symmetry

MDPI

Article

Unsupervised Multi-Object Detection for Video Surveillance Using Memory-Based Recurrent Attention Networks

Zhen He [1,2,*] and Hangen He [1]

[1] College of Intelligence Science, National University of Defense Technology, Changsha 410073, China; hehangen2000@163.com

[2] Department of Computer Science, University College London, London WC1E 6BT, UK

* Correspondence: zhen.he@ucl.ac.uk

Received: 13 August 2018; Accepted: 27 August 2018; Published: 1 September 2018

Abstract: Nowadays, video surveillance has become ubiquitous with the quick development of artificial intelligence. Multi-object detection (MOD) is a key step in video surveillance and has been widely studied for a long time. The majority of existing MOD algorithms follow the "divide and conquer" pipeline and utilize popular machine learning techniques to optimize algorithm parameters. However, this pipeline is usually suboptimal since it decomposes the MOD task into several sub-tasks and does not optimize them jointly. In addition, the frequently used supervised learning methods rely on the labeled data which are scarce and expensive to obtain. Thus, we propose an end-to-end Unsupervised Multi-Object Detection framework for video surveillance, where a neural model learns to detect objects from each video frame by minimizing the image reconstruction error. Moreover, we propose a Memory-Based Recurrent Attention Network to ease detection and training. The proposed model was evaluated on both synthetic and real datasets, exhibiting its potential.

Keywords: object detection; unsupervised learning; recurrent network; memory; attention; video surveillance

1. Introduction

Video surveillance aims to analyze video data recorded by cameras. It has been widely used in crime prevention, industrial processes, traffic monitoring, sporting events, etc. A key step in video surveillance is object detection, i.e., locating multiple objects with bounding boxes in each video frame. This is crucial for downstream tasks such as recognition, tracking, behavior analysis, and event parsing.

Multi-object detection (MOD) from visual data has been extensively studied for many years by computer vision communities. Classical methods such as Deformable Part Models (DPMs) [1] follow the "divide and conquer" pipeline that a sliding window approach is first used to generate image regions, then a classifier (e.g., a Support Vector Machine [2]) is employed to categorize each region into object/non-object, and finally post-processing is applied to refine the bounding boxes of object regions (e.g., removing outliers, merging duplicates, and rectifying boundaries). To improve both the efficiency and performance of MOD, methods based on Region-based Convolutional Neural Networks (R-CNNs) [3–6] are proposed and perform well on various popular object detection datasets [7–11]. In contrast to previous methods, they selectively generate only a small amount of image region proposals and use Convolutional Neural Networks (CNNs) [12,13] as more expressive classifiers. However, as this "divide and conquer" pipeline breaks down the MOD problem as several sub-problems and optimizes them separately, the resulting solutions are usually sub-optimal. To jointly optimize the MOD problem, Huang et al. [14], Redmon et al. [15] and Liu et al. [16] formulated object detection as a

single regression problem that directly maps the image to object bounding boxes, achieving end-to-end learning which greatly simplifies the MOD process.

Nevertheless, all above methods rely on supervised learning that requires labeled data, while manually labeling the object bounding boxes is very expensive. Moreover, unlike general MOD tasks, for video surveillance, we are more interested in a specific class of objects, and the backgrounds usually does not change over time and thus can be easily extracted to ease detection. To this end, we propose a novel framework to achieve unsupervised end-to-end learning of MOD for video surveillance. We summarize our contribution as follows:

- We propose an Unsupervised Multi-Object Detection (UMOD) framework, where a neural model learns to detect objects from each video frame by minimizing the image reconstruction error.
- We propose aMemory-Based Recurrent Attention Network to improve detection efficiency and ease model training.
- We assess the proposed model on both the synthetic dataset (Sprites) and the real dataset (DukeMTMC [17]), exhibiting its advantages and practicality.

2. Unsupervised Multi-Object Detection

The UMOD framework is composed of four modules, including: (i) an *image encoder* extracting input features from the input image; (ii) a *recurrent object detector* recursively detecting objects using the input features; (iii) a parameter-free *renderer* reconstructing the input image using the detector outputs; and (iv) a reconstruction *loss* driving the learning of Modules (i) and (ii), in an unsupervised and end-to-end fashion.

