Multiple Object Tracking and Segmentation: Leveraging Segmentation to Generate Online Learned Object Representations (and other methods)

John Allen
Supervisor: Prof. Andrea Cavallaro
MSc. Artificial Intelligence
Department of Electronic Engineering and Computer Science
QMUL
Module Code: ECS750P

August 2019

Abstract

This report presents a novel approach to Multiple Object Tracking. Using a tracking by detection pipeline, segmentation results output from an object detector are directly utilised in order to generate online learned object representations, which can then be compared through the use of scale independent similarity measure, which incorporates both raw image data and segmentation information. Tracking results for this method are reasonably poor due to that fact that building representations robust enough to be able to reliably identify objects in the first few frames of a video sequence, where only a few stored representations can be cross referenced. This method does however extend beyond the normal bounding box output for most state of the art object trackers, and also incorporates segmentation through the use of an Object Detector with a segmentation branch.

Code available at: https://github.com/ilovecocolade/Multiple-Object-Tracking-Masters-Project

Contents

Chapter 1

Introduction

1.1 Motivation

A principal area of interest in computer vision and robotics, especially with regard to the recent surge in autonomous driving, is the problem of Multiple Object Tracking. In order for robotic systems to interact safely with, and to traverse, their environments in close proximity to external agents with unpredictable behaviour, they must have some ability to track the motion and position of these agents. Once the locations and motions of these object can be inferred accurately from visual data, avenues such as intention prediction and action recognition can be explored. The ability to track objects in real time enables the development of more complex, safer and increasingly efficient systems, which as a result can be much more deeply integrated into modern society.

Aside from consumer applications, the ability to track multiple objects is vital in automated manufacturing and other commercial domains. As well as incorporation to robotic systems, vision systems alone can utilise Multiple Object Tracking for security and retail. By using vision to understand where an object or person is, and what path someone has followed in any particular environment, indicative algorithms can be deployed anywhere that CCTV can be mounted. If deployed in a supermarket for example, Multiple Object Tracking could be used to store the paths people take through the supermarket and this information could be used by Grocers to determine where to strategically place goods in order to increase purchases. In summary, the motivation for this work stems from the demand for models capable of Multiple Object Tracking as predictive and autonomous systems become increasingly utilised in both consumer and commercial domains.

In the realm of scientific research, Multiple Object Tracking is only recently seeing significant breakthroughs, likely due to the introduction of Deep Learning, which has revolutionised computer science as a whole. Because Multiple Object Tracking is a less mature area of Computer Vision, progress can be made quickly with the adoption of techniques applied in other similar domains like Single Object Tracking and Object Detection.

The principal task of this project is to introduce a novel technique for Multiple Object Tracking and compare it to the most desirable traits of a high performance Multiple Object Tracker.

1.2 Problem Definition

The problem of Multiple Object Tracking can be broken down into two distinct steps; detecting the objects to be tracked in video frames; and using data association to match these detections such that objects have identities which can be propagated across the temporal dimension of a video sequence. The task of Object Detection is non trivial, but it is well established. State of the art object trackers can localise and segment objects from numerous classes with high accuracy. Once detections can be accurately computed, associating data to cross reference and match instances between frames can be done.

One premise of Multiple Object Tracking which makes it unique to Single Object Tracking is the idea of multiple targets. For a Single Object Tracker, one target must be identified in a frame and then localised. For a Multiple Object Tracker, localisation of multiple targets must be computed and each object must be matched to an object in the previous frame, or identified as an object introduced into a video sequence at that particular frame. Multiple Object Trackers must also be robust to object identity switching. For example, if two objects approach one another the tracker must be able to identify clearly which object is which and avoid the predicted identity of the two objects swapping as they coalesce in the image frame. This means Multiple Object Trackers must generate robust object representations which can be used to uniquely identify many objects. Typically in Multiple Object Tracking, many of the objects to be tracked are of the same type, like people or cars. This means Multiple Object Trackers must have the architecture to be able to distinguish similar objects robustly, in a way which has a more unique identification procedure than an Object Detector.

One other requirement for high performance Multiple Object Trackers is low computational expense. In order for systems to utilise Multiple Object Tracking practically, they must be able to deploy and execute algorithms in real time. For example, self driving cars tracking algorithms must operate at extremely high frequencies and latencies in order to meet safety standards.

1.3 Contributions

This project heavily utilises the state of the art Object Detection network Mask R-CNN [15]. Mask R-CNN [15] is used to generate detection results for all of the tracking by detection pipelines discussed in this report. This project also makes use of SiamMask [42], a class agnostic single object tracker. The implementations in this report were developed using a dataset provided by Professor Andrea Cavallaro (a.cavallaro@qmul.ac.uk) and Jeremy Keusters (j.r.d.keusters@se18@qmul.ac.uk), containing video sequences of either one or two people interacting with various different types of cups. The 2D-MOT 2015 [20] dataset was also briefly used.

1.4 Structure

This report follows a conventional structure and includes a literature review and analysis of state of the art methods; a methodology detailing the theory and implementation of ideas used; analysis of the implementation; and a conclusion summarising the achievements of the implementation and discussing future work.

Chapter 2

Background

This chapter discusses relevant work in Object Detection, Object tracking and Multiple Object tracking. The development of these fields in Computer Vision (CV) is explained according to the narrative of Deep Learning and Neural Networks, which have been extensively implemented across an array of problems in not only CV but in Computer Science as a whole in the last decade. The introduction and consequences of Convolutional Neural Networks (CNNs) is described and discussed alongside its implementations in CV problems. The significance of past challenges relevant to all three previously mentioned CV problems are described with the relevant research which highlighted; the challenges and the research which led to these challenges being overcome.

2.1 Object Detection

The task of Object Detection, crudely, is the identification of relevant objects in images and can be dismantled into four distinct sub problems; Image Classification - Is there a specific object in this image? Semantic Segmentation - Which image elements' correspond to the specific objects? Object Localisation - How many instances of a specific object are in an image and where are these instances located? Instance Segmentation - Which image elements are occupied by each individual instance of a specific object? Object detection is a well established area of Computer Vision and is a beneficiary of a large amount of research due to its multitude of applications.

Since the early 2010's Deep Learning and principally, Convolutional Neural Networks (CNN's) have allowed a significant increase in the performance of Object Detectors. Beginning with AlexNet [18] in 2012, which computed image classification and was entered into the ImageNet Large Scale Visual Recognition Challenge [31]; a 20,000 class, several hundred image classification task. AlexNet [18] contained five convolutional layers followed by three fully connected layers, the last of which is a SoftMax layer. The SoftMax layer is used to output a probability for each class learned by the network, the sum of which is 1. Network activation functions were chosen to be Rectified Linear Units, which showed significantly improved performance compared to the previously conventional functions; Tanh and Sigmoid, shown in figure 2.1. Max Pooling was also introduced to decrease the image size as it is propagated through the network, reduce the number of networks parameters and in turn to reduce the computational cost of training and forward passes of the network.

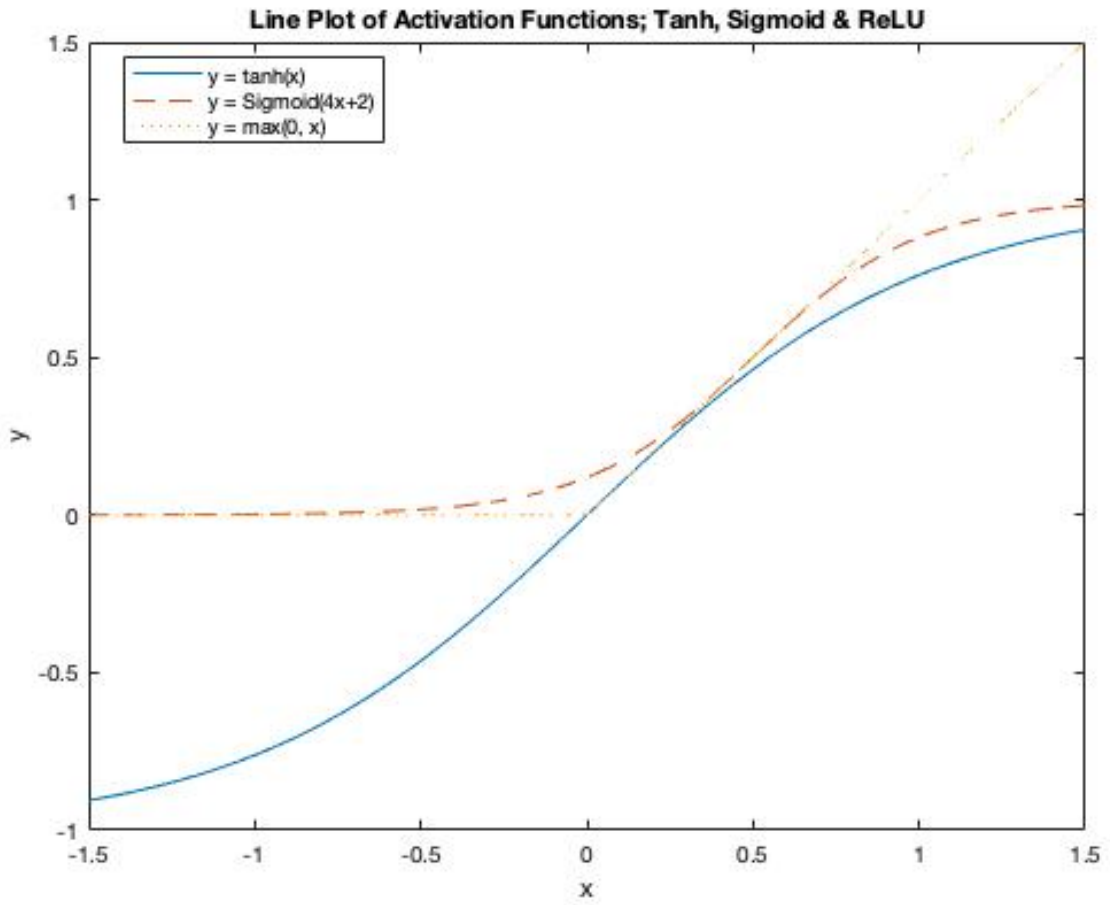


Figure 2.1: *Plot of three activation functions considered for AlexNet [18].*

Since AlexNet [18], CNN's have continued to be developed for Object Detection and are now capable of instance segmentation at a high level of performance. Compared to Image Classification, Object Detection is a much more demanding problem. This causes a rise in model complexity compared to classification as detection requires accurate localisation of objects. Object Localisation is a non trivial task and is usually comprised of two parts: First, proposed object regions must be computed, then, because these proposed regions generally only provide an approximate solution to the localisation problem, they must go through a discrimination and refinement process. One of the first implementations of a CNN capable of fully integrated Object Recognition, Localisation & Detection was OverFeat [34]. OverFeat [34] is an example of a sliding window and multi-scale approach implemented in a CNN and was able to show improved performance in all three aforementioned tasks by leveraging information obtained for all of the tasks for each specific task.

R-CNN [14] was developed shortly after OverFeat [34] and is a multi-stage detector that features a region proposal generator making use of AlexNet [18], followed by a CNN to compute detection. Performance improvement by R-CNN [14] over OverFeat [34] was achieved by applying a CNN to region proposals and by undertaking a process of generic pre-training and domain specific fine tuning for the CNN. One problem with all three of the Object Detection pipelines discussed thus far is that the input to the CNN is fixed because of the fully connected layers in the deeper areas of the network. This means region proposals often have to be cropped or warped in order to be compatible with the subsequent half of the pipeline. Cropping and warping introduces noise, creates a distorted and unrealistic representation of objects, and in the case of cropping, simply removes information which may be strongly inferential.

SPP-Net [16] solves this problem by introducing Spacial Pyramid Pooling on the final convolutional layer of the network in order to adjust the output of the convolutional layers to fit the required input of the first fully connected layer. This means region proposals of any size can be used as the input to SPP-Net [16]. The preservation of information obtained through the use of Spacial Pyramid Pooling allowed for improved performance over R-CNN [14], whilst computational expense for a forward pass of SPP-Net [16] is more than 24 times improved over R-CNN [14].The most significant problems with R-CNN [14] are these; training is multi-stage; training is computationally expensive in terms of time and storage; and forward passes are slow. SPP-Net [16] improves on R-CNN [14] by sharing computation for each proposal, not requiring a forward pass for each. SPP-Net [16] still suffers from the fact it is a multi-stage detection pipeline with large storage needed for training.

Fast R-CNN [13] is a neural network designed for object detection which is capable of training in only one stage, updating all network weights during each backpropogation. It solves the problems suffered by R-CNN [14] and SPP-Net [16] by fully integrating the Object Detection process into a deep neural network, the input to which is simply an image and a set of object proposals. Training for Fast R-CNN [13] is computed in one stage, after initialisation from pre-trained networks. This reduces training time significantly. Fast R-CNN [13] also incorporates an RoI pooling layer. This is the specific case of the Spacial Pyramid Pooling incorporated into SPP-Net [16] where the pyramid height is 1. Fast R-CNN [13] offers improved detection performance over R-CNN [14], is approximately 9 times faster at training time and 213 times faster executing forward passes. So far, all of the aforementioned Object Detection pipelines have one common bottleneck for performance - they all require object region proposals to be generated in pre-processing, and then used as input to some detection algorithm.

