Learning to detect occluded objects in videos

Satyaki Chakraborty

May 09, 2019

The Robotics Institute

Carnegie Mellon University

Pittsburgh, Pennsylvania 15213

Thesis Committee:

Martial Hebert, advisor

Simon Lucey, co-chair

Achal Dave, co-chair

Thesis proposal submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics

Abstract

Occlusion is one of the most significant challenges encountered by object detectors or trackers. While object detection in videos has received a lot of attention in the past few years, most existing methods in this domain do not target detecting objects when they are occluded. However, being able to detect or track an object of interest through occlusion has been a long standing challenge for different autonomous tasks. Traditional methods that employ visual object trackers with explicit occlusion modeling experience drift and (or) make several fundamental assumptions about the data. We propose to address this with an end-to-end method that builds upon the success of region based video object detectors which aims to learn to model occlusion in a data-driven way. Finally, we show that our method is able to achieve superior performance with respect to state-of-the-art video object detectors on a dataset of furniture assembly videos collected from the internet, where small objects like screws, nuts, and bolts often get occluded from the camera viewpoint.

Contents

Chapter 1

Introduction

Occlusion is abundant in natural world and often poses a significant challenge in reliably tracking or detecting objects. Current object detection tasks [5,6,8,21,27] mostly ignore full occlusion. However, detecting fully occluded objects can be useful in certain cases and help us better understand video scenes. For instance, in order to build machines that are capable of assembling furniture by looking at demonstration videos, it might be useful to be able to reliably detect or track small tools that often get occluded from the camera viewpoint.

While some older methods model explicit occluder-occludee relationships in visual object trackers to track objects through occlusion, in this paper we plan to develop our ap-

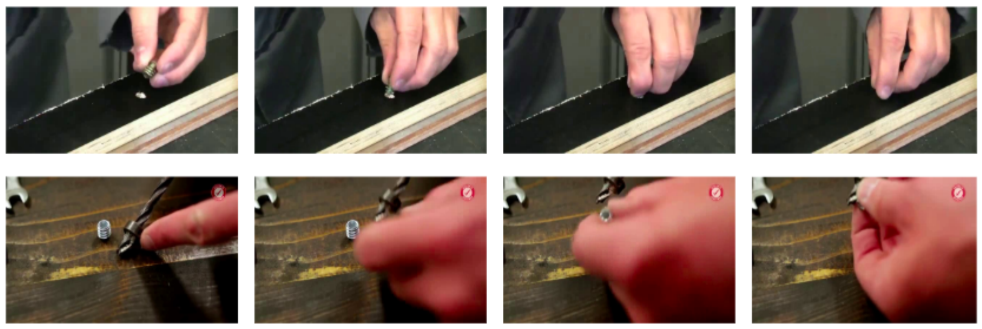


Figure 1.1: Occlusion observed in furniture assembly demonstrations. Small tools like screws often get occluded from camera viewpoint.

proach following the more recent *tracking by detection* paradigm, where instead of using visual object trackers to track occluded objects, we use video object detectors to detect them. The frame level detections obtained from the detector can be linked temporally using association methods like [3,31] to generate tracks. This way we are able to prevent catastrophic failures encountered by visual object trackers where we completely lose track of the object and are unable to track it further even when it has reappeared. Also *tracking by detection* enables us to easily extend our method for multi-object scenarios. In this paper, however, we only concentrate on the detection subproblem, since there has been considerable amount of work done for developing different association methods to solve multi-object tracking.

Frame level object detectors like Faster RCNN [26], RFCN [4] can hardly distinguish between the two situations, one where an object is occluded and the other where the object is not present. Video object detectors on the other hand, look into a temporal context surrounding the query frame to accumulate features, and then reason on top of the accumulated features. This helps the detector to collect features of the object of interest from surrounding frames in the video even if the object is occluded in the current frame.

Existing video object detectors [2,9,12,17–19,32–35] are mostly designed to handle partial occlusion, motion blur, unseen viewpoints amongst other issues that frame level detectors are not capable of dealing with. Although we treat full occlusion the same way we treat any of the above mentioned issues, we believe detecting a fully occluded object is slightly more challenging mainly because of the following reasons. a) When an object is occluded in a frame, information about the class of the object does not come from that particular frame and hence the architecture needs to heavily rely on the temporal connections to obtain this information. b) In full occlusion, the temporal connections usually act on features belonging to completely different classes of objects. For example, if a coffee mug is occluded by the hand of the person using the mug, the temporal connections need to combine features of the mug and the human hand and determine that the mug is occluded by the human hand. In traditional tasks for video object detection, temporal connections usually act on features belonging to the same object category. c) Since information of the object doesn't come from the frames where it is occluded, and the duration of occlusion can vary, it is necessary to propagate information from the beginning or end of the video sequence.

In this research, we aim to solve the problem in a data driven end-to-end fashion by adding a recurrent computational unit inside region based object detectors following [32] to enable propagation of features of the occluded object from both ends of the video. In doing so, we are able to maintain an approximate position of the object through occlusion. We use spatio-temporal memory networks from [32] as our baseline model and show that our model achieves a substantial improvement in terms of raw detection score (mAP) under such severe cases of occlusion. Finally, we further show that our method is also able to achieve competitive results with the state of the art methods on video object detection datasets like ImageNet VID [1] where full occlusion is hardly present.

Chapter 2

Prior work

2.1 Tracking through occlusion

Prior work on tracking objects through occlusion with visual object trackers mostly model explicit inter-occlusion relationships between objects in the scene [14], formulating motion models [23] or use external knowledge [15] with visual object trackers. Although such methods are useful when we do not have prior information about the objects that we want to track, visual object trackers suffer from a fundamental issue: once they lose track of the object of interest, they can hardly recover from that. Also, methods that model occlusion explicitly end up making several fundamental assumptions about the data (which includes motion of objects, objects belonging to foreground or background, etc.) which do not make them ideal for real world scenarios. For eg., [14] assumes both *occluder* and *occludee* objects belong to the foreground and thus learn to model occlusion relationships explicitly by classifying different events of occlusion. Specifically, when an object is getting occluded, they learn to identify it as a region merging event, whereas when an occluded object is becoming visible they learn to identify it as a region splitting event. Thus, by identifying such region merging, splitting and (or) continuation events, their approach is able to track an object through occlusion. Unless the *occluder* objects are known before hand, it is hard to make such methods work in a more general setting.

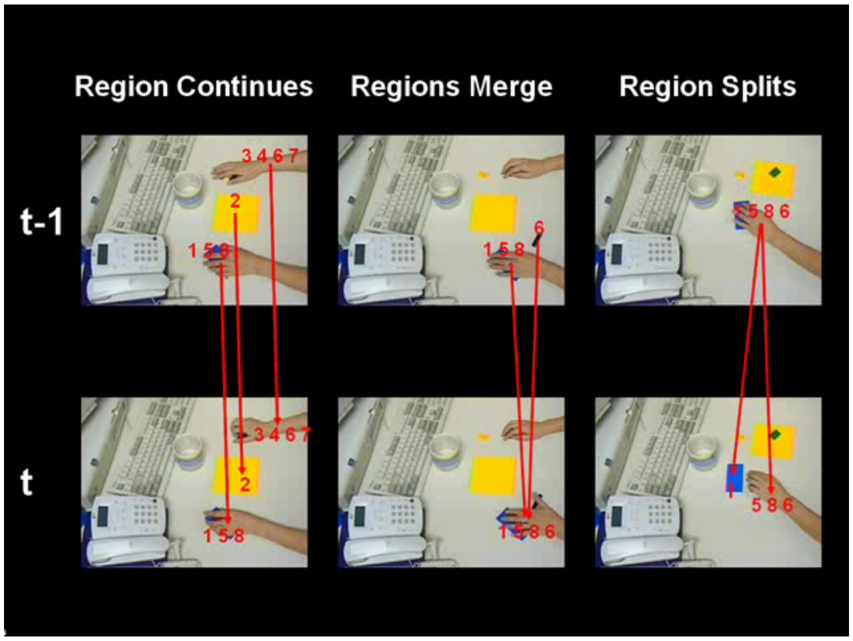


Figure 2.1: Different occlusion events being determined as per Huang and Essa. Image is being reused from [14]