2.1. Image Encoder

Firstly, we use a neural image encoder NN^{enc} to compress the input image $\boldsymbol{X}$ into input feature $\boldsymbol{C}$:

$$\boldsymbol{C} = \mathrm{NN}^{enc}\left(\boldsymbol{X}; \boldsymbol{\theta}^{enc}\right) \tag{1}$$

where $\boldsymbol{X} \in [0,1]^{H \times W \times D}$ has a height H, width W, and channel number D); $\boldsymbol{C} \in \mathbb{R}^{M \times N \times S}$ has a height M, width N, and channel number S; and $\boldsymbol{\theta}^{enc}$ is the network parameters to be learned. By making $\boldsymbol{C}$ contain significantly fewer elements than $\boldsymbol{X}$ and taking it as the input for succeeding modules, we can largely reduce the computation complexity for object detection.

2.2. Recurrent Object Detector

Based on the observation that different objects usually have common patterns in video surveillance MOD tasks, we iteratively apply a same neural model, namely the recurrent object detector, to extract objects from the input feature $\boldsymbol{C}$. This can not only regularize the model, but also reduce the number of parameters, thereby maintaining learning efficiency when object number increases.

The Recurrent Object Detector consists of a recurrent module NN^{rec} and a neural decoder NN^{dec}. In the t-th iteration ($t \in \{1, 2, \ldots, T\}$ where T is the maximum detection step), the detector first updates its state vector (Throughout this paper, we assume the vectors are in row form) $\boldsymbol{h}_t \in \mathbb{R}^R$ via NN^{rec} (parameterized by $\boldsymbol{\theta}^{rec}$):

$$\boldsymbol{h}_t = \mathrm{NN}^{rec}\left(\boldsymbol{h}_{t-1}, \boldsymbol{C}; \boldsymbol{\theta}^{rec}\right) \tag{2}$$

Although NN^{rec} could be naturally represented as a Recurrent Neural Network (RNN) [18–20] ($\boldsymbol{C}$ needs to be vectorized), we model NN^{rec} using a novel architecture to improve the network efficiency, which is discussed in Section 3.

Given $\boldsymbol{h}_t$, the detector output $\mathcal{Y}_t$ can be then generated through a neural decoder NN^{dec} (parameterized by $\boldsymbol{\theta}^{dec}$):

$$\mathcal{Y}_t = \mathrm{NN}^{dec}\left(\boldsymbol{h}_t; \boldsymbol{\theta}^{dec}\right) \tag{3}$$

where the quintuple $\mathcal{Y}_t = \{y_t^c, \boldsymbol{y}_t^l, \boldsymbol{y}_t^p, \boldsymbol{Y}_t^s, \boldsymbol{Y}_t^a\}$ is a *mid-level* representation of the object. Concretely, $y_t^c \in [0,1]$ is the object *confidence* denoting its existence; $\boldsymbol{y}_t^l \in \{0,1\}^I$ is an I-dimensional one-hot vector denoting which image *layer* the object possesses; $\boldsymbol{y}_t^p \in [-1,1]^4$ is the object *pose* containing a normalized scale $[s_t^x, s_t^y]$ and translation $[t_t^x, t_t^y]$, where the real scale is produced by $[\tilde{s}_t^x, \tilde{s}_t^y] = [1+\eta^x s_t^x, 1+\eta^y s_t^y]$ and $\eta^x, \eta^y > 0$ are constants; $\boldsymbol{Y}_t^s \in \{0,1\}^{U\times V\times 1}$ is the mask representing the object *shape*; and $\boldsymbol{Y}_t^a \in [0,1]^{U\times V\times D}$ is the object *appearance*. To obtain output variables of desired range, in the final layer of NN^{dec}, we use the sigmoid function to generate y_t^c and $\boldsymbol{Y}_t^a$, use the tanh function to generate $\boldsymbol{y}_t^p$, and sample from the Categorical and Bernoulli distributions to get $\boldsymbol{y}_t^l$ and $\boldsymbol{Y}_t^s$, respectively. As the sampling process is not differentiable, a Straight-Through Gumbel–Softmax estimator [21] is employed to reparameterize both distributions so that back propagation can still be applied. We have found by experiments that discretizing $\boldsymbol{y}_t^l$ and $\boldsymbol{Y}_t^s$ is crucial to obtain interpretable output variables.

The mid-level representation defined above is both flexible and interpretable. As would be shown below, the output variables can be directly used to reconstruct the input image, through which their interpretability is enforced.

2.3. Renderer

Given the detector outputs $\{\mathcal{Y}_t \mid t = 1, 2, \ldots, T\}$ without training labels, how can we define a training objective? Our solution is to first convert all these outputs into a reconstructed image using a renderer which is differentiable, and then use back propagation to minimize the reconstruction error. To enforce the model to learn to produce desired detector outputs, we make the renderer deterministic and contain no parameters. In this case, correct detector outputs correspond to a correct reconstruction.