Faster R-CNN [28] proposes an end-to-end deep neural network for Object Detection, inte-

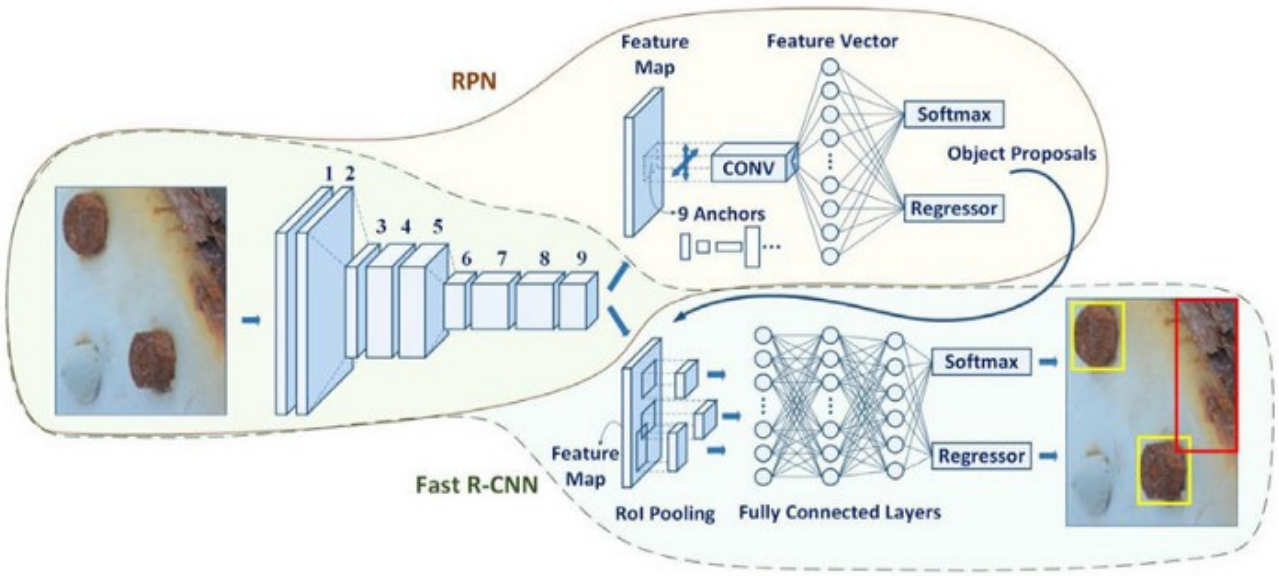


Figure 2.2: *Diagram of Faster-RCNN [28] architecture from Medium. [11]*

grating region proposal and detection into a single CNN (architecture diagram shown in figure 2.2). Part of Faster R-CNN [28] is the Region Proposal Network (RPN), which was designed independently from the Object Detection algorithm in order to generate high quality region proposals. The RPN was trained indvidually and is configured to output a rectangular and axis aligned object proposal region and an 'objectness' score, indicating the likelihood of the rectangular proposal containing a relevant object. Faster R-CNN [28] is the combination of the RPN and Fast R-CNN [13] into one convolutional neural network for Object Detection. Figure 2.2 details the connection between the RPN and Fast R-CNN [13] and displays how the object proposals and 'objectness' score are propagated between the RPN and Fast R-CNN [13] in order to create a fully end-to-end model for Object Detection. Faster R-CNN [28] shows improved Object Detection performance compared to all previously mentioned methods.

Mask R-CNN [15] is an extension and adaptation of Faster R-CNN [28] which includes a branch for determining segmentation for objects parallel to the network branch, designed to determine bounding box locations. Mask R-CNN's [15] extension of Faster R-CNN [28] allows for leveraging of information between region classification, bounding box regression and mask generation in order to achieve improved performance. At the time of publication, Mask R-CNN [15] outperformed all other existing entries across every task. Most recently the use of Hybrid Task Cascade (HTC) [7], which is an effective amalgamation of Mask R-CNN [15] and Cascade R-CNN [6] achieves current state of the art for Instance Segmentation on the COCO dataset [22]. Hybrid Task Cascade [7] is able to achieve this superior performance by communicating information between region classification, mask prediction and semantic segmentation, while leveraging contextual information about an image and the scene it depicts.

2.2 Object Tracking

The task of Object Tracking is that of determining the position of an object, within an image frame, throughout the temporal evolution of a video sequence. This requires identifying the most likely object in a scene to match the object in the scene in the previous video frame. Like Object Detection necessity for Object Tracking is Object Identification. However, where tracking and detection differ is in that trackers may be class agnostic. This means that trackers can leverage temporary information during a specific video sequence about an object in order to ease in its identification in subsequent frames. An example of this may be using the optical trajectory of an object across previous frames to inform on the most likely regions at which an object may appear in the subsequent frame. Another example of sequence specific information being leveraged is in object deformation and occlusion: A deforming or occluded object may be unidentifiable as an object of a certain class (i.e. cup, person, car), however by using the temporal information about the object as it is being deformed or occluded, objects can be accurately identified by analysing the small changes in the object between adjacent frames and using this to determine how the object may appear in the subsequent frame. This temporal information allows for predictions unhindered by the constraints of object classes, giving object trackers the ability to be extremely robust if temporal information provided for individual video sequences is fully exploited.

Traditionally, Object Tackers contain a classification process to discriminate between the object to be tracked and other image regions. Classifiers are trained online using the initialisation of the tracker, usually through user input in the first frame of a video sequence. As previously mentioned, information is learned online about the object in subsequent video frames and then leveraged in order to provide more accurate classification results, and thus increased tracking performance. The use of the Correlation Filter in Visual Object Tracking (VOT), was most successfully introduced by Bolme et al. [5] and provided a lightweight, robust method for VOT. Because VOT requires inferential and robust filters to be trained using the initialisation (usually the initial frame) of a video sequence, which then have the scope to evolve with the temporal evolution of the video sequence and the adaptation of the appearance of the object being tracked. A new type of Correlation Filter was devised by Bolme et al. [5] called Minimum Output Sum of Squared Error filter (MOSSE) which is capable of generating Correlation Filters from the initial frame of a video sequence which satisfy the aforementioned condition. The MOSSE filter tracker provided tracking results robust to varying; illumination; scale; position; orientation; and object deformation. Occlusion can also be identified by the MOSSE filter tracker by examining the peak-to-sidelobe ratio. This enables the tracker to pause tracking procedures while the object is occluded and then resume when the object's appearance is sufficient.

More recently, the use of deep learning in VOT, as it has similarly done for Object Detection, has vastly improved the performance of tracking algorithms. With the use of deep features Correlation Filters can be optimised for specific domains and their performance improved. Torr et al. [36] were the first to implement the use of deep features for Correlation Filtering. By implementing the Correlation Filter learner as layer in a CNN, the learned filters are task specific, not manually designed or learned for a different domain and closely linked to the Correlation Filter. Figure 2.3 details the structure of the tracking model developed by Torr et al. [36], detailing the implementation of offline trained CNN features with dynamic online Correlation Filter adaptation. By incorporating both offline and online learned information,

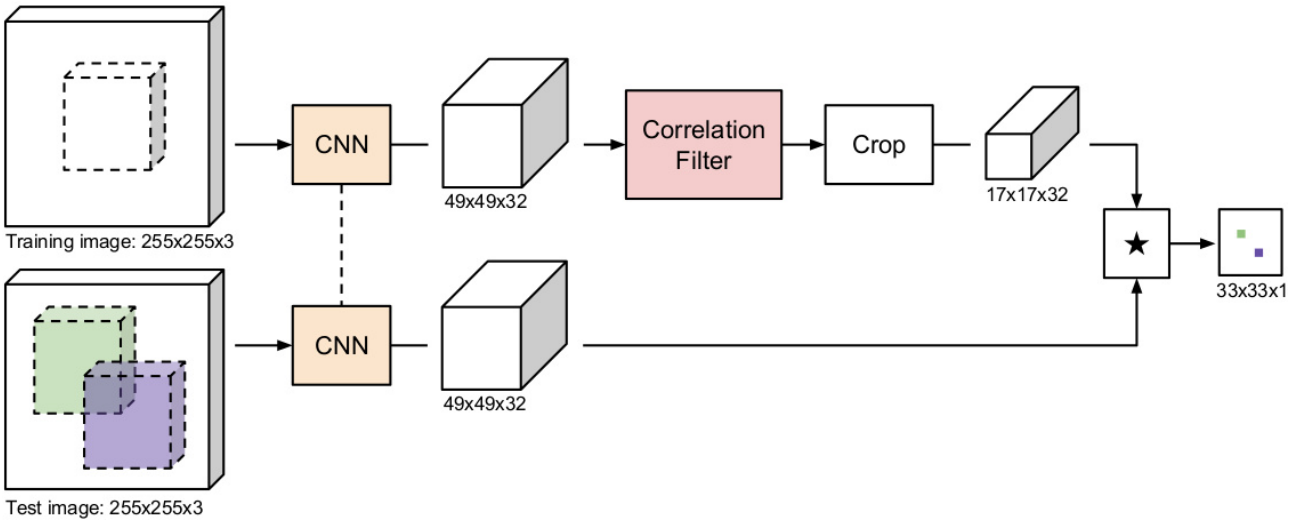


Figure 2.3: *Diagram of CFNet architecture [36].*

the problem of having a little information to initialise the tracker, and the problem of exploiting temporal video information, can both be overcome.

Recently, success has been achieved using a learned similarity function which compares two adjacent video frames instead of discriminative classifiers which are trained online. Bertinetto et al. [2] adopted a Deep Learning approach to the similarity function method and achieved state of the art results at the time of publication with a lightweight and simple model. Figure 2.4 displays the architecture of the Siamese Network trained to estimate a similarity function between two images. In the case of tracking, the two images are an exemplar image and a candidate image of which a score is generated indicating the likelihood of the candidate containing the same object as the exemplar. This can then be applied to image regions, using exhaustive search to determine the location of the object. The offline training of the model allows for operation at frame rates exceeding real time. The deep similarity model achieves this fast test speed by ignoring spaciotemporal cues for each individual video and utilises only knowledge gained from offline learning.

Like for Object Detection, determining candidate image regions is of paramount importance for Object Tracking. The ability to accurately track objects in video sequences is heavily dependent on the ability a tracker has to reliably identify candidate image regions and objects. As per the advancements in Object Detection through the use of Region Proposal Networks (RPN's), Object Tracking is a beneficiary from the use of RPN's also. Li et al. [21] developed a tracker aptly named Siamese-RPN (Siam-RPN) that is an evolution of the previously mentioned implementation of a Siamese Network by Bertinetto et al. [2]. Figure 2.5 displays the network architecture of Siam-RPN and details how the output of the feature extractor (initial CNN) is amalgamated with the RPN which then provides two branches of inference; one for region classification; and the other for bounding box regression. With the use of the Region Proposal Sub-network Siam-RPN at the time of publication achieved leading performance on all Visual Object Tracking challenges (VOT2015, VOT2016 & VOT2017). Although the introduction of the RPN produced a more computational expensive model at test time, Siam-RPN [21] is still able to operate at 160fps, well beyond what is required for real-time Object Tracking.

All previously referenced trackers lack one significant level of inference which has long been a necessity for high performance Object Detectors; generating a binary mask to identify on the

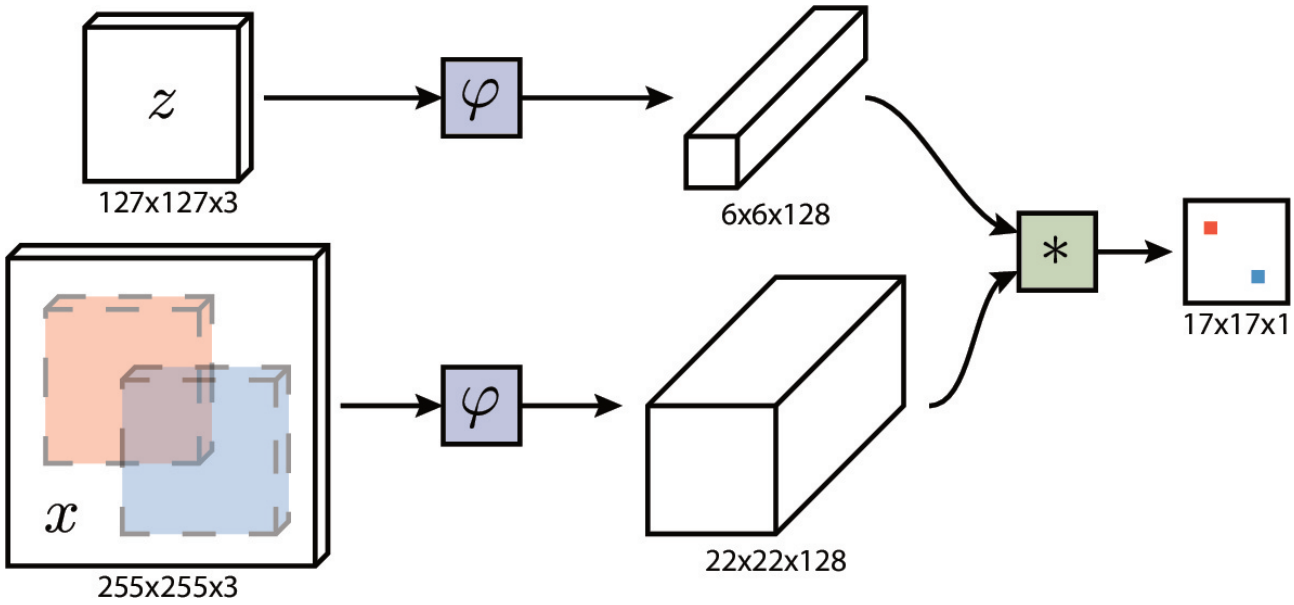


Figure 2.4: *Deep similarity network architecture developed by Bertinetto et al. [2]*

