue," in *Advances in Neural Information Processing Systems*, 2019, pp. 12 929–12 941.

[43] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 23–30.

[44] G. Wilson and D. J. Cook, “A survey of unsupervised deep domain adaptation,” *arXiv preprint arXiv:1812.02849*, 2019.

[45] H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, and M. Marchand, “Domain-adversarial neural networks,” *arXiv preprint arXiv:1412.4446*, 2014.

[46] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, “Training deep networks with synthetic data: Bridging the reality gap by domain randomization,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 969–977.

[47] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[48] A. Clark, J. Donahue, and K. Simonyan, “Efficient video generation on complex datasets,” *arXiv preprint arXiv:1907.06571*, 2019.

[49] W. Chen, H. Wang, Y. Li, H. Su, Z. Wang, C. Tu, D. Lischinski, D. Cohen-Or, and B. Chen, “Synthesizing training images for boosting human 3d pose estimation,” in *2016 Fourth International Conference on 3D Vision (3DV)*, IEEE, 2016, pp. 479–488.

[50] C. R. de Souza12, A. Gaidon, Y. Cabon, and A. M. López, “Procedural generation of videos to train deep action recognition networks,” 2017.

[51] C. M. G. Lab, *Carnegie mellon university motion capture database*, 2016.

[52] K. Mason, S. Vejdan, and S. Grijalva, “An” on the fly” framework for efficiently generating synthetic big data sets,” *arXiv preprint arXiv:1903.06798*, 2019.

[53] Z.-Q. Cheng, Y. Chen, R. R. Martin, T. Wu, and Z. Song, “Parametric modeling of 3d human body shape—a survey,” *Computers & Graphics*, vol. 71, pp. 88–100, 2018.

[54] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, “Scape: Shape completion and animation of people,” in *ACM transactions on graphics (TOG)*, ACM, vol. 24, 2005, pp. 408–416.

[55] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, “Smpl: A skinned multi-person linear model,” *ACM transactions on graphics (TOG)*, vol. 34, no. 6, p. 248, 2015.

[56] A. Toshev and C. Szegedy, “Deeppose: Human pose estimation via deep neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1653–1660.

[57] S. Johnson and M. Everingham, “Clustered pose and nonlinear appearance models for human pose estimation,” in *Proceedings of the British Machine Vision Conference*, doi:10.5244/C.24.12, 2010.

[58] B. Sapp and B. Taskar, "Modec: Multimodal decomposable models for human pose estimation," in *In Proc. CVPR*, 2013.

[59] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2014.

[60] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European conference on computer vision*, Springer, 2016, pp. 483–499.

[61] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4724–4732.

[62] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele, "Deepcut: Joint subset partition and labeling for multi person pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4929–4937.

[63] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *arXiv preprint arXiv:1812.08008*, 2018.

[64] R. A. Güler, N. Neverova, and I. Kokkinos, "Densepose: Dense human pose estimation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7297–7306.

[65] T. Golda, T. Kalb, A. Schumann, and J. Beyerer, "Human pose estimation for real-world crowded scenarios," in *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, IEEE, 2019, pp. 1–8.

[66] L. Sigal and M. J. Black, "Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion," *Brown Univertsity TR*, vol. 120, 2006.

[67] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt, "Monocular 3d human pose estimation in the wild using improved cnn supervision," in *3D Vision (3DV), 2017 Fifth International Conference on*, IEEE, 2017. DOI: `10.1109/3dv.2017.00064`. [Online]. Available: `http://gvv.mpi-inf.mpg.de/3dhp_dataset`.

[68] T. von Marcard, R. Henschel, M. Black, B. Rosenhahn, and G. Pons-Moll, "Recovering accurate 3d human pose in the wild using imus and a moving camera," in *European Conference on Computer Vision (ECCV)*, Sep. 2018.

[69] S. Li and A. B. Chan, "3d human pose estimation from monocular images with deep convolutional neural network," in *Asian Conference on Computer Vision*, Springer, 2014, pp. 332–347.