Firstly, we use the object pose $\boldsymbol{y}_t^p$ to scale and shift its shape $\boldsymbol{Y}_t^s$ and appearance $\boldsymbol{Y}_t^a$ by using a Spatial Transformer Network (STN) [22]:

$$\boldsymbol{T}_t^s = \mathrm{STN}\left(\boldsymbol{Y}_t^s, \boldsymbol{y}_t^p\right) \tag{4}$$

$$\boldsymbol{T}_t^a = \mathrm{STN}\left(\boldsymbol{Y}_t^a, \boldsymbol{y}_t^p\right) \tag{5}$$

where $\boldsymbol{T}_t^s \in \{0,1\}^{H\times W\times 1}$ is the transformed shape and $\boldsymbol{T}_t^a \in [0,1]^{H\times W\times D}$ is the transformed appearance.

Then, by using the object confidence y_t^c and layer $\boldsymbol{y}_t^l$, we compose I image layers ($I \leq T$), where the i-th layer can be possessed by several objects and is obtain by:

$$\boldsymbol{L}_i^m = \min\left(1, \sum_t y_t^c y_{t,i}^l \boldsymbol{T}_t^s\right) \tag{6}$$

$$\boldsymbol{L}_i^f = \sum_t y_t^c y_{t,i}^l \boldsymbol{T}_t^s \odot \boldsymbol{T}_t^a \tag{7}$$

where $\boldsymbol{L}_i^m \in [0,1]^{H\times W\times 1}$ and $\boldsymbol{L}_i^f \in [0,I]^{H\times W\times D}$ are the layer's foreground mask and foreground, respectively, and $\odot$ is the element-wise multiplication (If the operands have different sizes, we simply broadcast them).

Finally, by using these layers, the input image could be reconstructed in an iterative way, i.e., for $i = 1, 2, \ldots, I$:

$$\hat{\boldsymbol{X}}^{(i)} = (\boldsymbol{1} - \boldsymbol{L}_i^m) \odot \hat{\boldsymbol{X}}^{(i-1)} + \boldsymbol{L}_i^f \tag{8}$$

where we initialize $\hat{\boldsymbol{X}}^{(0)}$ as the background (Note that the background is assumed easy to extract or known in advance), and take $\hat{\boldsymbol{X}}^{(I)}$ as the final reconstructed image. Illustrations of the UMOD framework and the renderer are shown in Figures 1 and 2, respectively.

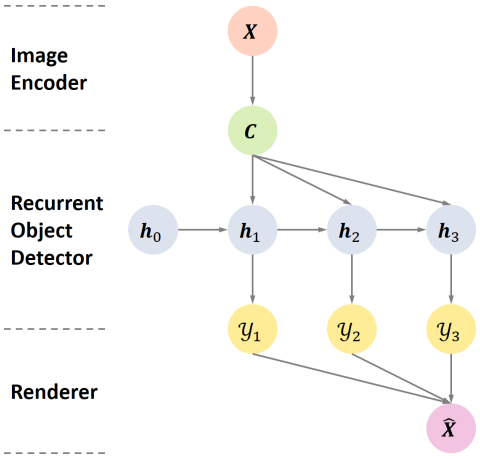

Figure 1. Illustration of the Unsupervised Multi-Object Detection (UMOD) framework, where the detection step $T=3$.