2.2 Tracking by detection paradigm

The *drift* encountered by traditional visual object trackers can in certain cases lead to irrecoverable failures. Even when the object reappears and is visible, if the appearance model of the tracker has changed considerably, it will not able to track the object of interest further. Also, visual object trackers cannot be easily extended to cases where there are multiple objects of interest appearing and disappearing through out a video sequence. These issues can be tackled by the recently popular method of tracking by detection. The core idea is that instead of tracking the objects directly across adjacent frames with visual object trackers, we run object detectors at every frame. Once we have the bounding boxes from the detector, we use some data association method like [3, 31] to link the boxes at different time steps to form tracks. Since there has been considerable amount of work in developing association methods, in this research we only concentrate on the subproblem of robust detection of objects under occlusion. It is be noted that, a frame level object detector, i.e. an object detector that runs on individual frames of a video is unable to understand occlusion. Hence in order to detect occluded objects it is necessary to use video object detectors which aggregate features from a temporal context surrounding a query frame to produce detections.

2.3 Frame level object detection

Over the last few years, object detection [4,10,11,20,22,24–26] in static images has received quite a lot of attention. Their success can mostly be attributed to very deep convolutional backbones [13, 30]. Earliest of such detectors is a two stage object detector R-CNN [11], where at first region proposals were computed from the image and then each and every proposal was classified. Later on, computationally lighter versions of the original R-CNN was made possible by leveraging ROI pooling layers [10] and by sharing the convolutional backbone with the region proposal network [26]. RFCN [4] introduces the *position sensitive* ROI pooling layer, and achieves significant speed up compared to [26] while achieving competitive detection accuracy. For our application, we mostly build up on Faster RCNN [26] and RFCN [4], the two most popular region based object detectors.

2.4 Video level object detection

A more recent task in the domain of object detection [2, 9, 12, 17–19, 32–35], video object detection, has also been given a lot of attention in the past few years. Even though static object detectors can be easily applied to individual frames of a video, there are certain difficult cases where they fail to perform reasonably well. Such difficult cases can mostly be attributed to occlusion, motion blur and unseen poses of the objects and these cases make the detection task challenging for the per-frame detector. This is where video object detectors come in. Instead of solely looking at the query frame, video object detectors solve these problems by accumulating features from a temporal context surrounding the given query frame with the hope that rich features of the object can be obtained from at least some of the frames in that context. Most of the recent approaches have concentrated on how to propagate such features efficiently in time.

Although, some of the older methods like [12, 17–19] have heavily relied on exhaustive post processing of detections produced by frame level detectors, more recent approaches [2, 9, 32–35] in this domain focus on building connections across the convolutional feature maps of the network backbones at different time steps to aggregate features and then reason on such aggregated features. Such methods significantly outperform methods which rely on post-processing of frame level detections. A good number of these methods [2,9,35] use a short window of frames centered around the frame of interest to accumulate features, while methods like [16, 32, 34] use recurrent computational units like LSTM cell, Spatio-temporal memory module(STMM) and ConvGRU cell [29] to propagate features in time.

Chapter 3

Datasets

Most existing datasets for video object detection do not take into consideration objects that undergo full occlusion. Since visual cues of objects are not present when they are fully occluded, existing datasets treat them as if they are not present and hence have no ground-truth annotations for such object instances. This creates the need for datasets with ground truth annotations for occluded objects. While detecting fully occluded objects in static images can lead to ambiguity and is often redundant, such is not always the case for videos (for example, learning from demonstration tasks). That being said, even in videos, one can handcraft specific situations where it might be impossible to annotate the location of the occluded object precisely. Hence, in this research, we restrict our task to indoor scenarios, where we try to detect common handheld objects that undergo occlusion (mostly by the human hand). When an object gets fully occluded, we annotate the object based on an approximate guess, which may not be precisely accurate. The datasets we collect are explained as follows.

3.1 Staged occlusion dataset

The first type of dataset we collect comprises of videos where common hand-held desktop objects like mugs, calculator, notepads, etc. are being manipulated by an individual in front of a simple white background and scenes are captured by a static camera. Objects are manipulated in such a way that they remain occluded by the hand for sufficiently long periods of time from the camera viewpoint while being in motion. Figure 3.1(a) shows an example scene from this dataset.

By reducing the noise from external factors like camera movement, scale variation, background clutter, etc. we are able better analyze and qualitatively evaluate the impact of occlusion on different methods in this *staged occlusion* dataset. We primarily use this dataset for debugging and understanding the effect of long term occlusion on different methods.

3.2 Furniture assembly dataset

The second type of dataset comprises of different furniture assembly tasks collected from the internet as shown in figure 3.1(b). This dataset is more representative of occlusions that happen in the natural world. In this case, we try to detect only one class of objects (all small tools like screws, nuts and bolts are grouped into one class). Objects have a lot of scale variation and are often occluded by hands and different tools like hammer and screwdriver. We use this *assembly* dataset for quantitative evaluation by reporting mean average precision (mAP) of different detection methods.

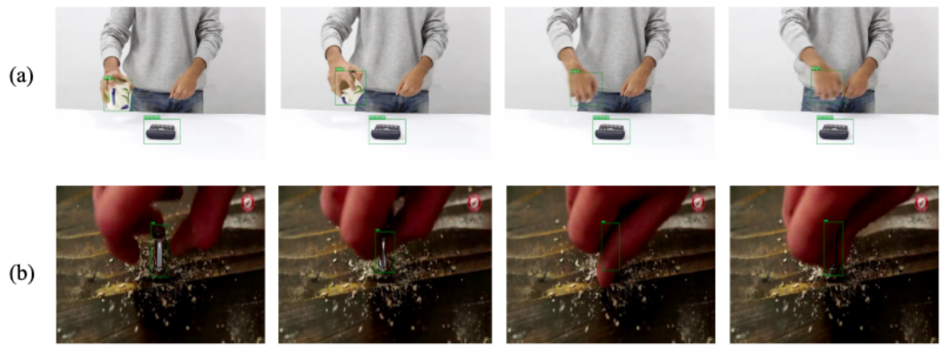


Figure 3.1: (a) *staged occlusion dataset* with long term occlusion and simpler background settings (b) *furniture assembly dataset* with resemblance to natural occlusion scenes. In both cases, the ground-truth annotations are shown in green.

Finally, we use the ImageNet VID dataset [1] to evaluate our method and see how it performs against existing methods for video object detection in datasets which do not target full occlusion. This is because in our data driven approach we do not explicitly model occlusion and hope to learn to do it through the temporal connections of the video object detector. Hence ideally our model should be able to adapt to both cases (where occlusion is ignored and where occlusion is annotated) fairly easily.

Chapter 4

Approach

4.1 Exploiting temporal context for detection

As mentioned earlier, in order to understand occlusion, it is important to build a video level object detector. This can be further explained with the following example in figure 4.1.