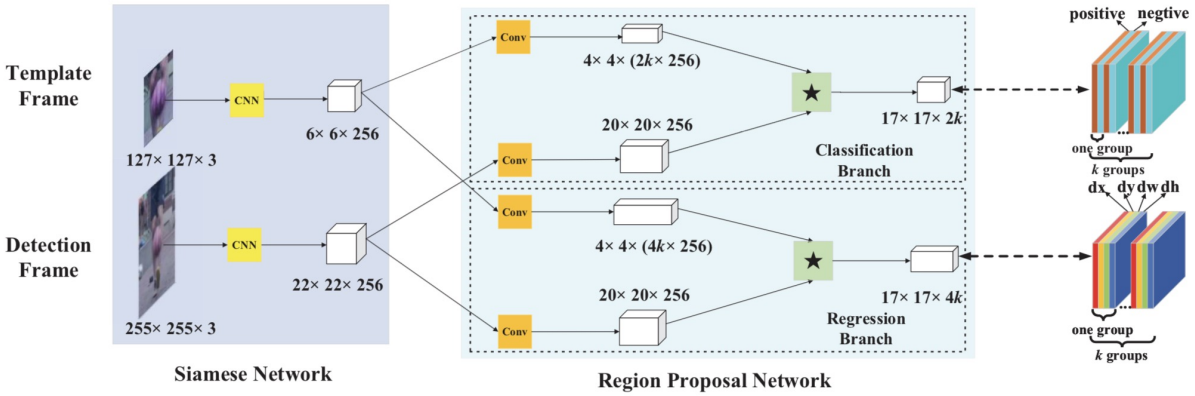


Figure 2.5: *Diagram of Siam-RPN architecture [21].*

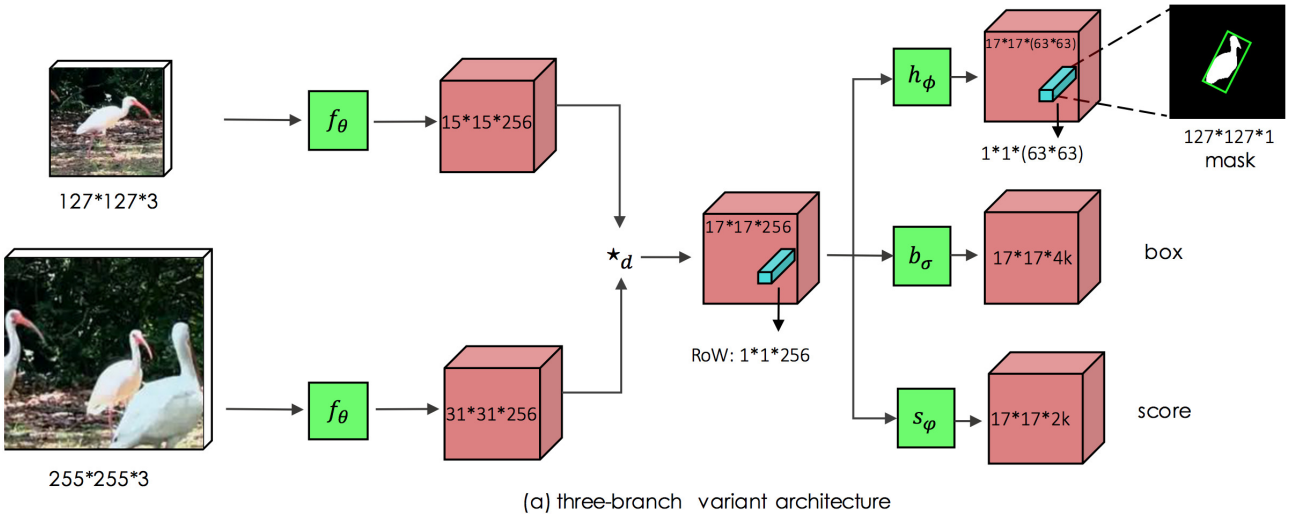


Figure 2.6: *Diagram of SiamMask architecture [42].*

pixel scale, the shape and appearance of the object in any individual image frame. SiamMask, developed by Wang et al. [42] provides a lightweight approach which aims to leverage knowledge gained offline as well as leveraging information about the binary mask of an object to improve accuracy of bounding box location. Figure 2.6 shows the three branch variant (which includes a score) of the SiamMask architecture developed by Wang et al. [42], note the outputs of the three branches where the bounding box is not axis aligned and instead is the box of minimum area which fully encloses the object (or mask). SiamMask [42], at the time of publication, was the only tracker capable of operating online (in real-time) while producing a frame-by-frame binary mask for the tracked object. Generating a mask for a tracked object inadvertently encodes information about the occlusion or deformation of the object over time, which then can be exploited to increase the tracking performance. SiamMask [42] achieved state of the art results for Visual Object Tracking on the VOT2018 dataset as well as achieving high performance on the Video Object Segmentation challenges; DAVIS-2016 & DAVIS-2017.

The current state of the art for Visual Object Tracking is THOR-SiamRPN developed by Sauer et al. [32]. Tracking Holistic Object Representations (THOR) [32] is designed to be implemented on top of Siamese Network trackers in order to diversify the object templates used thereby abstracting information beyond the initialisation of the tracker (ground truth bounding box). Object templates are determined with a diversity measure which operates inside the feature space of the Siamese Network. THOR is able to enhance Siamese Network trackers for only a small speed penalty, it also enables the enhancement of performance of simpler Siamese trackers beyond more complex Siamese trackers which operate at slower speeds.

2.3 Multiple Object Tracking

The problem of Multiple Object Tracking is extremely closely linked to Singular Object Tracking. However, tracking multiple objects simultaneously, especially those of the same class, presents problems unique to Multiple Object Tracking. The main problem specific to Multi-Object Tracking is 'ID switching'. This occurs when the tracker fails to correctly track two or more objects and their corresponding identities switch. For example: Figure 2.7 displays

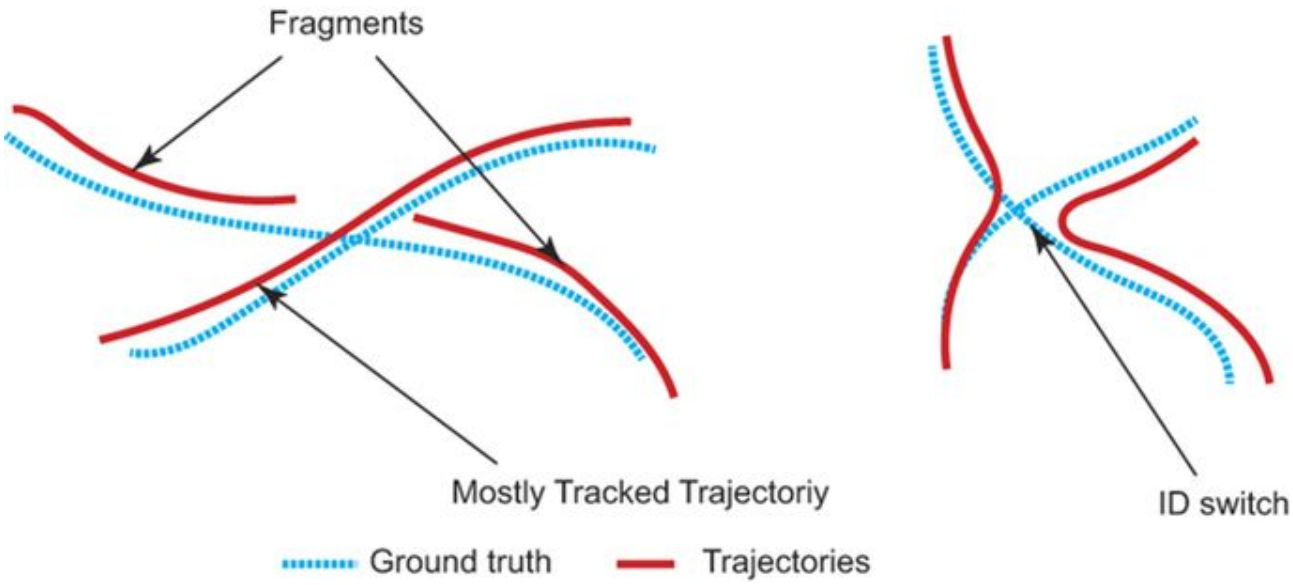


Figure 2.7: *Representation of 'ID switch' in multiple target tracking.*

two non-parallel identical target paths. As the paths cross, a tracker fails to predict trajectory of the targets correctly and fails to represent the objects as having crossed paths. Instead, a tracker would determine that the objects did not cross but 'bounced' away from one another after they coalesced, according to the tracker.

The problem of 'ID switching' can manifest itself in one of three ways; the Hijacking Problem; the Centralization Problem; and the Drifting Problem (Figure 2.8):

- The Hijacking Problem occurs as two (or more) objects approach one another. A tracker can cease to identify one of the objects completely and infer the remaining object as being two objects. This is shown on the far left of figure 2.8 where the blue object 'hijacks' the predicted identity of the red object while retaining its blue identity.

- The Centralization Problem occurs identically to a generic ID switch. As objects approach one another and separate the inferred identities of the objects can be swapped. This is shown in the centre of figure 2.8 where the blue and red objects both have the opposite colour boxes as they separate.

- The Drifting Problem occurs due to the motion of objects. Many trackers make use of a motion model in order to determine an image region with a high probability of containing a specific object being tracked. However, if an object quickly changes its velocity from the perspective of a visual sensor, the region of high likelihood suggested by the motion model does not match with the actual location of the object in the preceding video frame. This is displayed visually on the far right of figure 2.8 where the green object's motion is from left to right and then from right to left. However, because of the initial left to right movement the suggested region from the motion model does not represent the true region containing the object as it moves in the opposite direction from its initial movement.

A traditional approach to Multiple Object Tracking is through the use of a tracking by detection technique called Multiple Hypothesis Tracking (MHT). MHT was developed by Reid [26] and uses measurements of a target's location to determine the likelihood that the measurement is due to a known target (and likelihood of which specific target from the set of known targets),

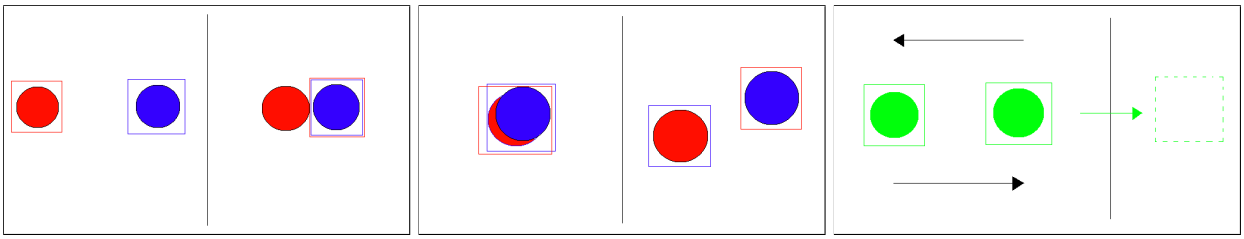


Figure 2.8: *Rudimentary representations of Hacking Problem (Left), Centralization Problem (Centre) & Drifting Problem (Right). Objects are displayed as circles, and their inferred identities as bounding boxes.*

the likelihood that the measurement is from a brand new target and the likelihood the measurement is false. MHT [26] is however a computationally slow and memory exhaustive method and therefore has been adopted to a greater degree by the RADAR community [4] (where the rate of measurement is much slower compared with vision) than the vision community, where the computational resources required to effectively use the algorithm would be significantly greater, even if the requirement of computing reliable detections could be satisfied. Another traditional approach to multiple target tracking is Joint Probabilistic Data Association Filtering (JPDAF) [12] which uses the posterior probability in order to determine, from a set of candidate measurements, a single measurement which represents the same target as for previous measurements. Additional measurements recorded are assumed to be Poisson distributed statistical clutter, however, the process of determining which target each measurement corresponds to is computed for each measurement. This means a joint posterior association probability is calculated for each target within Poisson clutter. Another filtering technique used for tracking multiple targets is the Probability Hypothesis Density (PHD) filtering [24]. PHD filtering has several approximations such as Gaussian Mixture PHD filtering [37] and Sequential Monte Carlo PHD filtering [38]. PHD filtering, like the other previously mentioned filtering methods, can still perform with false measurements and misdetections.

More recently, the use of SORT (Simple Online and Realtime Tracking) [3], a tracking-by-detection algorithm which uses only a simple combination of two traditional techniques; the Kalman Filter [17] and Hungarian algorithm [19]. SORT [3] utilises only the bounding box location and size as input and excludes any visual data. One desirable trait of Multiple Object Trackers is the ability to operate in real time in order to reduce model complexity and thus inference speed. SORT [3] has no architecture capable of object re-identification. This means SORT [3] cannot overcome problems related to occlusion. However, SORT [3] does achieve state of the art results on the 2D-MOT2015 [20] Multiple Object Tracking Task while retaining an operation speed of 260Hz. One tangential conclusion drawn from SORT [3] is that the performance of trackers which use detection techniques is heavily dependent on the underlying detection performance, with a reported maximum 18.9% difference in tracking performance of SORT [3] only through the use of different detectors. SORT [3] has since been adapted to utilise a Deep Association Metric (Deep-SORT) [43] in order to exploit visual data through an offline trained CNN. Deep-SORT [43] allows for the re-identification of occluded objects through the use of the learned association metric at a rate and reduces the frequency of ID switches by 45% compared to SORT [3] while continuing to operate at above real time frame rates.

Recently, Deep Learning has been more extensively adopted for Multiple Object Tracking. The use of Deep Affinity Network (DAN) [35] provides a novel approach to Multiple Object Tracking by introducing a tracking by detection model with a learned feature extractor and affinity

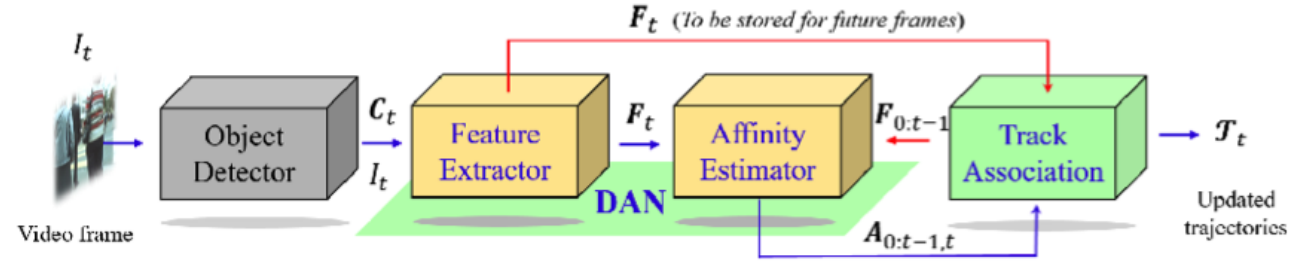


Figure 2.9: *Diagram of Deep Affinity Network architecture. [35]*

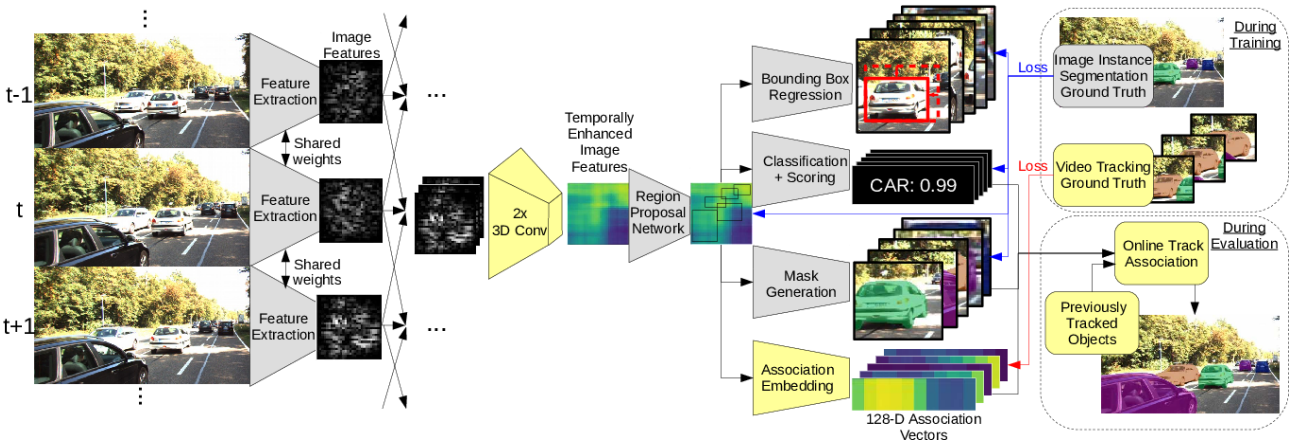


Figure 2.10: *Diagram of Track R-CNN [39] architecture. Additions to Mask R-CNN [15] are highlighted in yellow.*

estimator which are used to associate objects between any two video frames. Figure 2.9 displays the Deep Affinity Network pipeline for Multiple Object Tracking. A video frame is fed into an object detector which outputs object centres which are then input to the DAN, along with the video frame, which extracts features from each object and uses these to compare to a previously stored feature codebook for each object. This allows DAN [35] to manage full occlusions over long time periods a well as objects leaving and re-entering frame. DAN [35] achieves state of the art performance on 2D-MOT2015 [20] and other MOT challenges. All of the previously mentioned Multiple Object Trackers determine object locations using only bounding box representations and fail to exploit information provided by object mask generation. Voigtlaender et al. [39] present a fully end-to-end deep learning approach which integrates Object Detection, Multiple Object Tracking and Object Segmentation called Track R-CNN. Track R-CNN is built around object detection network Mask R-CNN [15] and extends it by 3D convolutions in order to capture temporal information of a video sequence. Track R-CNN [39] also adds an association branch used to compare 128 dimension feature vectors of detected objects in order to compare them for tracking. Figure 2.10 shows the network architecture for Track R-CNN and its extensions from Mask R-CNN [15] (in yellow). Track R-CNN [39] represents a baseline in Multiple Object Tracking and Segmentation and, to my knowledge, the only deep learning approach in Multiple Object Tracking to generate masks for tracked objects extending beyond conventional bounding box annotation of tracked objects.