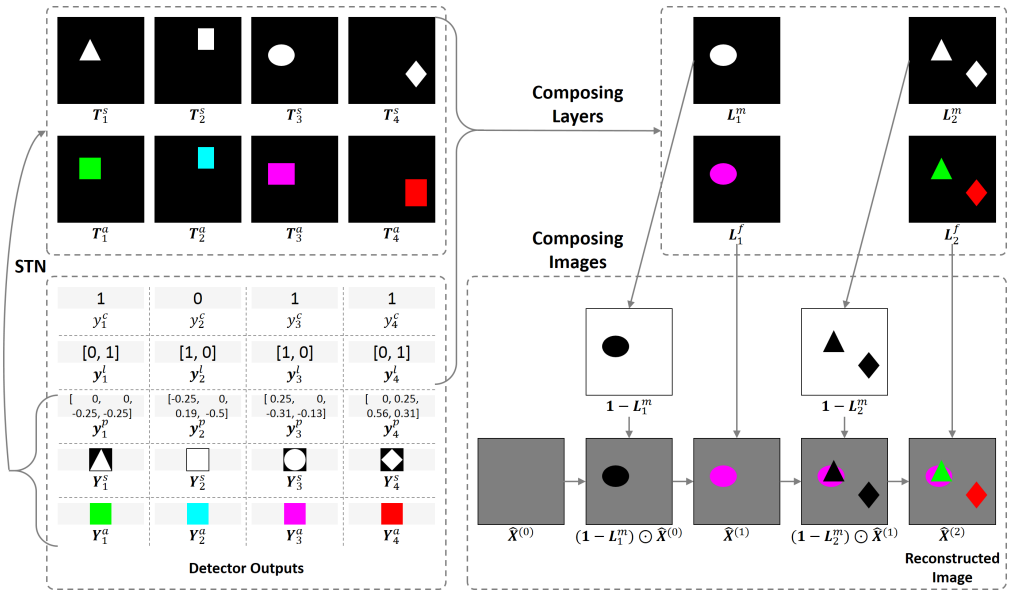

Figure 2. Illustration of the renderer that converts the detector outputs to the reconstructed image, where the detection step $T=4$ and the layer number $I=2$.

Note that our rendering process can be accelerated since matrix operations can be used to parallelize the composition of layers (defined in Equations (6) and (7)). Although handling occlusion requires iterations (defined in Equation (8)) that still cannot be parallelized, we can use fewer layers by setting a smaller I. This is reasonable since occlusion usually happens among a few objects, and it is unnecessary to allocate a layer for each object (non-occluded objects can share a same layer).

2.4. Loss

With the reconstructed image $\hat{\boldsymbol{X}}$, we can then define the loss l for each sample to drive the learning of the image encoder and recurrent object detector:

$$l = \text{MSE}\left(\hat{\boldsymbol{X}}, \boldsymbol{X}\right) + \lambda \cdot \frac{1}{T}\sum_{t} \tilde{s}_t^x \tilde{s}_t^y \tag{9}$$

where $\text{MSE}\left(\cdot,\cdot\right)$ is the Mean Squared Error for reconstruction, and $\lambda > 0$ is the coefficient of the *tightness* constraint $\frac{1}{T}\sum_t \tilde{s}_t^x \tilde{s}_t^y$ used to penalize object scales in order to avoid loose bounding boxes.

3. Memory-Based Recurrent Attention Networks

When using RNN to model the recurrent module NN^{rec} defined in Equation (2), it can suffer two issues: (i) to avoid repeated detection, the detector state $\boldsymbol{h}_t$ must carry information from the previous detections (for $t' < t$), which couples memory and computation, thereby making the detection of the current object less effective; and (ii) to extract features for a specific object, the detector must *learn to* focus on a local area on the input feature $\boldsymbol{C}$, making training more difficult. To this end, we propose a Memory-Based Recurrent Attention Network (MRAN), which overcomes Issue (i) by directly taking the input feature as an external memory, and overcomes Issue (ii) by explicitly employing the attention mechanism.

Concretely, we initialize the memory as $\boldsymbol{C}_0 = \boldsymbol{C}$, which is then sequentially read and written by the recurrent object detector so that all messages from the past t detections are recorded by $\boldsymbol{C}_t$ instead of $\boldsymbol{h}_t$. In iteration t, the detector first reads from the previous memory $\boldsymbol{C}_{t-1}$, then updates its state $\boldsymbol{h}_t$, and finally write new contents into the current memory $\boldsymbol{C}_t$. Thus, in contrast to Equation (2), the recurrent module NN^{rec} has the form:

$$\boldsymbol{C}_t, \boldsymbol{h}_t = \text{NN}^{rec}\left(\boldsymbol{C}_{t-1}; \boldsymbol{\theta}^{rec}\right) \tag{10}$$

We set NN^{rec} defined in Equation (10) as a MRAN, where a location-based addressing is first adopted to explicitly impose attention on the input feature. The attention weight $\boldsymbol{W}_t$ is generated by an attention network NN^{att}:

$$\boldsymbol{W}_t = \text{NN}^{att}\left(\boldsymbol{C}_{t-1}; \boldsymbol{\theta}^{att}\right) \tag{11}$$

where $\boldsymbol{W}_t \in [0,1]^{M \times N}$ satisfies $\sum_{m,n} W_{t,m,n} = 1$ (by using a softmax output layer).