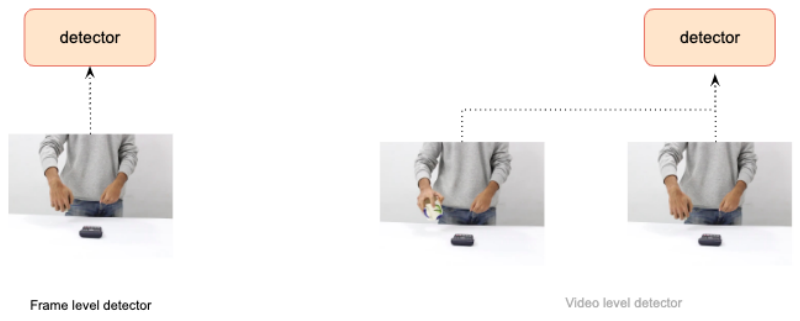


Figure 4.1: Frame level vs video level object detection. Object is only present on the right hand of the individual while the left hand is empty. The per frame object detector (left) is unable to distinguish between the two cases: one where the object is absent and the other where the object is present but fully occluded. The video level object detector, on the other hand, is able to distinguish between the two cases by exploiting the temporal context

Video object detection methods that use recurrent computational units are not bound by time and thus in theory have the capability of propagating information from the very ends of the video sequence. When an object is occluded, information about the occluded object do not come from the corresponding frames where it remains occluded. Window based approaches like [2, 9, 35] can be sub-optimal in this case because the length of the frame window can become a bottleneck. Instead, we build recurrent connections on top of region based object detectors that enable propagation of salient features from both ends of a video sequence. The architecture of a region based object detectors at frame level and at video level (with recurrent connections) is explained as follows.

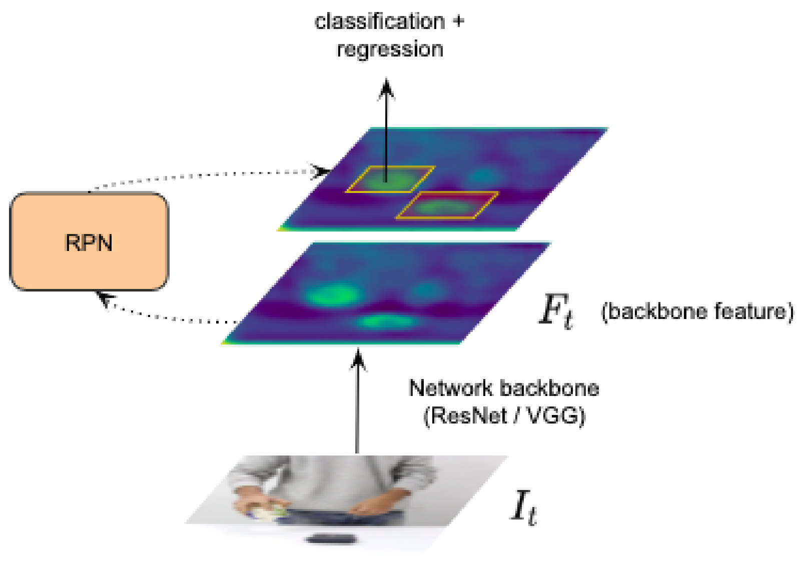
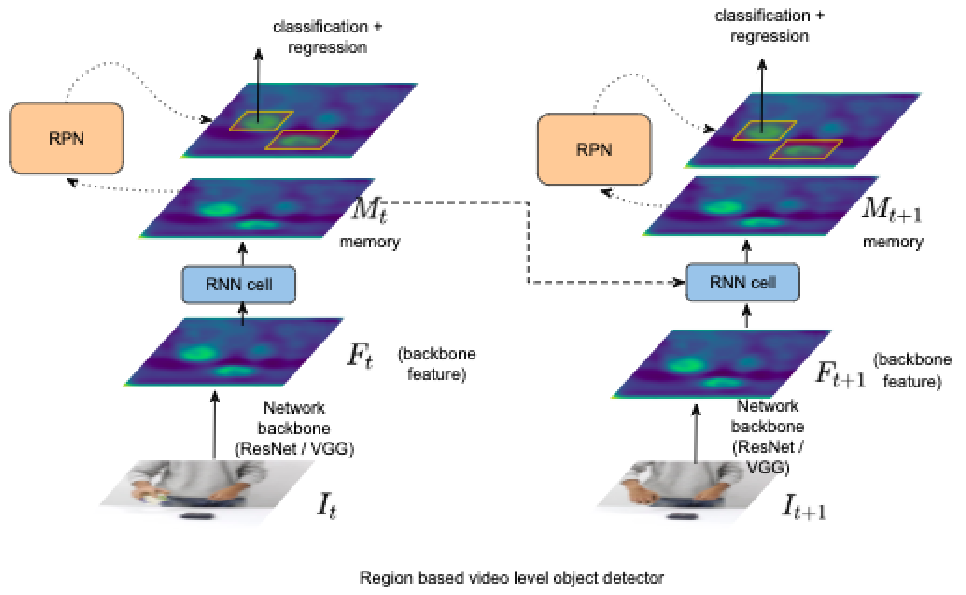


Figure 4.2: Frame level region based object detector

In frame level region based object detectors like Faster RCNN or RFCN, the input frame is passed through a convolutional backbone (typically ResNet or VGG backbones) to obtain a full image backbone feature F_t (subscript denotes time step of frame). A region proposal network (RPN) runs on F_t and produces several ROI crops. Each such ROI crop is then processed further for classification and class specific offset regression. The final offset regression helps in producing slightly tighter or relaxed boxes for better localisation. In order to extend this family of architectures, we first detach the RPN and the ROI specific layers (ROI pooling layers, classification and regression layers) and then add the recurrent connections on top of the backbone feature maps. We call the output of our RNN cell as memory

(M_t) since it has features from all past frames of the sequence. We then attach the RPN and the ROI specific layers on top of this memory. Since we are reasoning on top of the accumulated features and not on the backbone features of a particular time step, we hope to reason on top of features of the object of interest which are propagated to the current memory from when it was last visible in the past. This way we are able to detect the object of interest under occlusion. Our video level object detector is next shown as follows.



classification + regression
RPN
M_t
memory
RNN cell
F_t
(backbone feature)
Network backbone (ResNet / VGG)
I_t
classification + regression
RPN
M_{t+1}
memory
RNN cell
F_{t+1}
(backbone feature)
Network backbone (ResNet / VGG)
I_{t+1}
Region based video level object detector

Figure 4.3: Video level region based object detector

4.2 Redundancy of video data: 2-stage training

Video data is highly redundant. This is because consecutive frames belonging to the same video is highly correlated and thus the number i.i.d. samples in a video dataset is significantly less than the same in a corresponding static image dataset with similar size. For eg., consider the ImageNet VID and DET dataset (with the same classes). While VID has roughly 1 million frames in the training set and the corresponding subset in DET has frames in the order of 10^5, almost each and every training sample in DET is independently and identically distributed. On the other hand, in the VID training set all the frames are taken from a total of 3862 videos and thus the number of i.i.d. samples are of the order of 10^4. In order deal with this, it is recommended that we first pretrain the frame level detector in a static image dataset and then add the recurrent connections and finetune the entire architecture on the video dataset in an end-to-end fashion. In our case we use *Cut, paste and learn* [7] to generate synthetic static image datasets on which we pretrain our frame level detector. Some example images from our synthetic dataset is shown as follows.