Chapter 3

Methodology and Implementation

The chapter explains my implementations for solving the problems stated in section 1.2 and 2.3. Implementations are made across Object Detection and Multiple Object Tracking, with the most significant implementation made to solve the Multiple Object Tracking problem. The methods proposed in the following sections are driven by the relevant research discussed in chapter 2 according to the time constraints of this project. Implementations made for this project range from Object Detection Refinement to a full Multiple Object Tracking Pipeline capable of object re-identification through the use of data association of object representations.

3.1 Object Detection

All of the implementations discussed in this paper follow a tracking by detection pipeline. This means that the tracking performance is highly dependent on the detection performance. In order for the high performance Multiple Object Tracker to perform at its potential, the supporting Object Detector must be able to detect all relevant objects in every frame of a video sequence. While this may not be possible without the use of a tailored dataset for a particular domain (i.e. autonomous driving), a high detection performance can still allow for reasonably high tracking performance [3]. As previously mentioned, Mask R-CNN [15] is a CNN designed for object recognition and localisation by Facebook AI Research (FAIR) [29]. The original model [1] was trained on the Microsoft COCO dataset [22] and tested on all three tracks of the COCO suite of challenges. Mask R-CNN was chosen to be the principal Object Detector for the tracking-by detection pipelines discussed in this paper. Mask R-CNN provides state of the art detection performance on a diverse set of object classes. In a method discussed in the succeeding section, SiamMask [42] is used as a class agnostic detector for a Psuedo-Multiple Object Tracking baseline. This involves SiamMask [42] being initialised several times, once for each of the multiple objects to be tracked. Each initialisation then runs in parallel in order to track individual objects by tracking a single object across multiple instances. This type of tracking, however, is not true Multiple Object Tracking as no knowledge of the other objects is used in the computation of the tracked location of any specific object.

In order to refine detection results a minimum threshold was integrated onto the confidence for each detection. This reduces false detections with the payoff of increasing likelihood of increasing the probability that a relevant object goes undetected. The confidence threshold was implemented in post processing to Mask R-CNN [15]. Another refinement of detection

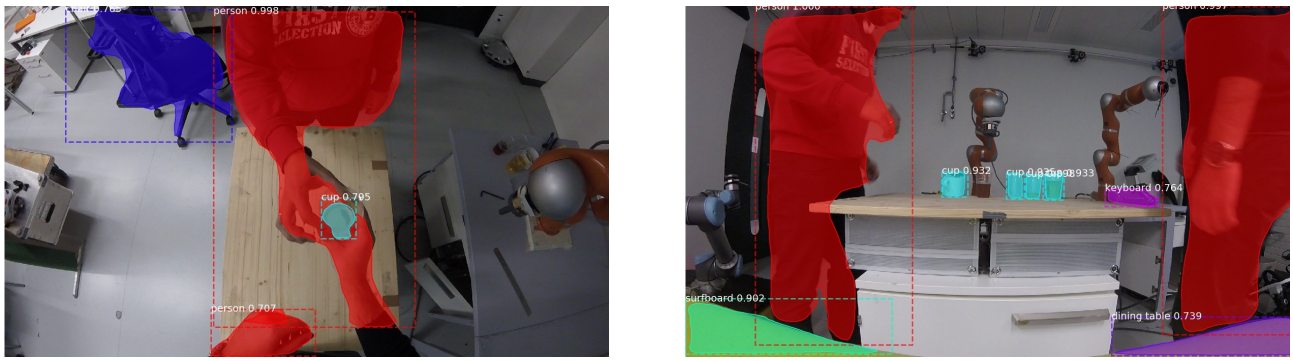


Figure 3.1: *Class wise segmented video frames from project dataset, fed through Mask R-CNN [15].*

results in post processing was a refinement of object classes. In practice, typically, the relevant objects to be tracked in a scene are of only a few different classes. This means unnecessary detection results must be removed in order to preserve objects of relevant class for a particular scene or domain. This refinement process removes all detection results not specified through manual input and preserves only those detections of relevant class.

3.2 Single and Multiple Object Tracking

3.2.1 Preliminary Methods

3.2.1.1 Mask R-CNN: Class-wise Segmentation

As per chapter 2, Mask R-CNN [15] is a state of the art object detector developed by a team at Facebook AI Research (FAIR) [29] who released open source code for the project [1] and has been used in several subsidiary projects since its inception [27][40][45][10][33][46][9][25]. Mask R-CNN [15], as created by FAIR [29], was trained on the Microsoft COCO dataset [22].

In order for multiple objects to be tracked, knowledge of the set of tracked objects must be used to temporally enforce object identities. The original Mask R-CNN implementation [15] does have the architecture to enable the generation of semantic instance segmentation, individually identifying each detected object. However, it does not have the capability to enforce segmentation temporally. In other words, Mask R-CNN [15] cannot identify objects between two video frames as being corresponding objects and therefore cannot be used to track objects in video. For video input, Mask R-CNN [15] can generate class wise segmentation, which is shown on a dataset created for this project in figure 3.1. A research group led by Chui Chee Kong [8] implemented Mask R-CNN [15] to identify and segment surgical robotic arms and rings which the robots use to execute tasks [45]. Kong's implementation [45], which was fine tuned on a specifically designed dataset, generates class-by-class video object segmentation. Kong's group [8] does generate instance segmentation of detected objects, however, this is only for individual video frames and therefore, means Kong's implementation [45] does not solve the proposed video object tracking task.

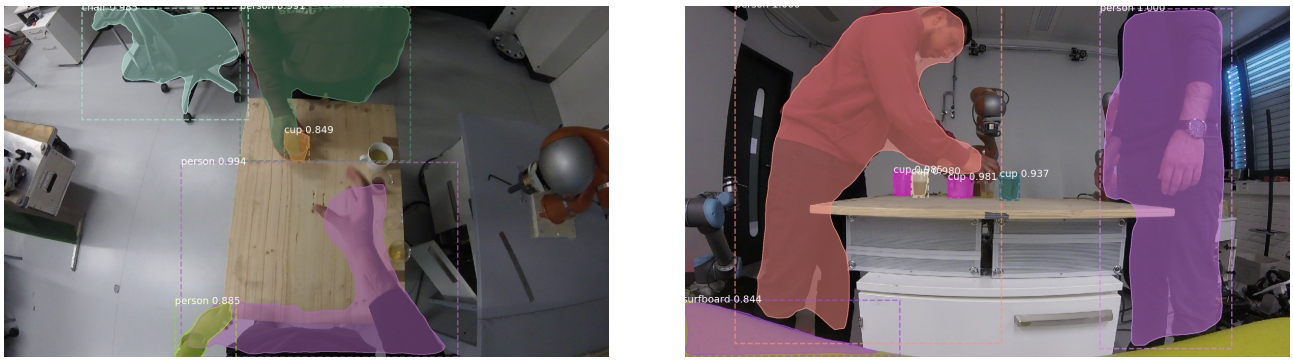


Figure 3.2: *Instance segmented video frames from project dataset, fed through my implementation of Mask R-CNN [15].*

3.2.1.2 Mask R-CNN: Loosely Enforced Temporal Instance Segmentation

My first implementation of Mask R-CNN [15] aims to solve both the video object segmentation (VOS) and video object tracking (VOT) tasks by generating semantic instance segmentation for detected objects. The problem of temporally enforcing instance segmentation, which effects both Kong's [45] and the original implementation [15] when acting as trackers, is overcome during post processing. By using information about the pixel location of bounding box centres from the previous video frame, the bounding box generated in the subsequent frame most likely to correspond to the same instance can be inferred. More specifically, the mask colour for each detected object is determined by computing the closest instance of the same class detected in the previous frame, then assigning the new detected instance the same colour mask. The distance between detected instances across frames is calculated by determining the distance between their respective bounding box centres in the image plane. The initial mask colours for each detected object are generated by randomly sampling in the RGB space while ensuring each generated colour is unique. Figure 3.2 displays two frames with instance segmentation produced using my implementation of Mask R-CNN to track detected objects. Appendix .1 displays ten sequential video frames with instance segmentation applied to track objects between frames, generated using my implementation of Mask R-CNN [15].

3.2.1.3 SiamMask: Class Agnostic Tracking by Detection

SiamMask [42] is a current state of the art real time object tracker and is able to solve both video object segmentation and video object tracking (semi-supervised) tasks. The original implementation [23] [42] was trained on the ImageNet-VID [30], Youtube-VOS [44] and Microsoft COCO datasets [22]. Open source code has been released for the original SiamMask implementation [41].

Unlike Mask R-CNN [15], SiamMask [42] is class agnostic and solves the object tracking problem without having to learn a set of image classes to refer to when detecting objects in the image. This means SiamMask [42] has comparatively less computational cost per tracked object than Mask R-CNN, and can also track objects outside of the domain of the classes learned by Mask R-CNN [15]. The original implementation of SiamMask [41] tracks just a single object per inference but can execute a rudimentary form of multi-object tracking by computing several inferences with different initialisations for each object to be tracked and stacking the frame-by-frame results. The video frames shown in figure 3.3 include mask and non-axis aligned

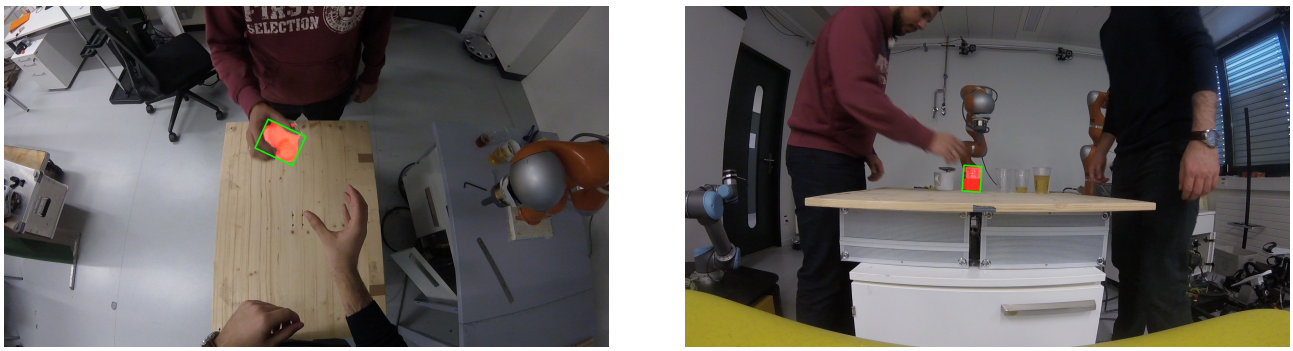


Figure 3.3: *Qualitative results of video object tracking using original SiamMask implementation [42], executed on this project's dataset.*

bounding box results from SiamMask [42]. Because the original implementation of SiamMask [42] operates with agnosticism with respect to object classes, the object to be tracked must be manually initialised via axis aligned region of interest (ROI). This means SiamMask is not fully automated and tracks objects in a semi-supervised fashion, requiring manual initialisation for every object to be tracked.

3.2.1.4 SiamMask: Automated Pseudo-Multiple Object Tracking

As described in section 3.1, SiamMask [42] was adapted in order to track multiple objects per frame in parallel with a single initialisation for each object executed simultaneously. These bounding box initialisations are generated using Mask R-CNN [15]. Detections are generated for the first frame of a video sequence and the bounding boxes are used to initialise SiamMask [42] for each object. This allows SiamMask to track multiple objects without the need for manual initialisation. Figure 3.4 displays two video frames post tracking results with boxes and masks overlayed. The left image in figure 3.4 shows the detections generated by Mask R-CNN [15] of which the bounding boxes are used for SiamMask initialisation. The right image in figure 3.4 shows the tracking results of SiamMask several video frames post initialisation.

All of the previously mentioned methods lack one feature necessary for tracking to be of high performance; object re-identification. While an object is occluded or exists out of view of the camera, tracking cannot be computed for this object. However, when this object returns back to the view of the camera, it should be recognised as the same object it was previously identified as.