[70] G. Varol, D. Ceylan, B. Russell, J. Yang, E. Yumer, I. Laptev, and C. Schmid, "Bodynet: Volumetric inference of 3d human body shapes," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 20–36.

[71] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black, “Keep it smpl: Automatic estimation of 3d human pose and shape from a single image,” in *European Conference on Computer Vision*, Springer, 2016, pp. 561–578.

[72] C.-H. Chen and D. Ramanan, “3d human pose estimation= 2d pose estimation+ matching,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7035–7043.

[73] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. Wei, “Towards 3d human pose estimation in the wild: A weakly-supervised approach,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 398–407.

[74] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, “End-to-end recovery of human shape and pose,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7122–7131.

[75] N. Kolotouros, G. Pavlakos, M. J. Black, and K. Daniilidis, “Learning to reconstruct 3d human pose and shape via model-fitting in the loop,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2252–2261.

[76] W. Xu, A. Chatterjee, M. Zollhöfer, H. Rhodin, D. Mehta, H.-P. Seidel, and C. Theobalt, “Monoperfcap: Human performance capture from monocular video,” *ACM Transactions on Graphics (ToG)*, vol. 37, no. 2, p. 27, 2018.

[77] M. Dou, S. Khamis, Y. Degtyarev, P. Davidson, S. R. Fanello, A. Kowdle, S. O. Escolano, C. Rhemann, D. Kim, J. Taylor, *et al.*, “Fusion4d: Real-time performance capture of challenging scenes,” *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 114, 2016.

[78] M. Habermann, W. Xu, M. Zollhoefer, G. Pons-Moll, and C. Theobalt, “Livecap: Real-time human performance capture from monocular video,” *ACM Transactions on Graphics (TOG)*, vol. 38, no. 2, p. 14, 2019.

[79] X. B. Peng, A. Kanazawa, J. Malik, P. Abbeel, and S. Levine, “Sfv: Reinforcement learning of physical skills from videos,” *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, p. 178, 2019.

[80] K. Wang, L. Lin, C. Jiang, C. Qian, and P. Wei, “3d human pose machines with self-supervised learning,” *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[81] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli, “3d human pose estimation in video with temporal convolutions and semi-supervised training,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7753–7762.

[82] A. Kanazawa, J. Y. Zhang, P. Felsen, and J. Malik, “Learning 3d human dynamics from video,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5614–5623.

[83] M. Kocabas, N. Athanasiou, and M. J. Black, “Vibe: Video inference for human body pose and shape estimation,” *arXiv preprint arXiv:1912.05656*, 2019.

[84] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, M. Elgharib, P. Fua, H.-P. Seidel, H. Rhodin, G. Pons-Moll, and C. Theobalt, “Xnect: Real-time multi-person 3d human pose estimation with a single rgb camera,” *arXiv preprint arXiv:1907.00837*, 2019.

[85] K. Soomro, A. R. Zamir, and M. Shah, “Ucf101: A dataset of 101 human actions classes from videos in the wild,” *arXiv preprint arXiv:1212.0402*, 2012.

[86] M. Marszałek, I. Laptev, and C. Schmid, “Actions in context,” in *IEEE Conference on Computer Vision & Pattern Recognition*, 2009.

[87] C. Schuldt, I. Laptev, and B. Caputo, “Recognizing human actions: A local svm approach,” in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, IEEE, vol. 3, 2004, pp. 32–36.

[88] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in *CVPR*, 2014.

[89] H. Zhao, Z. Yan, L. Torresani, and A. Torralba, “Hacs: Human action clips and segments dataset for recognition and temporal localization,” *arXiv preprint arXiv:1712.09374*, 2019.

[90] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.

[91] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[92] I. Maxon Computer, *Cinema4d*. [Online]. Available: https://www.maxon.net/en-us/products/cinema-4d/overview/.