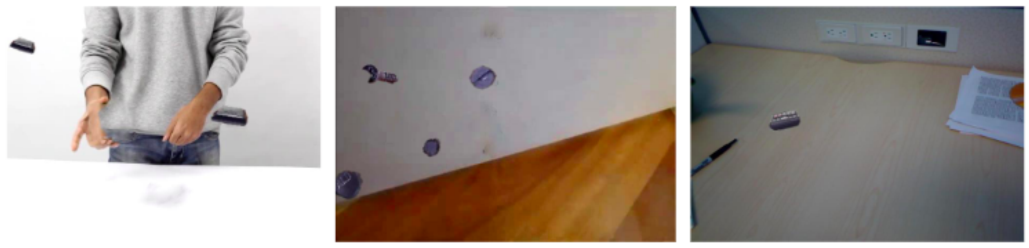


Figure 4.4: Examples of images in our synthetic dataset

As observed in figure 4.4, the objects of interest are randomly placed on respective background images. This works well enough for our purpose because region based object detectors rely heavily on local context for object detection.

4.3 Choice of recurrent computational unit

ConvGRU, ConvLSTM cells are common choices for the recurrent computational units of convolutional neural networks. These cells are inspired from the original GRU and LSTM cells with gating mechanisms, where the dot product layers are replaced by convolutional layers. While both ConvGRU and ConvLSTM cells have been widely used in the domain of computer vision, in our case we take a closer look at the ConvGRU cell since it has fewer parameters. It is to be noted that, region based video object detectors when unrolled for training can occupy a lot of GPU memory and thus fewer parameters of the RNN cell is always preferred.

Computation steps inside a ConvGRU is explained as follows.

$$z_t = \sigma(W_z * F_t + U_z * M_{t-1}) \tag{4.1}$$

$$r_t = \sigma(W_r * F_t + U_r * M_{t-1}) \tag{4.2}$$

$$\hat{M}_t = tanh(W * F_t + U * (M_{t-1} \odot r_t)) \tag{4.3}$$

$$M_t = (1 - z_t) \odot M_{t-1} + z_t \odot \hat{M}_t \tag{4.4}$$

Here, F_t and M_{t-1} denote the backbone feature map at time t and the memory from time $t-1$ respectively. This two features are used to obtain the memory at time t, M_t in the RNN cell. z_t and r_t are gates that control how much information from the previous memory, M_{t-1} to forget and how to combine M_{t-1} and the candidate memory $\hat{M}_t$.

However there is a slight disadvantage with the ConvGRU cell. In figure 4.2, the region proposal network and the RoI specific layers directly work on F_t which has values in the range $[0, \infty)$ because the ReLU activations in the convolutional backbone. When we attach the ConvGRU cell, and fine-tune the architecture in an end-to-end fashion, the region proposal network and the RoI specific layers work on M_t which now has values in the range [0, 1]. This makes the pretrained weights of the RPN and classification/regression layers redundant. In order to deal with that, we use a more recently developed ConvRNN cell called the Spatio-temporal memory module (abbreviated as STMM) which is given by the following set of equations.

$$z_t = BN^*(ReLU(W_z * F_t + U_z * M_{t-1})) \tag{4.5}$$

$$r_t = BN^*(ReLU(W_r * F_t + U_r * M_{t-1})) \tag{4.6}$$

$$\hat{M_t} = ReLU(W * F_t + U * (M_{t-1} \odot r_t)) \tag{4.7}$$

$$M_t = (1 - z_t) \odot M_{t-1} + z_t \odot \hat{M_t} \tag{4.8}$$

Here BN^* is not the typical batch norm layer in convolution neural networks. Rather this layer ensures takes the unbounded inputs and squashes them into the range [0, 1]. It does so by first computing mean μ and standard deviation σ. It then sets $k = \mu + 3\sigma$, clamps the input tensor within the range $[0, k]$ and then divides it by k. This makes sure the values for the update and reset gates are within [0, 1] while the values in the output memory are in the range $[0, \infty)$. It is to be noted that both ConvGRU and STMM have the same basic structure but use different activations for non-linearity.

In our approach, we build on top of STMM because of

- its impressive performance on the ImageNet VID dataset
- additional advantage of easy transfer of pretrained backbone weights from static image datasets

We observe pretraining the weights of the baseline per-frame detector to be particularly useful in our case due to the small volume of training videos.

4.4 Memory misalignment issue

A vanilla STMM is unable to align the memory properly. Successive such misalignments end up forming a trail of salient features in the memory, which often leads to false positive detections and inaccurate localisation. Xiao and Lee [32] address this issue by introducing the *MatchTrans* module. They use correlation between the backbone features to determine affinity coefficients, Γ, which are then used to warp the spatio-temporal memory for alignment. More specifically, the coefficients for alignment at location (x, y) are determined by computing the affinity between $F_t(x, y)$ and the feature cells in a small $2k + 1$ by $2k + 1$ window around (x, y) in F_{t-1}. Affinity coefficient between $F_t(x, y)$ and $F_{t-1}(x + i, y + j)$ is given by equation 4.9. Finally, the aligned memory, M'_{t-1} is obtained from the unaligned memory M_{t-1} following equation 4.10.

$$\Gamma_{x,y}(i, j) = \frac{F_t(x, y).F_{t-1}(x + i, y + j)}{\sum_{i,j\in\{-k,..,k\}} F_t(x, y).F_{t-1}(x + i, y + j)} \tag{4.9}$$

$$M'_{t-1}(x, y) = \sum_{i,j\in\{-k,..,k\}} \Gamma_{x,y}(i, j).M_{t-1}(x + i, y + j) \tag{4.10}$$

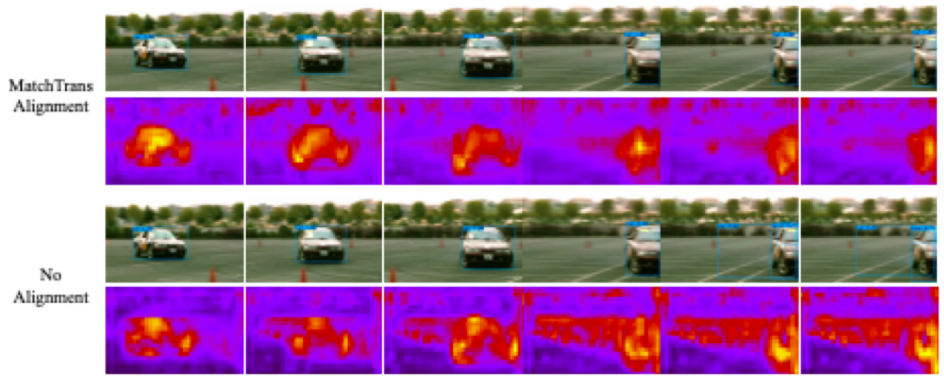


Figure 4.5: Misalignment issue. 2nd row and 4th row from top show L2 norm of memory with and without the MatchTrans module. Image is reused from [32]

4.5 Issues with explicit alignment

While this method of explicit alignment works well for objects that are not completley occluded, we observe that under severe long-term occlusion, correlation based alignment can do away with salient features in the memory of the RNN cell. This happens because, when an object is occluded, backbone feature activations are not always fired for the object (since occluders can often belong to the background class). This in turn results in lower affinity coefficients for the spatial locations where the occluded object exists at a given time. Applying *MatchTrans* over successive time steps, results in dying out of the feature activations in memory and can thus result in false negatives as shown in figure 4.6.