3.2.1.5 Mask R-CNN: Object Re-identification with Temporal Backlog

The implementation of Mask R-CNN [15] described in section 3.2.1.2 was adapted in order to store a temporal backlog of bounding box location, mask color and class identity (i.e. cup, person, table). This then enables for object re-identification with a class discriminator. This allows an object to be occluded, move out of frame, or go undetected and then be re-identified based on the object's class. For example, figure 3.5 shows object re-identification through the use of the temporal backlog. The left hand image in figure 3.5 displays a person holding a cup after beginning to rotate it in their reference frame. The right image in figure 3.5 displays the frame one time instance after the left image. In the right image, because of it's rotation, the

Figure 3.4: *Qualitative results of video object tracking of pseudo MOT with SiamMask [42], executed on this project's dataset. Left image shows initial detections generated by Mask R-CNN [15] passed into SiamMask [42].*

cup is not detected by Mask R-CNN. Finally, in the bottom image in figure 3.5 the cup has been re-identified using the temporal backlog after it has been rotated back toward its original orientation. Because the video sequence used to generate figure 3.5 contains only two relevant objects; a cup and a person, if either of these objects is occluded or not detected for a single or group of frames, object identities of either can be recovered easily by matching the class of the objects (i.e. if the cup is not detected, when it is detected at a subsequent time instance, because it is the only cup in the scene, it will always be recovered with the same identity). This method is only capable of object re-identification when each object in the scene is of a different class with respect to the detector, and only one object occupies a specific class. This means the representation, and thus identifier for each object, needs to be its unique class ID. In order to have a robust Multiple Object Tracker, multiple instances of objects of the same class (i.e. multiple cups) must be uniquely identifiable. This means that objects can be identified after occlusion, misdetection and if they move out of frame, regardless of the number of objects of the same class.

3.2.2 Mask R-CNN: Leveraging Information from Segmentation to Generate Object Representations

The main implementation of this paper is that of a multi-object tracking by detection through data association method. This tracker uses the detection output of Mask R-CNN [15], specifically the mask, in order to generate a representation for each detected object in each frame. Object representations are generated sequentially, frame-by-frame, with the identity of each object in all frames apart from the initial frame, determined by association using a similarity measure. This method aims to solve the problem of object re-identification as well as remaining robust to the three common problems in Multiple Object Tracking highlighted in figure 2.8.

3.2.2.1 Generating Object Representations

As previously mentioned, object representations are generated for every object across every frame in a video sequence. In order to generate these representations information from the mask is utilised. The following process is how object representations are generated for each

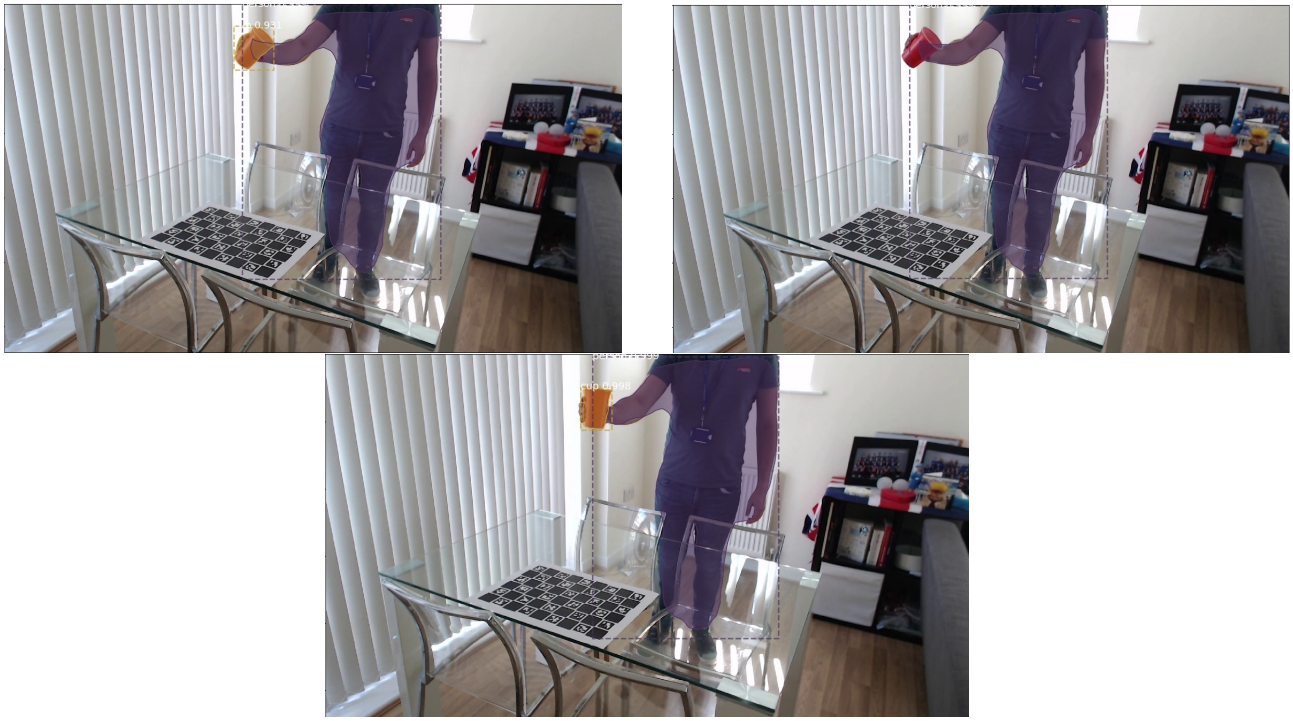


Figure 3.5: *Qualitative results of Mask R-CNN [15] with temporal backlog tracking. Left image shows frame where cup begins manual rotation. Right image shows frame where cup is not detected by Mask R-CNN [15] due to rotation. Bottom image shows re-identification of cup after its rotation has allowed it to become detectable again at a later time instance.*

object:

1. Every image element apart from those identified by the mask (i.e. every pixel apart from the pixels the object is occupying) are reduced to 0 across all three RGB color channels to generate an initial representation.

2. The representation is summed across color channels to generate a single plane array.

3. The representation is cropped such that both image dimensions are of size equal to the difference between the maximum and the minimum pixel indices which are non zero in those specific dimensions. In other words, the representation is cropped in order have minimum size while still containing every object occupied pixel. This is the final representation of an object, which is then stored for association.

Figure 3.6 displays some generated object representations which have been converted to single channel (grayscale) images. As described, each representation is a single channel image with only pixels occupied by objects as non zero. All representations in figure 3.6 are displayed at their original aspect ratios.

3.2.2.2 Comparing Object Representations

In order to match objects in each frame through data association, object representations must be compared and the similarity between two representations measured. The following process is used to determine the error between two object representations:

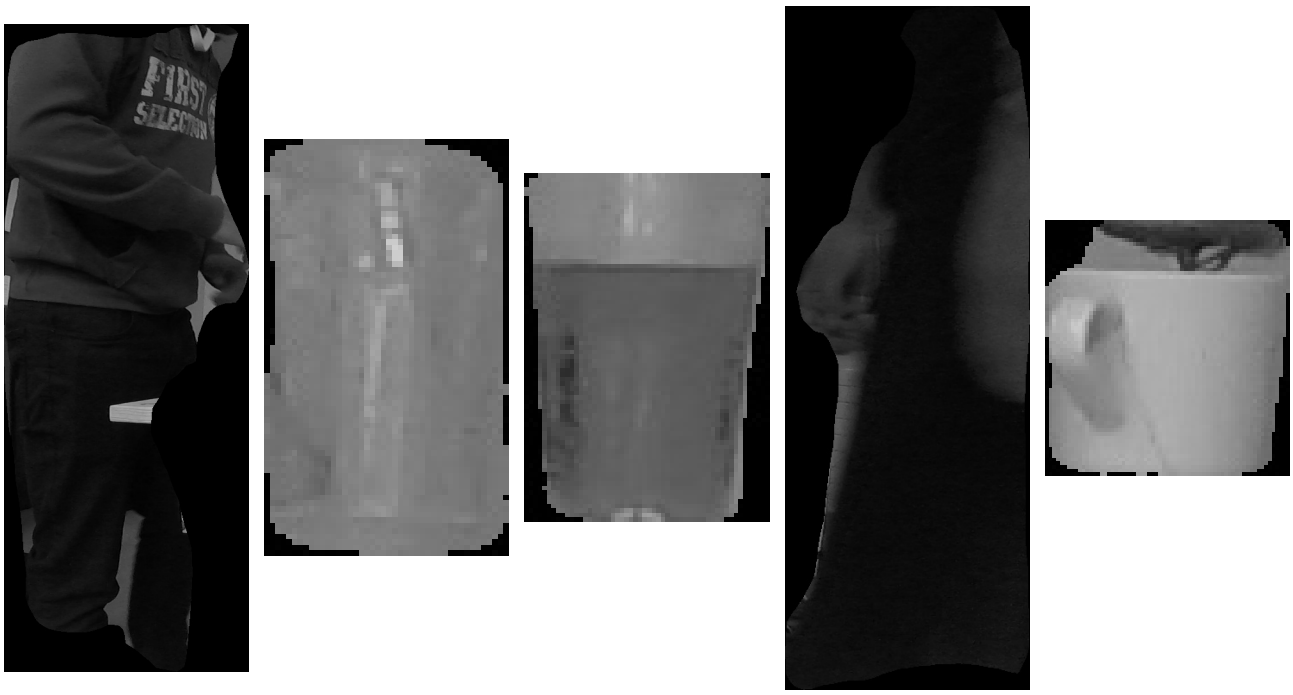


Figure 3.6: *Five example generated object representations, converted to grayscale. From left to right object classes are as follows; person, cup, cup, person, cup.*

1. Object representations are first reshaped and their values scaled so that both representations have matching dimensionalities. This is computed by up-scaling the representation with fewer pixels and, using bicubic interpolation, either upsampling or downsampling the rows and columns in the smaller representation to match that of the larger representations. In the case where representations have the same number of pixels but not equal dimensionalities, the representation which identifies the object in the current video frame is preserved, and the stored representation is reshaped and resampled.

2. The intersection area and union area of the non zero pixels in each representation is then computed. An error coefficient is then calculated according to equation 3.1,

$$coefficient = \frac{union}{intersection} \tag{3.1}$$

   Where the **union** represents the number of pixels inside the union of the two representations and **intersection** represents the number of pixels inside the intersection of the two representations. The **coefficient** generated always holds a value of at least 1.

3. The mean pixel wise (and color wise) absolute error is then calculated for all of the pixels inside the intersection of the two representations and an error shown in equation 3.2

$$error = coefficient \cdot \frac{\sum_{i=1}^{intersection} |a_i - b_i|}{3 \cdot intersection} \tag{3.2}$$

   Where **coefficient** is the value generated from equation 3.1, **intersection** is the number of pixels inside the intersection of the two representations, **a** and **b** represent the set of pixel values for pixels inside the intersection of the two representations. Because the coefficient generated in 3.1 outputs a value of at least 1, with a high value for if the union is much larger than the intersection of the two representations and 1 if the intersection

matches the union. This means the coefficient acts as an amplifier for the average pixel wise error, enabling **error** to propose an error for each representation which utilises not only both the pixel wise errors of the raw image data inside the intersection of two representations, but also with the amplification coefficient generated using information of the representation's shape. Because the denominator of equation 3.2 is the the intersection tripled (for each color channel), the error indicates the mean error per pixel, per color channel in the intersection between the two representations, which means the error is completely independent of representation size.

3.2.2.3 Algorithmic Procedure

When a video is inputted to the tracker, a random set of unique colors is initialised for all of the relevant objects in the scene, and representations for these objects are stored with their corresponding color. For all subsequent frames, after detections are computed, each new object undergoes a process to determine its best color. For each new object, its representation is generated. Then this representation is compared to every other object representation of the same class (i.e. cup, person) in every previous video frame. For each previous frame, the representation with the lowest error for that frame is chosen stored. The mode of these stored colors is then taken and stored as the proposed color along with the lowest error for the representation associated with the mode color in the previous frames. Once a proposed color for each object in the new frame is computed, if all proposed colors are unique, the objects are assigned these proposed colors. If any of the proposed colors match, the object with the proposed color of lowest minimum error (stored after taking the mode) is assigned that color and the process of iterating through all previous frames is computed again for this object. However, the previously denied proposed color is removed from the pool of possible proposed colors.

If an object fails to obtain a color after many iterations of this process, the object is considered to be a new object, and is assigned a new unique randomly generated color. This ensures all colors in frame are unique, and no two instances occupy the same identity and because of the nature of tracking by detection, no object is assigned more than one identity. Because the object representation - color codebook is generated online, the process of representation matching should become more robust over time and provide a better chance at object re-identification. However, because in the first few frames of a video sequence, there are few previous frames across which to take the mode color and if malleable objects (like people or animals) generate vastly different representations between frames, this can cause ID switches before a larger and more robust dictionary of representations is accumulated after significant time. Figure 3.7 displays the first two frames of a video sequence where the ID of two objects have switched due to their being only insufficient frames over which to determine the best color.