[93] U. Technologies, *Unity*. [Online]. Available: https://unity.com/.

[94] I. Epic Games, *Unreal engine*. [Online]. Available: https://www.unrealengine.com/en-US/.

[95] I. Autodesk, *3ds max*. [Online]. Available: https://www.autodesk.nl/products/3ds-max/overview.

[96] B. O. Community, *Blender - a 3d modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org.

[97] T. Roosendaal, *Fbx binary file format specification - blender developers blog*, Stichting Blender Foundation, Amsterdam: Blender Foundation, 2013. [Online]. Available: https://code.blender.org/2013/08/fbx-binary-file-format-specification/.

[98] R. Poppe, S. Van Der Zee, D. K. Heylen, and P. J. Taylor, “Amab: Automated measurement and analysis of body motion,” *Behavior research methods*, vol. 46, no. 3, pp. 625–633, 2014.

[99] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, “Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop,” *arXiv preprint arXiv:1506.03365*, 2015.

[100] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[101] *Surveillance cameras - 12 things you need to know*, Accessed: 2020-07-14. [Online]. Available: https://insights.htacertified.org/articles/surveillance-cameras-12-things-you-need-know.

[102] *Hoogste kwaliteit relevante beelden*, Accessed: 2020-07-14. [Online]. Available: https://www.boschsecurity.com/nl/nl/oplossingen/videosystemen/beeldkwaliteit/.

[103] M. Loper, N. Mahmood, and M. J. Black, "Mosh: Motion and shape capture from sparse markers," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 6, pp. 1–13, 2014.

[104] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black, "Amass: Archive of motion capture as surface shapes," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5442–5451.

[105] *Gta v's ps4 population is up to five times more than pc's – and is bigger than spain*, Accessed: 2020-06-16. [Online]. Available: https://www.pcgamesn.com/ps4-player-population.

[106] *Call of duty: Warzone hits 50m players in first month*, Accessed: 2020-06-16. [Online]. Available: https://www.polygon.com/2020/4/10/21216734/call-of-duty-warzone-50-million-player-count.

[107] *Valve leaks steam game player counts; we have the numbers*, Accessed: 2020-06-16. [Online]. Available: https://arstechnica.com/gaming/2018/07/steam-data-leak-reveals-precise-player-count-for-thousands-of-games/.

[108] *Street fighter v review: Definitely good, definitely unfinished*, Accessed: 2020-06-16. [Online]. Available: https://arstechnica.com/gaming/2016/02/street-fighter-v-review-definitely-good-definitely-unfinished/.

[109] *Mafia 3 box cage fight*, Accessed: 2020-06-16. [Online]. Available: https://www.youtube.com/watch?v=T5TboqE2vh0.

[110] C.-H. Demarty, C. Penet, M. Soleymani, and G. Gravier, "Vsd, a public dataset for the detection of violent scenes in movies: Design, annotation, analysis and evaluation," *Multimedia Tools and Applications*, vol. 74, no. 17, pp. 7379–7404, 2015.

[111] E. B. Nievas, O. D. Suarez, G. B. García, and R. Sukthankar, "Violence detection in video using computer vision techniques," in *International conference on Computer analysis of images and patterns*, Springer, 2011, pp. 332–339.

[112] T. Hassner, Y. Itcher, and O. Kliper-Gross, "Violent flows: Real-time detection of violent crowd behavior," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, IEEE, 2012, pp. 1–6.

[113] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial cnn for traffic scene understanding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[114] L. Schneider, M. Cordts, T. Rehfeld, D. Pfeiffer, M. Enzweiler, U. Franke, M. Pollefeys, and S. Roth, “Semantic stixels: Depth is not enough,” in *2016 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2016, pp. 110–117.

[115] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova, “Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8001–8008.

[116] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, “Snorkel: Rapid training data creation with weak supervision,” in *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, NIH Public Access, vol. 11, 2017, p. 269.