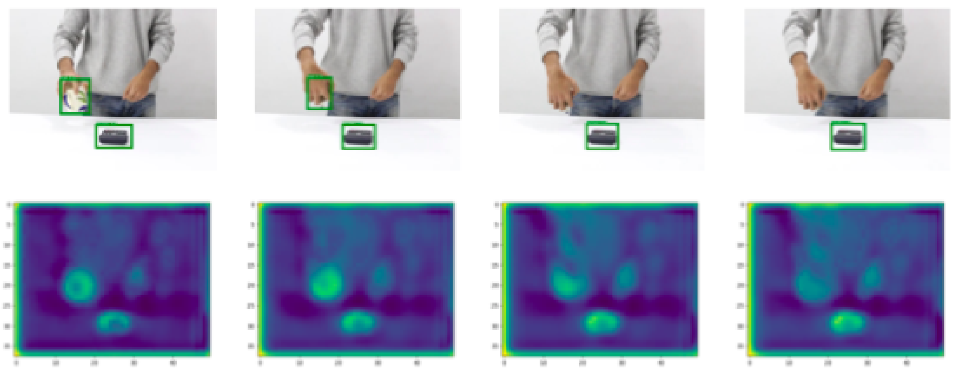


Figure 4.6: Memory activations of the object held die with time resulting in false negatives. Adjacent time steps shown are 5 frames apart.

This introduces a trade-off between long-term propagation of features under occlusion and better alignment for accurate localisation. We propose to address this via an alignment learning module, that can act as an alternative to explicit correlation based alignment.

4.6 Learning the alignment

Standard implementation of convRNN cells including the STMM cell uses features from the same spatial locations of its inputs, M_{t-1} and F_t to update a cell in its output feature M_t. However, unless the objects in a scene are static or moving very slowly, such operations can be problematic, especially since it is a common convention to skip frames from a video to deal with the redundancy of adjacent frames. We believe in order to align memory with standard convolution layers, we should at least ensure large enough receptive fields for the layers of the RNN cell with respect to its input features. A naive implementation of this can be achieved by increasing the kernel size or adding successive 3 by 3 convolution layers. Although simple, such architectures are not memory efficient since adding each convolution layer only increases the receptive field by a finite amount. Hence, the number of parameters, ΔP needed to be added scale linearly with respect to the increase in the effective receptive field, Δf i.e. $\Delta P = O(\Delta f)$. In stead we propose the following method.

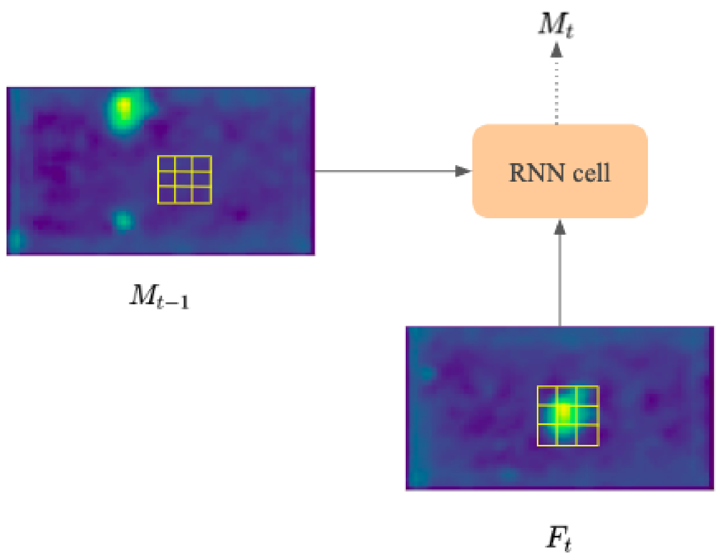


Figure 4.7: A standard convolution rnn cell using features from the same spatial location of the previous memory and input feature map to form the new memory

First, we build feature pyramids of the input features of the RNN cell (M_{t-1} and F_t).

For this step, we use standard 2 x 2 max pooling operation for downsampling, and set the number of levels of the pyramid to 3, which gives a good enough balance between memory usage and performance. Feature pyramids of M_{t-1} and F_t are given by M^f_{t-1} and F^f_t where $M^f_{t-1} = \{M_{t-1}, M^{0.5}_{t-1}, M^{0.25}_{t-1}\}$ and $F^f_t = \{F_t, F^{0.5}_t, F^{0.25}_t\}$. Here, numerical superscript denotes scale. We next propagate information using only the top most level of the pyramid i.e. using $M^{0.25}_{t-1}$ and $F^{0.25}_t$ instead of their corresponding full resolution feature maps. This way, we are able to increase the effective receptive field of the layers in the RNN cell without adding more parameters to it. The output of the STMM cell, M_t thus needs to be upsampled to be passed on to other subnetworks of the object detector like the region proposal network, ROI pooling layers etc. In order to upsample the newly updated memory , we use skip connections from the backbone feature pyramid F^f_t to combat the information loss due to downsampling and to aid the network in better alignment of the memory. Every level of upsampling has three fundamental steps. Firstly, we do bilinear upsampling to scale the feature maps by 2x followed by an optional zero padding along the width, height or both axes to match the spatial resolution of the corresponding feature map from F^f_t. It is to be noted that this zero padding causes additional misalignment by 1 pixel in the feature space along its corresponding axis. To deal with that, we apply 3 by 3 convolution on top of the feature maps accompanied by the skip connections from the backbone feature pyramid. The entire architecture of our modified recurrent computational unit is shown in figure 4.8. This way we also end up adding much fewer parameters to the network. The only parameters that we add are for the skip connections and the number of such parameters linearly increases with the number of levels in the pyramid, L. On the other hand, the effective receptive field exponentially increases with L. Thus, in our model, $\Delta P = O(\log \Delta f)$

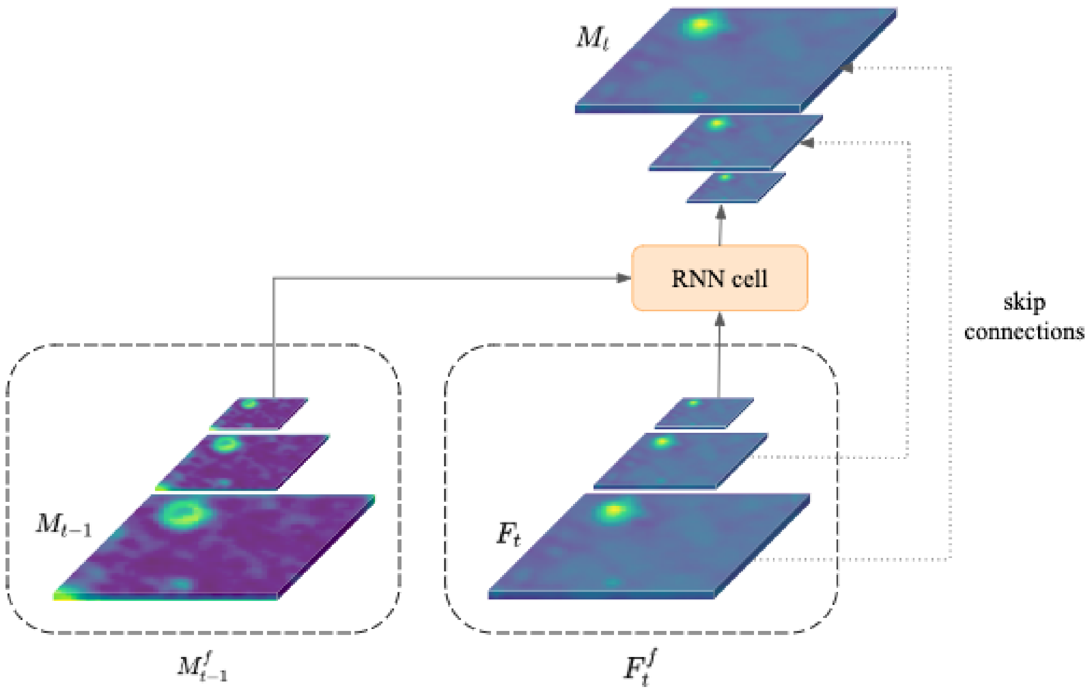


Figure 4.8: Architecture of alignment learning module

Chapter 5

Experiments

In this section, we evaluate our model quantitatively and qualitatively on the respective datasets and show the effectiveness of our model in learning the alignment. We use frame level detectors and STMN as baseline models to compare against our method. Unlike [32], we do not take the ensemble of the frame level detector and STMN. Through out all our experiments we only evaluate the single model performance. It is to be noted that all the modules discussed earlier can be plugged into any existing region based object detector with any backbone. For each of the three datasets we take different combinations of the convolutional backbone and frame level base network and hence show that our method is invariant of the type of backbone and frame level detector.