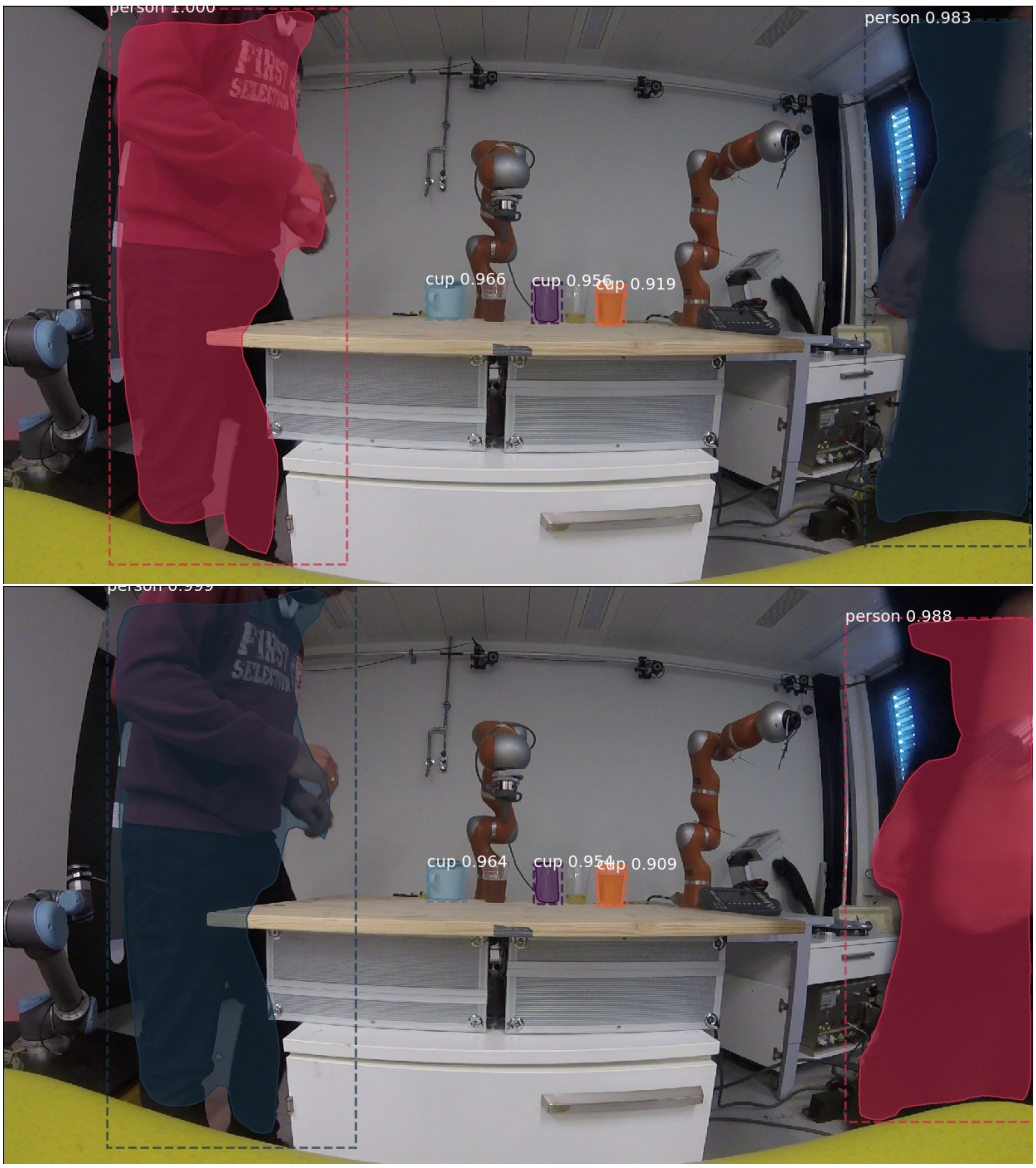


Figure 3.7: *Qualitative results of ID switch on first (above) and second (below) frame of a video sequence due to lack of stored object representations.*

Chapter 4

Analysis and Validation

This chapter of the report discusses the strengths and limitations of the implementations discussed in chapter 3. Critical analysis of all implementations is conducted with reference to the output of the methods. The implementation's ability to solve the problems described in sections 1.2 and 2.3 is discussed with reference to the physical outputs of the implementation. Methods are compared and contrasted according to their effectiveness and in which areas of they excel most significantly. Implementations are also evaluated on their progress toward the aims of this project listed in section 1.1.

4.1 Object Detection Refinement

All of the object tracking pipelines discussed in this paper are tracking by detection. This means the tracking algorithms work on top of the detector as video frames are sequentially fed into the detector and detection results are output. As previously stated, Mask R-CNN [15] outputs results in four different categories. For each detected object, Mask R-CNN [15] generates; an axis aligned bounding box indication a region of interest enclosing an object; a Boolean/binary array (mask) indicating the image pixels occupied by an object; a class ID (car, house, person, tree etc); and a confidence score between 0 and 1.

The first refinement executed on tracking results was to abstract only the relevant detections for the purposes of Object Tracking in any specific domain. Architecture which removes detections of irrelevant classes and preserves those of classes required for tracking was integrated into Mask R-CNN [15] in post processing. The dataset used in development of the implementations in this paper contains only people and cups which are of relevance to Object Tracking, no other class of object have significant movement in the image frame to constitute tracking being necessary to describe their existence within the video. Figure 4.1 displays two identical video frames, the left frame is displayed with all detections output by Mask R-CNN [15]. The second frame shows only refined detections of relevant objects (cups and people). In the unrefined frame, out of the three unnecessary detections; surfboard, clock and oven, both the surfboard and oven are misdetections and are successfully removed by class refinement. The other undesired detection; clock is also removed successfully. In this case the clock (watch) may be inferential for purposes not discussed in this paper, like action recognition. However, the watch does not bare significance to the scene in question and it's detection rightfully omitted. This detection refinement was implemented in order to remove clutter from the detections and abstract only

Figure 4.1: *Results of class and confidence refinement on Mask R-CNN [15]. Left image shows detections without refinement. Right image shows detections after refinement for only people and cups in identical frames in the same scene. Bottom image shows the same scene with both the class refinement and confidence threshold at a value of 0.9.*

the detections from relevant objects. There is only one possible downside to this refinement process, and that is if classes in the object detector are similar (i.e motorbike and bicycle). If classes are similar, a misdetection could occur causing a relevant object to be detected as belonging to a class not considered relevant. The implications of this for an Object Tracker which uses class information to match objects, like the main implementation of this project, would be that it loses track of the object until it is re-identified as being in its original object class.

The second refinement to the object detection process was to introduce a confidence threshold. This enabled detections with a confidence lower than the threshold to be ignored by the tracker, effectively aiming to reduce the number of false detections generated. Figure 4.2 displays the results of the confidence threshold on identical video frames. The left image in figure 4.2 displays the detections of Mask R-CNN without the threshold active. The right image of figure 4.2 displays the same scene with the confidence threshold active at 0.9. The object detected as a cup on the piece of furniture in the background of the scene in the left image of figure 4.2 is a misdetection. In the right hand frame in figure 4.2, the object has not been detected due to the threshold on the confidence at 0.9. The bottom image in Figure 4.1 however, displays an identical scene as the other two images in the figure but with the threshold active at 0.9, one of the cups previously detected on the table is not detected. This indicates thresholding the confidence of detections comes at a price; increasing the threshold decreases the probability incorrect detections will be generated while increasing the probability of not detecting a relevant object in the scene. The two scenes in figures 4.1 and 4.2 are different and the effect of the threshold has a positive effect on detections in the scene in figure 4.2 and a negative effect on detections in 4.1. This indicated the effect of the threshold may be highly domain specific as

Figure 4.2: *Results of confidence threshold at 0.9 and class refinement with Mask R-CNN [15]. Left image shows detections without threshold. Right image shows detections after threshold.*

well as the ideal value for the threshold to take.

4.2 Single and Multiple Object Tracking

4.2.1 Preliminaries

4.2.1.1 Class-wise Segmentation

Results of class-wise segmentation can be seen in figure 3.1. This is a very rudimentary form of class by class tracking. If all objects in a frame are of different classes, each object can be uniquely identified by its class. However, for practical applications like autonomous driving, many of the objects to be tracked will be of the same class, and therefore cannot be differentiated between using class ID. Also, using simply the class ID as the identity of objects, even if there is only a single instance of each class in frame at any one time, if an object leaves frame, and a different object of the same class appears in frame some time after, it will be assigned the same ID as the original object. Overall, this method only works if all objects are of the same class with respect to the detector and if none of the objects are ever replaced in frame by a new object of the same class.

4.2.1.2 Loosely Enforced Temporal Instance Segmentation

This method utilises a bounding box distance measure by calculating the distance between bounding boxes of objects of the same class between frames. In other words, which bounding box in the previous frame for an object of class ID matching the new object is the closest to the position in the image plane of the new object? Although it is very simplistic, this method performs well for objects which are never occluded; are detected in every frame of a video sequence; and never leave frame. Figure 4.4 displays 10 sequential frames of video with tracking results. Both the people and the cups detected at the first frame remain with the ID they they are assigned. For the tracked objects in figure 4.4 there are no occlusions; cases of leaving and re-entering frame; or detection faults. These are the conditions required for this method to work effectively. However, in practice, occlusion and other perturbations cause these conditions to be broken. Figure 4.3 displays tracking result where an ID switch occurs. In the

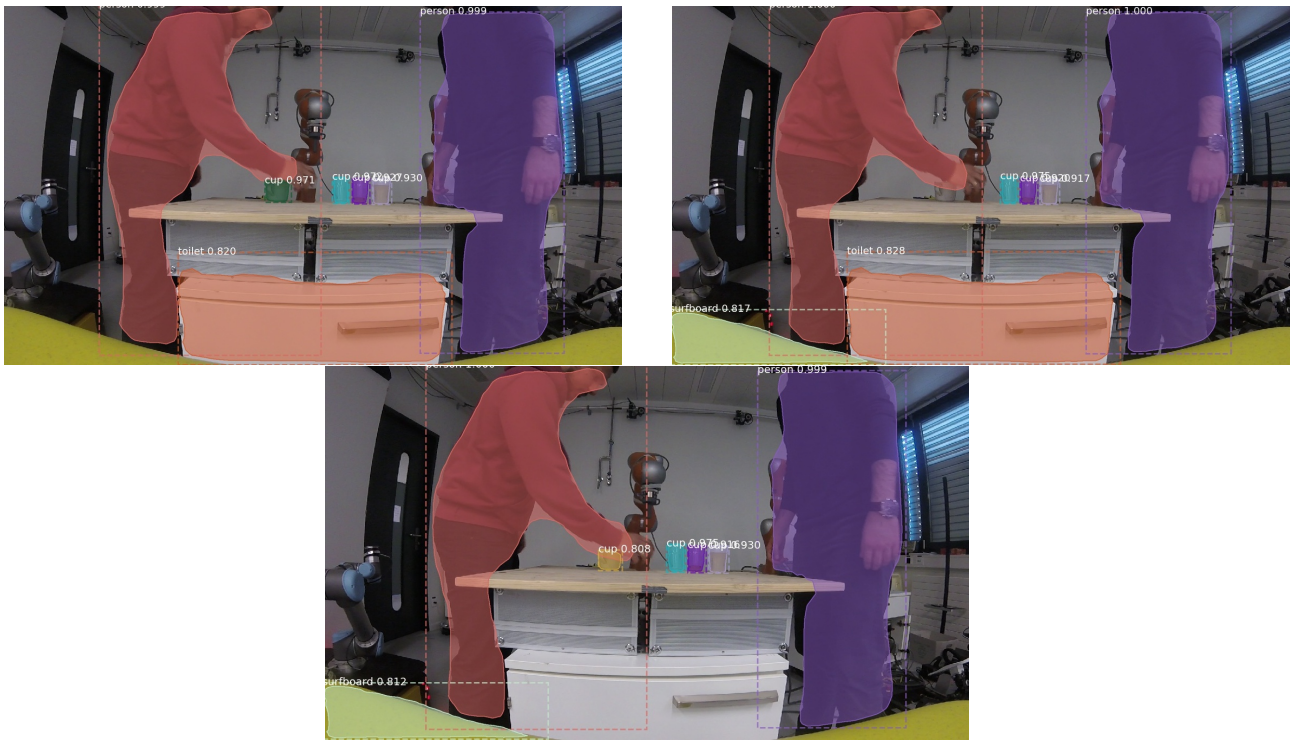


Figure 4.3: *3 sequential frames showing results of method 3.2.1.2 with ID switch due to occlusion. Temporal order of frames from initial to final: Left, right, bottom.*

left image shown in figure 4.3 the cup which is partially occluded by the person on the left of the frame is not detected in the subsequent frame. Finally, in the bottom image of figure 4.3 the cup is detected once again. However, its previous identity is not recovered because this method can only refer to the previous frame to determine bounding box distances. Because the cup was not detected for a single frame, the tracker generated an ID for the cup as if it was a completely new object in the scene. This method cannot, in any case, solve any problems associated with occlusion, moving in and out of frame, or detection issues as it only has the architecture to use the previous one frame to use to determine the object identities for the next frame. Something this method is also heavily dependent upon is frame rate. For slow frame rates, its possible that two objects with trajectories toward one another can encounter an ID switch if the objects paths cross. Its possible for two crossing objects to appear in positions closer to their counterparts position rather than their own position in the previous frame. With higher frame rates, this can be avoided by sampling object positions more frequently, so that the positions of the objects has a smaller change between frames and thus the tracker less likely to generate tracking results with ID switches.

4.2.1.3 Pseudo-Multiple Object Tracking with SiamMask [42]

The method described in section 3.2.1.4 is an extension and adaptation of SiamMask [42] where by multiple objects are tracked using bounding box initialisation generated using Mask R-CNN [15]. The bounding boxes of the initial detection are then used as initialisation to SiamMask [42]. This method is then able to track single object instances in parallel to mimic true multiple object tracking. SiamMask [42], however, does come with some caveats, it assumes objects are constantly in frame and are constantly visible for the duration over which they need to

be tracked. Because SiamMask [42] is class agnostic it is not effected by the problems of the previously discussed methods where if an object is obscured slightly and is failed to be detected. SiamMask [42] remains to be a tracking by detection technique however, its detection process looks only for the object identified upon initialisation. Figure 4.5 displays the 6 initial video frames from a video sequence with track objects generated with this method. The first frame in figure 4.5 displays the detection results output by Mask R-CNN [15]. The subsequent frames display the tracking output of SiamMask [42]. The tracking performance of SiamMask [42] is poor. After detection, the tracked object locations output by SiamMask [42] wander from the initial object they represented for all objects in the scene. This is likely because of SiamMask's [42] class agnosticism, because the scene depicted in figure 4.5 is a laboratory environment and highly atypical compared to datasets SiamMask [42] was trained on (COCO [22], ImageNet-VID [30] and Youtube-VOS [44]).

4.2.1.4 Object Re-identification with Temporal Backlog

All of the methods discussed so far fail to solve one task desirable in all multiple object trackers - object re-identification. For all previous methods, if an object is occluded, goes undetected or leaves and re-enters the frame, it's identity is either reset or the object fails to be considered at all. By storing a temporal backlog of all detections with their assigned colors, for a specified number of video frames prior to the current frame, object identities can be recovered after occlusion, misdetection or exiting and entering frame. This method utilises the same procedure to match object in adjacent frames as the initial instance segmentation method presented in section 3.2.1.2. Object re-identification results are shown in figure 3.5 where the identified cup in the left frame is unidentified in the right frame and once again identified with its original ID. This method is able to recover objects when only one object of a particular class is obscured at a time. For example, if two objects of the same class were to simultaneously become undetected, then after several frames of random undetected movement, both objects were to reappear, their identities would be predicted to be the same as the identity of the object closest to the location at which the objects were last detected. After many frames of untracked movement, the chances of two objects recovering their initial identities approaches 0.5. This essentially means after long term occlusion or periods of misdetection of more than one object of the same class simultaneously, the likelihood of an ID switch is equal to that of the correct identities being assigned, as this method does not allow duplicate ID's.

4.2.2 Leveraging Information from Segmentation to Generate Object Representations

The main method of Multiple Object Tracking this report proposes attempts to utilise information from the mask in order to build object representations which can be used to identify objects. This method provides a structure with the capability of recovering multiple occluded or lost objects at any location after they have been lost. Figure 4.6 shows the tracking results on 10 sequential video frames from the cup/person dataset used to develop the implementations in this paper. Tracking results here are reasonable with only one ID switch that persists for the remainder of the sequence.