5.1 Experiments on Staged Occlusion dataset

Framework hyp	**Settings**
Base detector	Faster RCNN
Backbone	vgg16
bptt steps	5
Type of RNN	unidirectional
RoI sampling	random
Type of nms	standard

Table 5.1: main configurations of video level detector for *Staged Occlusion* dataset

5.1.1 Effect on long term occlusion

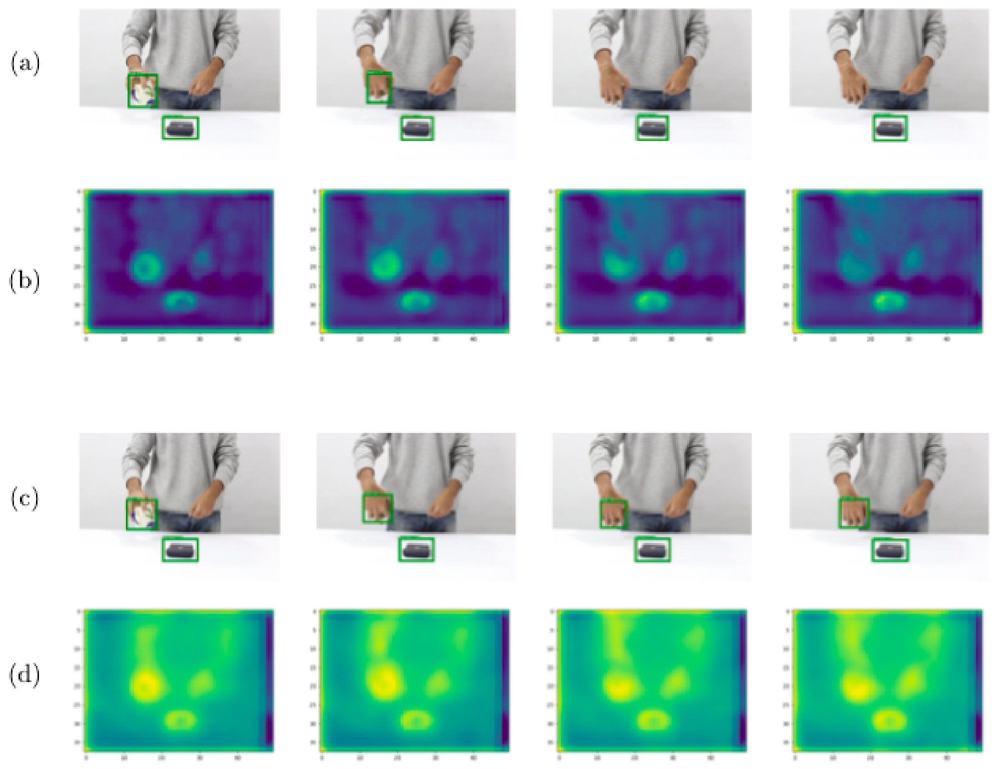


Figure 5.1: L2 norm of memory and detections for different methods. Row(a) and (b) correspond to MatchTrans. Row(c) and (d) correspoding to our method. As is apparent, our method of learning to align the memory keeps the features of the object of interest alive in the memory for longer amount of time, resulting in fewer false negatives.

5.1.2 Qualitative results

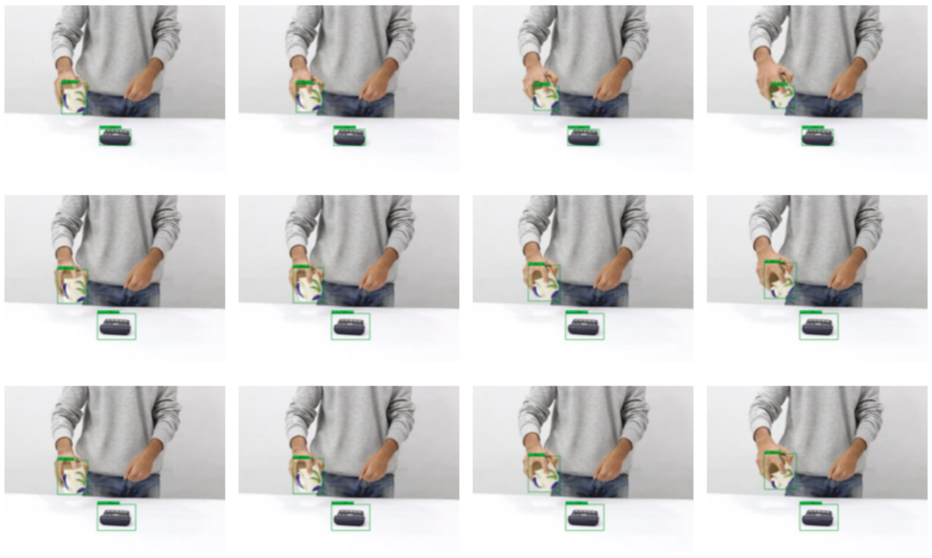


Figure 5.2: Quality of detections when object is visible. Top row: frame level detections, Middle row: video level detecions with MatchTrans, Bottom row: video level detections with learned alignment. Time increases from left to right.

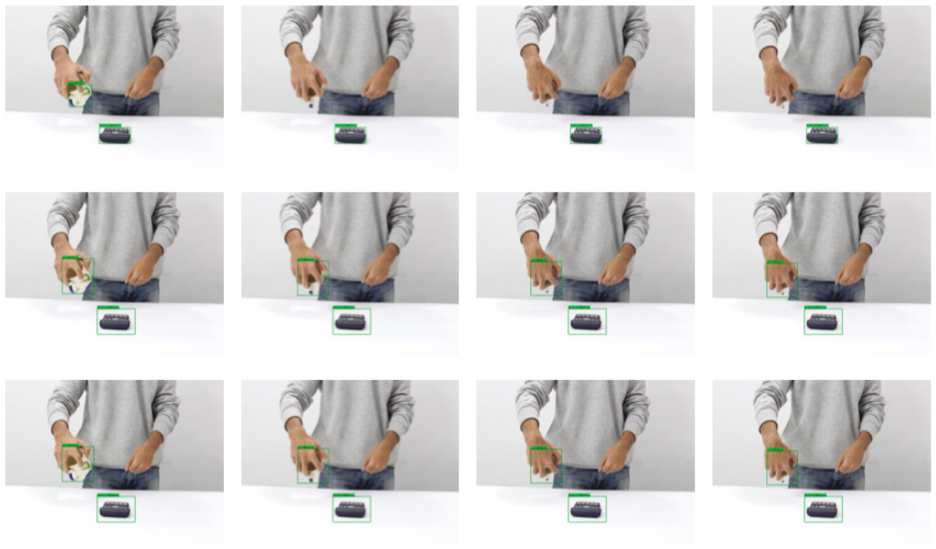


Figure 5.3: Quality of detections as the objects starts undergoing occlusion. Top row: frame level detections, Middle row: video level detecions with MatchTrans, Bottom row: video level detections with learned alignment. Time increases from left to right.

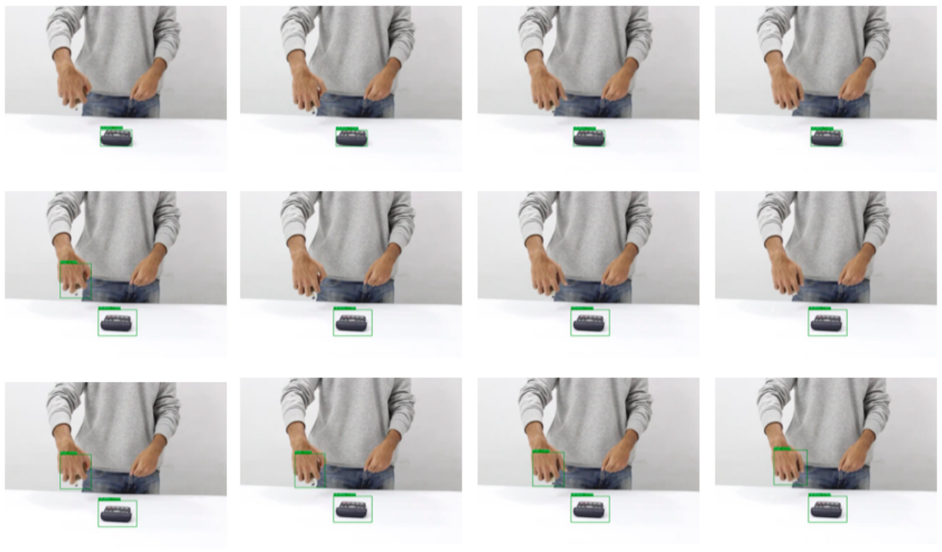


Figure 5.4: Quality of detections as the duration of occlusion increases. Top row: frame level detections, Middle row: video level detecions with MatchTrans, Bottom row: video level detections with learned alignment. Time increases from left to right.

5.2 Experiments on furniture assembly dataset

Framework hyp	Settings
Base detector	Faster RCNN
Backbone	ResNet-50
bptt steps	5
Type of RNN	unidirectional
RoI sampling	random
Type of nms	standard

Table 5.2: main configurations of video level detector for *Furnuture Assembly* dataset

For both the occlusion datasets, we use Faster RCNN with vgg16 and ResNet-50 backbone as the frame level baseline detector. We train this detector on synthetic datasets generated by [7]. Once the Faster RCNN baseline is trained, we add the recurrent connections into the model and fine-tune the entire network in an end-to-end fashion. Since we were interested in building online methods, for our case the RNN is uni-directional where information only flows from the beginning to the end. We use stochastic gradient descent with learning rate 1e-3 in the beginning and lower it to 1e-4 as the training loss plateaus. During training, we employ standard left-right flipping for data augmentation and during test time we use standard non-max suppression with an IoU threshold of 0.3. While there are additional techniques to boost the mAP like OHEM [28] for better ROI sampling, or seq-NMS [12] for better post processing of raw detections, in this case we do not use them. Under these settings, we obtain the following detection scores shown in table 5.3.

Method	Base network	RNN cell	Alignment	mAP
Frame level	Faster RCNN	-	-	0.12
Video level	Faster RCNN	STMM	-	0.15
Video level	Faster RCNN	STMM	MatchTrans	0.21
Video level	Faster RCNN	STMM	learned	0.26

Table 5.3: mAP of different models on the *Furniture Assembly* dataset

Unsurprisingly, we observe that our method significantly outperforms the baseline frame level object detector. Also, the detection scores from table 5.3 confirm that our method of aligning the features are more suitable under such strong cases of occlusion.

5.2.1 Qualitative results

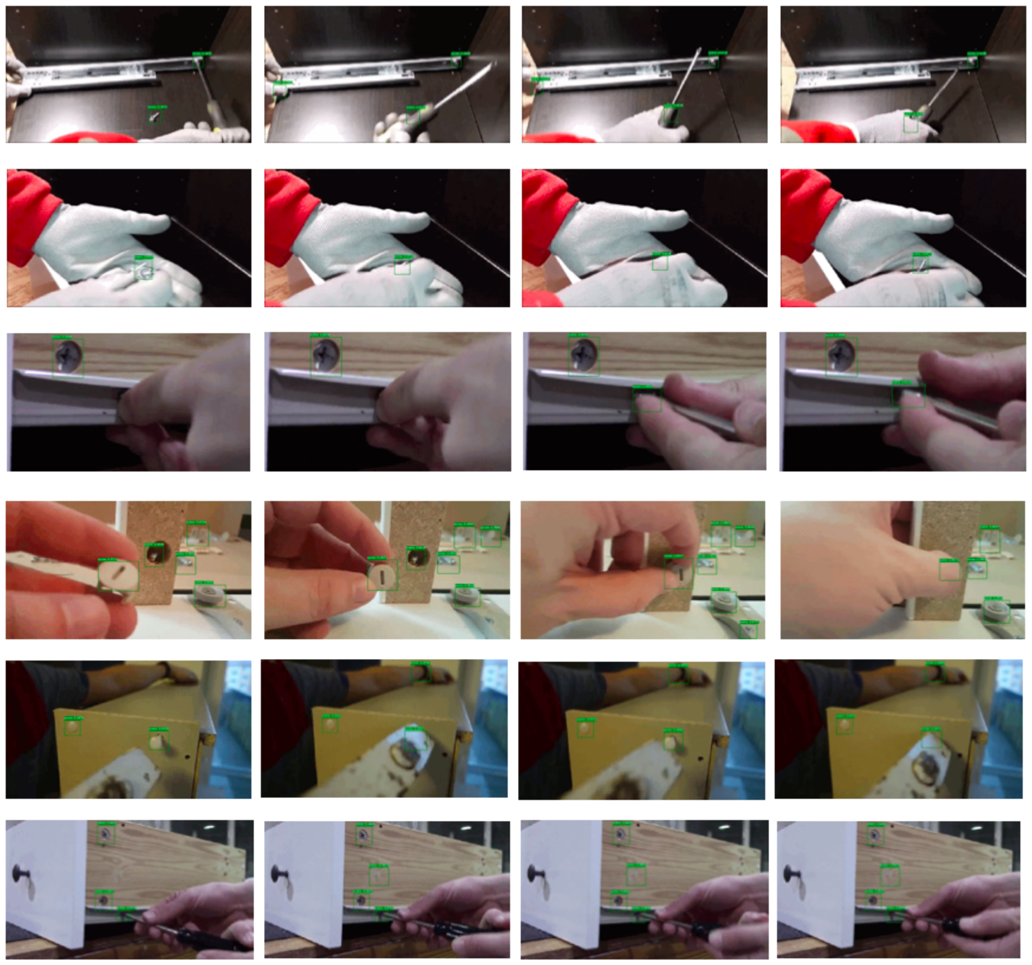


Figure 5.5: Qualitative results on our *Furniture Assembly dataset*. Each row corresponds to a different sequence. Time increases from left to right.

5.3 Experiments on ImageNet VID dataset

In our approach, we try to build a data driven end-to-end method for detecting occluded objects in videos and do not plan to model occlusion explicitly. Hence, it be worthwhile to see how our way learning the alignment compares against explicit alignment with MatchTrans when the occluded object is visible. To do so, we consider the ImageNet VID dataset, a common dataset for benchmarking video object detectors. From figures 5.6, 5.7, **??** we observe that learned alignment gives a relatively better aligned memory when compared to that of MatchTrans.