One of the main pitfalls of the object representation method described in section 3.2.2 is that in the initial frames of the sequence there are relatively few object representations stored for

previous frames to compare with object representations in the next frame. This means in the first frames of a sequence, if objects change shape, are occluded, or leave the frame, the set representations generated for a particular object may not be robust enough to allow for reliable representation comparison. The tracking performance of this method compared to the method described in section 3.2.1.2, is relatively low on this particular sequence. This may be because the frame rate is such that using a nearest neighbour like method, although simple, allows for high tracking performance.

Although visually the two people in figure 4.6 are relatively different from the view of the camera, they have still had an ID switch. One factor which may have caused this ID switch specifically after the second frame is that when the mode of the best matched representations is taken across each frame, if two representations have the same frequency across all frames, the mode calculation chooses at random what the mode should be. In the case of figure 4.6 when comparing representations to determine the ID of objects in frame 3, if the objects with best representations for each frame are different the mode will be randomly selected between the two representations. This is another drawback that comes from generating online object representations for objects. One way this could be overcome would be to not only store the best matching representation for each frame when determining an object's ID, but also storing the associated error with this representation. By storing the error, if two representations have an equal frequency across all previous frames, as is possible with the third frame in figure 4.6, the errors of these representations across all frames can then be to inform on the best representation to match the current object.

Another factor which may have contributed toward this is the fact that the representations are scaled in order for their sizes to be matched. Scaling the representations warps and distorts the information stored inside them. This probably alters the likeness of the representation to the original object and obscures it, making the reliability of error calculations heavily dependent on the severity of the scaling required to match the dimensionality of any two representations.

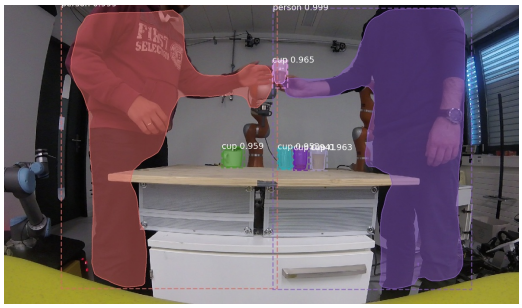
Figure 4.7 displays the tracking results for the first 9 frames in the ETH-Bahnhof video in the 2D-MOT 2015 dataset. Tracking results are poor for this video, where ID switches are apparent across all frame transitions. This is again likely due to the fact that few object representations are available for comparison across the initial frames of a video sequence. If this causes severe issues with tracking across the initial frames of a video, the same problem is propagated through into later frames because the tracking results from the initial frames are so heavily distorted.

One possible adaptation of this method which seeks to amalgamate the method described in section 3.2.1.2 could overcome the problem of building a robust representation dictionary during the early frames of a video sequence. If the number of objects of each class, and the total number of objects in the scene is the same between 2 adjacent frames, a reasonable assumption would be that no objects have entered or exited frame, or have been occluded. If this is the case, the simplistic method of nearest neighbour matching could be used to determine object identities. During the initial frames of a video sequence, this could enable for more robust representation dictionary storage by eliminating the need to determine the mode of only a few representations.

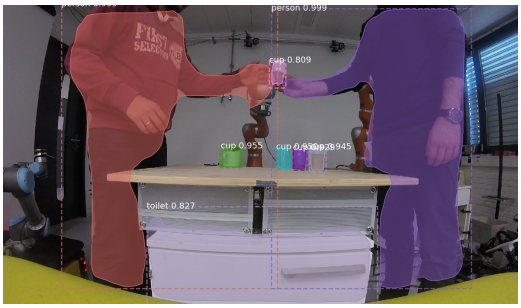
An additional problem with this method is that fact that it has a bias against new objects, similar to the previous method. Because the mode is used to determine the object identities, if an object leaves the frame and is replaced by a different object of the same class, it will be identified as the object that originally left the frame, even if there are high errors when

comparing representations. This is due to the mode being used to determine the object identity, even if errors are large, because the mode does not take into account the errors, and this method will not allow duplication of identities, new objects will be identified as previously seen objects.

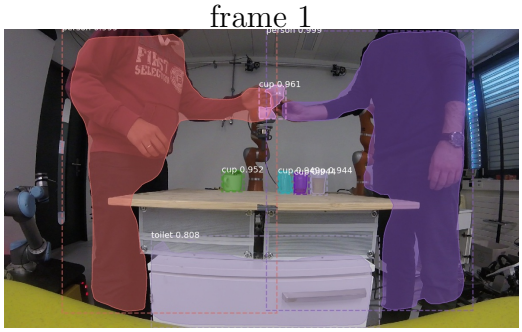
Finally, the computational expense of this method, both in terms of speed and storage are poor. Any high performance Multiple Object Tracking algorithm must be able to operate in real time. This algorithm initially provides little computational expense over Mask R-CNN [15] as there are very few object representations to be compared at the beginning of a video sequence. However, once the algorithm has reached significant time instances, the number of stored object representations becomes vast, requiring a large storage facility as well as superior processing power to execute in real time.



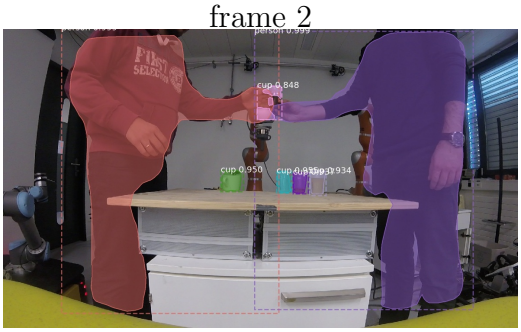
frame 1



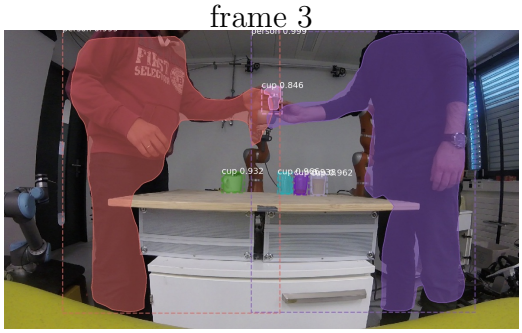
frame 2



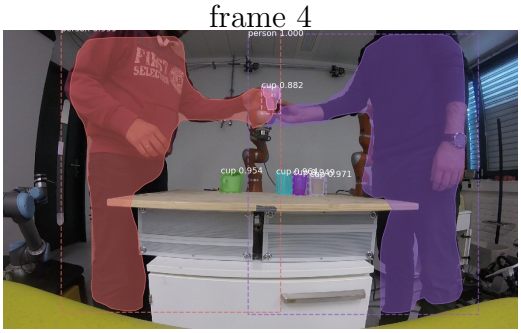
frame 3



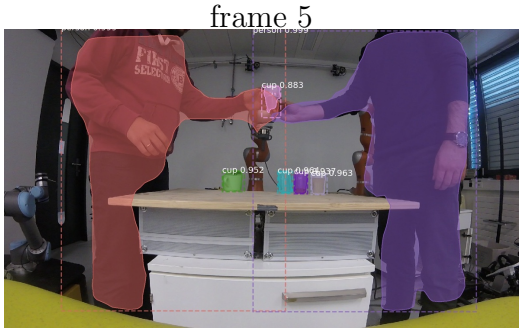
frame 4



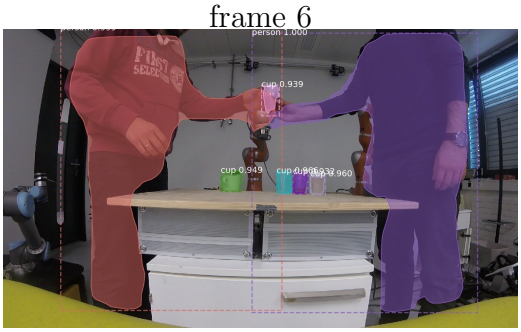
frame 5



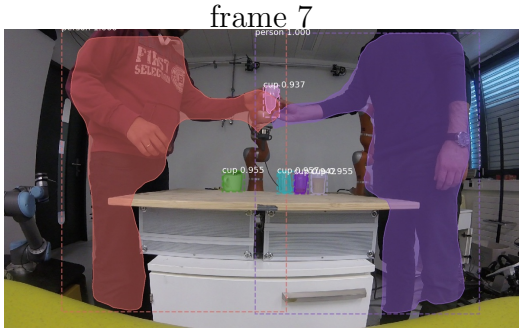
frame 6



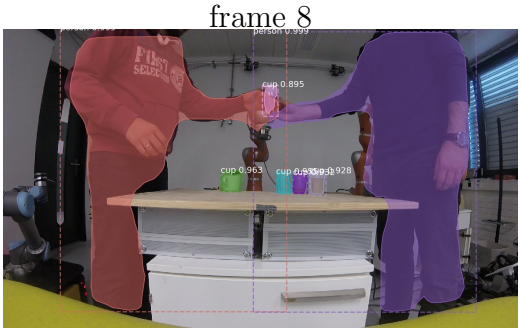
frame 7



frame 8

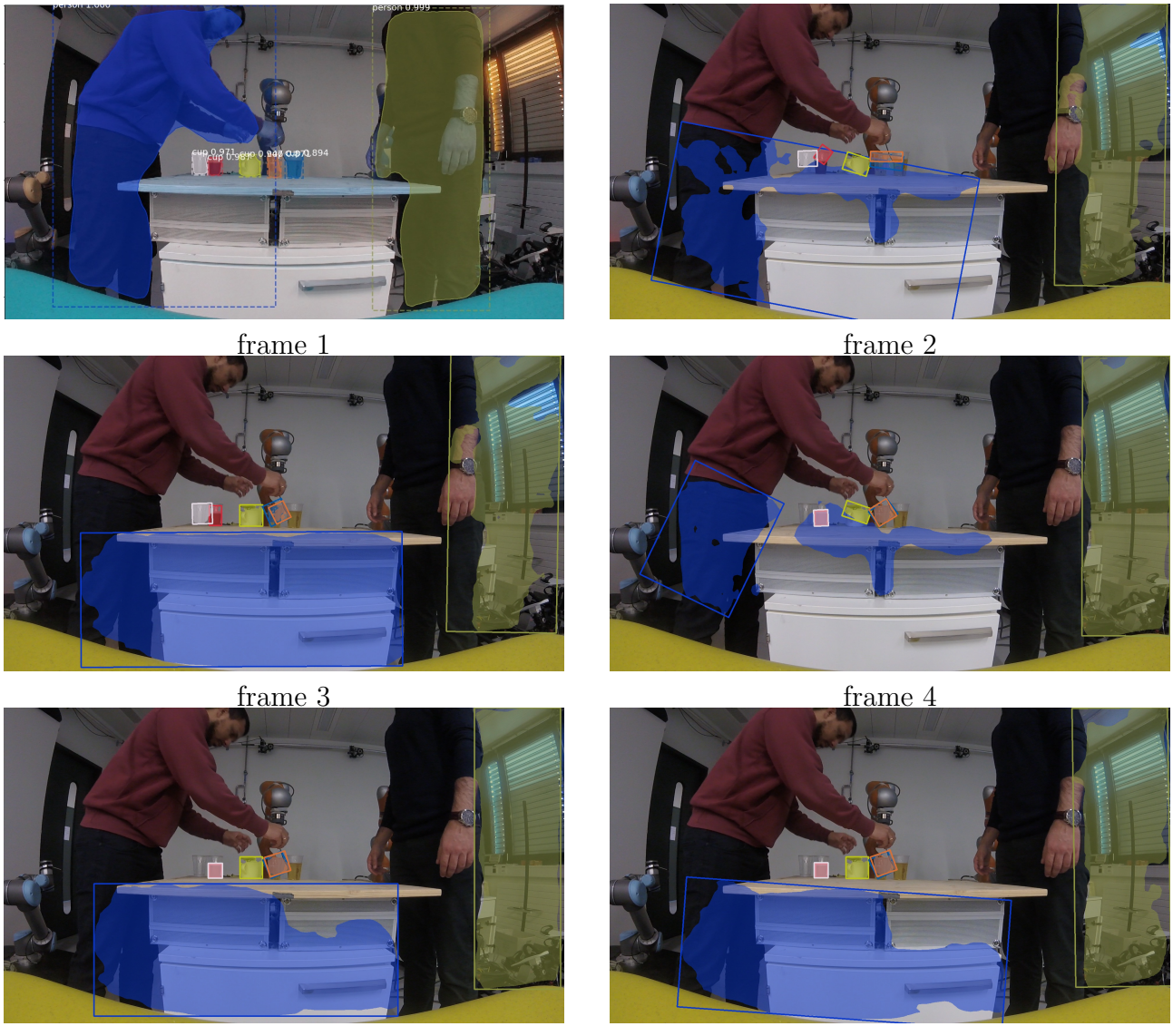


frame 9



frame 10

Figure 4.4: *10 sequential video frames with tracked objects using method from section 3.2.1.2: Loosely Enforced Temporal Instance Segmentation using Mask R-CNN [15].*



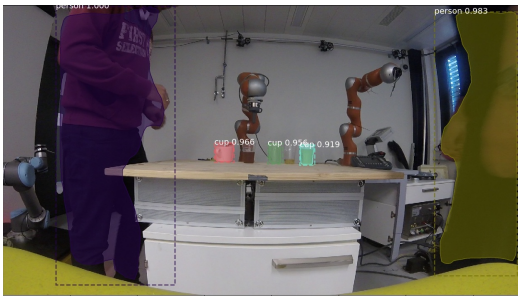
frame 1 frame 2

frame 3 frame 4

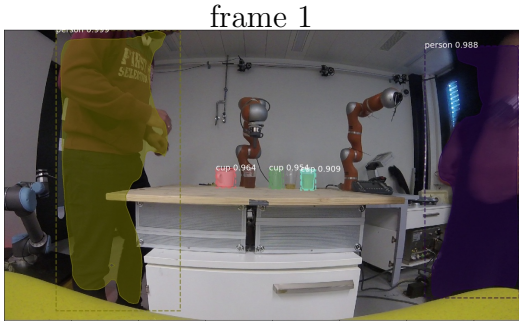
frame 5 frame 6

Figure 4.5: *6 initial video frames with tracked objects using method from section 3.2.1.3: Pseudo-Multiple Object Tracking with SiamMask [42].*

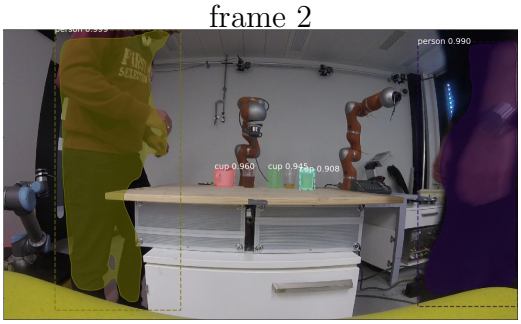
frame 1



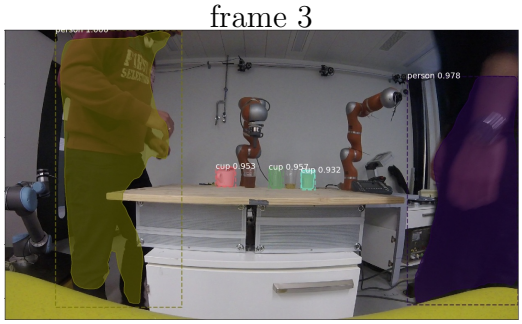
frame 2



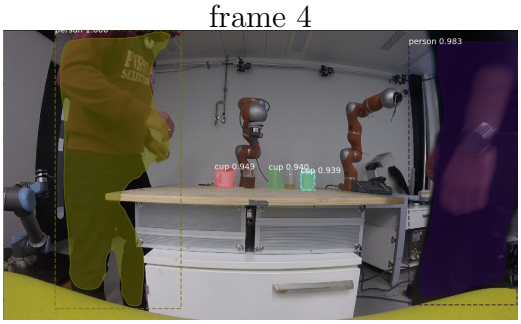
frame 3



frame 4



frame 5

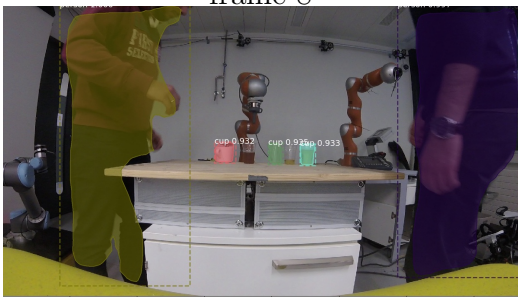


frame 6

frame 7

frame 8

frame 9



frame 10

Figure 4.6: *10 sequential video frames with tracked objects using the main method of this report: Leveraging Information from Segmentation to Generate Object Representations.*

Figure 4.7: *9 sequential video frames with tracked objects using the main method of this report: Leveraging Information from Segmentation to Generate Object Representations. Tested on the video: ETH-Bahnhof from the 2D-MOT 2015 dataset [20].*

Chapter 5

Conclusion

5.1 Summary

Overall, the methods described and evaluated in this paper did meet the aim of the project. A novel approach was used undertaken in Multiple Object Tracking, where information from the segmentation results from the object tracker were leveraged in order to create stored object representations which could be used to compare to future object representations.

Surprisingly, in some cases, a simple class object tracker can be implemented to successfully track multiple objects. If only one object of per class is present in a frame, and no new objects of those classes are introduced for the duration of a video sequence, class object trackers can successfully track multiple objects, even through occlusions. This method fails however as soon as more than one object of a single class is introduced. In practice, when a Multiple Object Tracking algorithm needs to be deployed, there are several instances of at least one class of object. Essentially, this method of Multiple Object Tracking only works in a very small niche of domains, unlikely to be present in the real world.

A small improvement over class tracking was achieved by using a nearest neighbour matching algorithm. By matching the closest bounding box for objects of the same class to identify new objects, multiple instances of objects of the same class could be tracked with reasonable performance. This method does however have significant drawbacks; if an object is occluded, leaves frame or is not detected for a single frame, the tracker will assign a new identity to the object after it is detected once again.

In order to become more robust to detection faults, SiamMask [42] was implemented so that the tracker did not have to rely on the performance of an underlying Object Detector. Multiple Object Tracking results were poor executing several instances of SiamMask[42] in parallel. This was likely because the training data used for SiamMask [42] encompasses a number of realistic domains. Because the dataset used for this project was a fairly niche laboratory environment, a domain SiamMask [42] was unfamiliar with, the tracker failed to identify the relevant objects in the scene after as little as one frame. SiamMask [42] was also not able to deal with new objects being introduced during the video or occlusions.

To overcome problems with occlusions, a temporal backlog of objects was stored for the nearest neighbour method. This allowed for object identities to be recovered after occlusion or other obscurity. This method however did have a bias against new objects. For example, if an object

left the scene, and a new object of the same class entered the scene during the others absence, the identity of the initial object would be assigned to the new object.

The main method of this report, which used data association to match objects using an online generate catalog of object representations from previous frames, had poor performance overall. This was mainly due to the fact that in the first few frames of a video sequence, there are very few object representations to use to infer the identity of objects in subsequent frames. This meant ID switches are abundant in the first few frames of tracking, which then get compounded and obscure tracking results for frames later in a video sequence. As proposed, to solve this, the nearest neighbour method could be incorporated into frames where it is probable no occlusions or introduction of new objects has occurred.

To conclude, the aim of developing a novel approach to Multiple Object Tracking was achieved, however, the tracking performance of this implementation was poor. Overall, this was likely due to the problem of generating unique and robust representations of object from very little data. This method does however include a segmentation output for a Multiple Object Tracking algorithm, where most state of the art Multiple Object Trackers produce only a bounding box output for each tracked object.

5.2 Future work

Given more time and computational resources, exploring the effects of different detectors or fine tuning detector network weights could improve tracking performance significantly, especially on niche datasets like for autonomous driving. Another improvement which could be implemented to improve tracking performance is to use some sort of feature descriptor like SIFT to represent objects in a way which makes them more easily distinguishable. Overall, given a significantly longer amount of time, a Recurrent Neural Network could be implemented to sequentially input video frames and compute Multiple Object Tracking with the use of an end to end model, which has seemed to significantly improve the performance of state of the art methods in Object Detection and Single Object Tracking.

Bibliography

[1] Waleed Abdulla. *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. https://github.com/matterport/Mask_RCNN. 2017.

[2] Luca Bertinetto, Jack Valmadre, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. "Fully-Convolutional Siamese Networks for Object Tracking". In: *CoRR* abs/1606.09549 (2016). arXiv: 1606.09549. URL: http://arxiv.org/abs/1606.09549.

[3] Alex Bewley, ZongYuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. "Simple Online and Realtime Tracking". In: *CoRR* abs/1602.00763 (2016). arXiv: 1602.00763. URL: http://arxiv.org/abs/1602.00763.

[4] S.S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House radar library. Artech House, 1999. ISBN: 9781580530064. URL: https://books.google.co.uk/books?id=lTIfAQAAIAAJ.

[5] David S. Bolme, J. Ross Beveridge, Bruce A. Draper, and Yui Man Lui. "Visual object tracking using adaptive correlation filters". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2010), pp. 2544–2550.

[6] Zhaowei Cai and Nuno Vasconcelos. "Cascade R-CNN: Delving into High Quality Object Detection". In: *CoRR* abs/1712.00726 (2017). arXiv: 1712.00726. URL: http://arxiv.org/abs/1712.00726.

[7] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. "Hybrid Task Cascade for Instance Segmentation". In: *CoRR* abs/1901.07518 (2019). arXiv: 1901.07518. URL: http://arxiv.org/abs/1901.07518.

[8] Chee-Kong Chui. *Website/blog of Chee-Kong Chui and his research group in NUS*. 2019.

[9] CrowdAI. *Mask-RCNN baseline for the crowdAI Mapping Challenge*. https://github.com/crowdAI/crowdai-mapping-challenge-mask-rcnn. 2018.

[10] Daniel Hedges Dmitry Kudinov. *Reconstructing 3D buildings from aerial LiDAR*. 2018.

[11] *Faster R-CNN*. 2019.

[12] T.E. Fortmann, Yaakov bar-shalom, and Molly Scheffe. "Sonar tracking of multiple targets using joint probabilistic data association. IEEE Journal of Oceanic Engineering, OE-8, 173-184". In: *Oceanic Engineering, IEEE Journal of* 8 (Aug. 1983), pp. 173–184. DOI: 10.1109/JOE.1983.1145560.

[13] Ross Girshick. "Fast R-CNN". In: *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*. ICCV '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 1440–1448. ISBN: 978-1-4673-8391-2. DOI: 10.1109/ICCV.2015.169. URL: http://dx.doi.org/10.1109/ICCV.2015.169.

[14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition* (June 2014). DOI: 10.1109/cvpr.2014.81. URL: http://dx.doi.org/10.1109/CVPR.2014.81.

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. "Mask R-CNN". In: *CoRR* abs/1703.06870 (2017). arXiv: 1703.06870. URL: http://arxiv.org/abs/1703.06870.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *Lecture Notes in Computer Science* (2014), pp. 346–361. ISSN: 1611-3349. DOI: 10.1007/978-3-319-10578-9_23. URL: http://dx.doi.org/10.1007/978-3-319-10578-9_23.

[17] R. E. Kalman. "A New Approach to Linear Filtering And Prediction Problems". In: *ASME Journal of Basic Engineering* (1960).

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[19] H. W. Kuhn and Bryn Yaw. "The Hungarian method for the assignment problem". In: *Naval Res. Logist. Quart* (1955), pp. 83–97.

[20] Laura Leal-Taixé, Anton Milan, Ian D. Reid, Stefan Roth, and Konrad Schindler. "MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking". In: *CoRR* abs/1504.01942 (2015). arXiv: 1504.01942. URL: http://arxiv.org/abs/1504.01942.

[21] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. "High Performance Visual Tracking with Siamese Region Proposal Network". In: June 2018, pp. 8971–8980. DOI: 10.1109/CVPR.2018.00935.

[22] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. "Microsoft COCO: Common Objects in Context". In: *CoRR* abs/1405.0312 (2014). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312.

[23] University of Oxford. *SiamMask project webpage*. 2019.

[24] Michele Pace. *Multi-target Tracking with PHD Filters*. 2019.

[25] Ondrej Pesek. *GRASS GIS Addon to generate vector masks from geospatial imagery*. https://github.com/ctu-geoforall-lab-projects/dp-pesek-2018. 2018.

[26] Donald B. Reid. "An Algorithm for Tracking Multiple Targets". In: *IEEE Transactions on Automatic Control* 24 (1979), pp. 843–854.

[27] Jason Remillard. *Use Mask R-CNN/Keras/TensorFlow and OSM to find features in satellite images for fun*. https://github.com/jremillard/images-to-osm. 2017.

[28] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

[29] Facebook Artificial Intelligence Research. *Facebook AI Research webpage*. 2019.

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. "ImageNet Large Scale Visual Recognition Challenge". In: *CoRR* abs/1409.0575 (2014). arXiv: 1409.0575. URL: http://arxiv.org/abs/1409.0575.

[31] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[32] Axel Sauer, Elie Aljalbout, and Sami Haddadin. "Tracking Holistic Object Representations". In: *ArXiv* abs/1907.12920 (2019).

[33] Okinawa Institute of Science Technology. *Usiigaci: stain-free cell tracking in phase contrast microscopy enabled by supervised machine learning*. https://github.com/oist/usiigaci. 2018.

[34] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Robert Fergus, and Yann Lecun. "Overfeat: Integrated recognition, localization and detection using convolutional networks". English (US). In: *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*. 2014.

[35] ShiJie Sun, Naveed Akhtar, HuanSheng Song, Ajmal Mian, and Mubarak Shah. "Deep Affinity Network for Multiple Object Tracking". In: *IEEE transactions on pattern analysis and machine intelligence* (2018).

[36] Jack Valmadre, Luca Bertinetto, João F. Henriques, Andrea Vedaldi, and Philip H. S. Torr. "End-to-end representation learning for Correlation Filter based tracking". In: *CoRR* abs/1704.06036 (2017). arXiv: 1704.06036. URL: http://arxiv.org/abs/1704.06036.

[37] Ba-Ngu Vo and Wing-Kin Ma. "The Gaussian Mixture Probability Hypothesis Density Filter". In: *Signal Processing, IEEE Transactions on* 54 (Dec. 2006), pp. 4091–4104. DOI: 10.1109/TSP.2006.881190.

[38] Ba-Ngu Vo, Samiksha Singh, and Arnaud Doucet. "Sequential Monte Carlo implementation of the PHD filter for multi-target tracking". In: vol. 2. Feb. 2003, pp. 792–799. ISBN: 0-9721844-4-9. DOI: 10.1109/ICIF.2003.177320.

[39] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. "MOTS: Multi-Object Tracking and Segmentation". In: *CoRR* abs/1902.03604 (2019). arXiv: 1902.03604. URL: http://arxiv.org/abs/1902.03604.

[40] Waleedka. *Nuclei Counting and Segmentation in Microscopy Images*. https://github.com/matterport/Mask_RCNN/tree/master/samples/nucleus. 2018.

[41] Qiang Wang. *SiamMask*. https://github.com/foolwood/SiamMask. 2019.

[42] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H. S. Torr. "Fast Online Object Tracking and Segmentation: A Unifying Approach". In: *CoRR* abs/1812.05050 (2018). arXiv: 1812.05050. URL: http://arxiv.org/abs/1812.05050.

[43] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. "Simple Online and Realtime Tracking with a Deep Association Metric". In: *CoRR* abs/1703.07402 (2017). arXiv: 1703.07402. URL: http://arxiv.org/abs/1703.07402.

[44] Ning Xu, Linjie Yang, Yuchen Fan, Jianchao Yang, Dingcheng Yue, Yuchen Liang, Brian L. Price, Scott Cohen, and Thomas S. Huang. "YouTube-VOS: Sequence-to-Sequence Video Object Segmentation". In: *CoRR* abs/1809.00461 (2018). arXiv: 1809.00461. URL: http://arxiv.org/abs/1809.00461.

[45] Su Ye. *Mask R-CNN for Surgery Robot*. https://github.com/SUYEgit/Surgery-Robot-Detection-Segmentation. 2017.

[46] Weixing Zhang, Chandi Witharana, Anna Liljedahl, and Mikhail Kanevskiy. "Deep Convolutional Neural Networks for Automated Characterization of Arctic Ice-Wedge Polygons in Very High Spatial Resolution Aerial Imagery". In: *Remote Sensing* 10 (Sept. 2018), p. 1487. DOI: 10.3390/rs10091487.