Framework hyp	**Settings**
Base detector	RFCN
Backbone	ResNet-101
bptt steps	4
Type of RNN	bidirectional
RoI sampling	OHEM
Type of nms	seq-NMS

Table 5.4: main configurations of video level detector for *ImageNet VID* dataset

Further more, we quantitatively evaluate our method's performance on the ImageNet VID dataset to show how our method of video object detection stacks up against current state-of-the-art approaches. In order to make a fair comparison, we make some changes to our method to match the experimental settings of [32]. The details are available in table 5.4. Our settings differ with that of [32] only in two aspects: i) we evaluate single model performance and not performance of the ensemble model with RFCN and ii) during training we unroll the rnn for 4 time steps in stead of 7, because we were unable to fit the latter in a 12 GB Nvidia Titan X GPU. Under these settings, we observe that STMN with *MatchTrans* achieves an mAP of 78.9 and our STMN with learned alignment achieves an mAP of 79.6. Although, we acknowledge that the 0.7 mAP improvement is not necessarily statistically significant, we are able to show that our able is able to learn the alignment well enough to act as an alternative to explicit alignment for videos object detection tasks that do not specifically target fully occluded object detection.

5.3.1 Quality of memory alignment

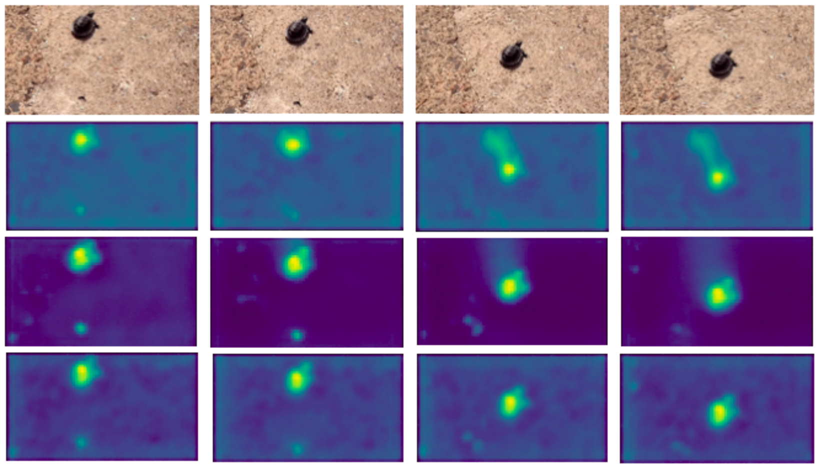


Figure 5.6: Top row: input sequence, 2nd row: L2 norm of memory without alignment, 3rd row: L2 norm of memory with MatchTrans, 4th row: L2 norm of memory with learned alignment. Time increases from left to right.

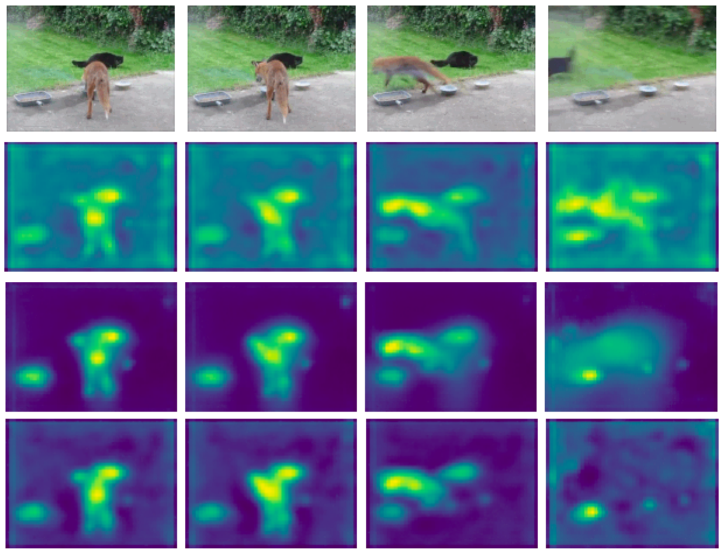


Figure 5.7: Top row: input sequence, 2nd row: L2 norm of memory without alignment, 3rd row: L2 norm of memory with MatchTrans, 4th row: L2 norm of memory with learned alignment. Time increases from left to right.

Chapter 6

Conclusion and future work

In this paper, we present a data driven approach to detecting occluded objects in videos. To the best of our knowledge, prior work on this domain has avoided data driven occlusion reasoning primarily due to lack of available data to train on. Although the advantage of such data driven methods is that we do not need to make any fundamental assumptions about the data, we observe that our method learns some biases for commonly occluding objects that it has seen during training time. This means, not only does it produce false positives, but it also unable to generalize to unseen occluder objects at test time. Future work will be concentrated on generalisation across different occluding objects.

Also, whether a purely data-driven method is not very useful unless there is a significant volume of training data available. Building datasets with such varying levels of occlusion can be laborious. Future work will also target creating synthetic videos and using domain adaptation techniques to address this problem.

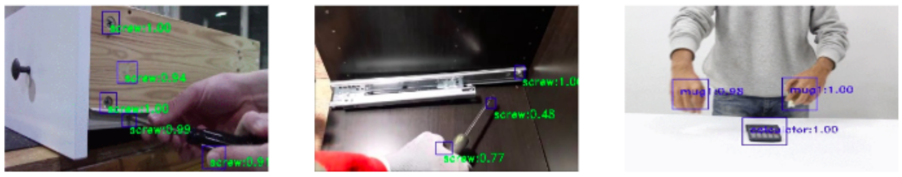


Figure 6.1: Failure cases of our method

Bibliography

[1] http://image-net.org/challenges/lsvrc/2017/results#vid.

[2] G. Bertasius, L. Torresani, and J. Shi. Object detection in video with spatiotemporal sampling networks. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[3] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[4] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[5] D. Damen, H. Doughty, G. Maria Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018.

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

[7] D. Dwibedi, I. Misra, and M. Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1301–1310, 2017.

[8] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[9] C. Feichtenhofer, A. Pinz, and A. Zisserman. Detect to track and track to detect. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3038–3046, 2017.

[10] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[12] W. Han, P. Khorrami, T. L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang. Seq-nms for video object detection. *arXiv preprint arXiv:1602.08465*, 2016.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Y. Huang and I. Essa. Tracking multiple objects through occlusions. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 1051–1058. IEEE, 2005.

[15] N. Joshi, S. Avidan, W. Matusik, and D. J. Kriegman. Synthetic aperture tracking: tracking through occlusions. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[16] K. Kang, H. Li, T. Xiao, W. Ouyang, J. Yan, X. Liu, and X. Wang. Object detection in videos with tubelet proposal networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 727–735, 2017.

[17] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, et al. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):2896–2907, 2018.

[18] K. Kang, W. Ouyang, H. Li, and X. Wang. Object detection from video tubelets with convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 817–825, 2016.

[19] B. Lee, E. Erdenee, S. Jin, M. Y. Nam, Y. G. Jung, and P. K. Rhee. Multi-class multi-object tracking using changing point detection. In *European Conference on Computer Vision*, pages 68–83. Springer, 2016.

[20] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[23] J. Pan and B. Hu. Robust occlusion handling in object tracking. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[25] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[26] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[28] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–769, 2016.

[29] M. Siam, S. Valipour, M. Jagersand, and N. Ray. Convolutional gated recurrent networks for video segmentation. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3090–3094. IEEE, 2017.

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[31] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[32] F. Xiao and Y. Jae Lee. Video object detection with an aligned spatial-temporal memory. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 485–501, 2018.

[33] X. Zhu, J. Dai, L. Yuan, and Y. Wei. Towards high performance video object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7210–7218, 2018.

[34] X. Zhu, J. Dai, X. Zhu, Y. Wei, and L. Yuan. Towards high performance video object detection for mobiles. *arXiv preprint arXiv:1804.05830*, 2018.

[35] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei. Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 408–417, 2017.