Then, let $\boldsymbol{c}_{t-1,m,n} \in \mathbb{R}^S$ be a feature vector of $\boldsymbol{C}_{t-1}$, we define the read operation as:

$$\boldsymbol{r}_t = \sum_{m,n} W_{t,m,n}\ \boldsymbol{c}_{t-1,m,n} \tag{12}$$

where the read vector $\boldsymbol{r}_t \in \mathbb{R}^S$ represents the attended input features relevant to the current detection.

Next, the detector state is updated through a linear transformation followed by a tanh function, where $\boldsymbol{r}_t$ is taken as the input feature (instead of $\boldsymbol{C}_{t-1}$):

$$\hat{\boldsymbol{h}}_t = \text{Linear}\left(\boldsymbol{r}_t; \boldsymbol{\theta}^{upd}\right) \tag{13}$$

$$\boldsymbol{h}_t = \tanh\left(\hat{\boldsymbol{h}}_t\right) \tag{14}$$

Finally, we use $\boldsymbol{h}_t$ to generate an erase vector $\boldsymbol{e}_{t,i} \in [0,1]^S$ and a write vector $\boldsymbol{v}_{t,i} \in \mathbb{R}^S$:

$$\left(\hat{\boldsymbol{e}}_t, \boldsymbol{v}_t\right) = \text{Linear}\left(\boldsymbol{h}_t; \boldsymbol{\theta}^{wrt}\right) \tag{15}$$

$$\boldsymbol{e}_t = \text{sigmoid}\left(\hat{\boldsymbol{e}}_t\right) \tag{16}$$

and define the write operation as:

$$\boldsymbol{c}_{t,m,n} = (\mathbf{1} - W_{t,m,n}\boldsymbol{e}_t) \odot \boldsymbol{c}_{t-1,m,n} + W_{t,m,n}\boldsymbol{v}_t \tag{17}$$

An illustrations of the MRAN is shown in Figure 3.

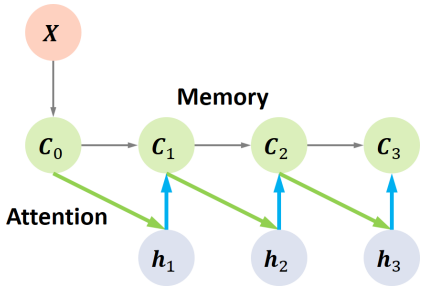

Figure 3. Illustration of the Memory-Based Recurrent Attention Network (MRAN), where the detection step $T = 3$ and the green/blue bold lines denote the attentive read/write operations on the memory.

Although the attention is now imposed on the input feature $\boldsymbol{C}_t$, we would like to further impose it on the input image $\boldsymbol{X}$ so that the detector is only related to a local image region rather than the whole image. Therefore, we use a Fully Convolutional Network (FCN) [23] (only contains convolution layers) as the image encoder NN^{enc}. By controlling the receptive field of $\boldsymbol{c}_{t,m,n}$, $\boldsymbol{X}$ can also be attentively accessed by the detector. Another advantage of using FCN is that, through parameter sharing, it can well-capture the regularity among different objects (they usually have similar patterns).

Our MRAN (Equations (11))–((17)) is similar to the Neural Turing Machine [24,25]. As the detector uses interface variables to interact with the external memory, messages from the previous detections do not need to be encoded into its working memory $\boldsymbol{h}_t$, thereby improving the detection efficiency.

4. Experiments

The goals of our experiments were: (i) to investigate the importance of the layered representation and MRAN in our model (Note that setting a supervised counterpart for our model could be difficult as computing the supervised loss requires finding the best matching between the detector outputs and the ground truth data, which is also an optimization problem); and (ii) to test whether our model is well-suited for video surveillance data taken from cameras. For Goal (i), we created a synthetic dataset, namely Sprites, and were interested in the configurations below:

- **UMOD-MRAN**. UMOD with MRAN, which is our standard model as described in Sections 2 and 3.
- **UMOD-MRAN-noOcc**. UMOD-MRAN without occlusion reasoning, which is achieved by fixing the layer number I to 1.
- **UMOD-RNN**. UMOD with RNN, which is achieved by setting the recurrent module NN^{rec} as a Gated Recurrent Unit [20] as described in Section 2.2, thereby disabling the external memory and attention.
- **AIR**. Our implementation of the generative model proposed in [26] that could be used for MOD through inference.

For Goal (ii), we evaluated UMOD-MRAN on the challenging DukeMTMC dataset [17], and compared results with those of the state-of-the-art.

There are some common settings for the implementation of the above configurations. For the image encoder NN^{rec} defined in Equation (1), we set it as a FCN, where each convolution layer was composed via Convolution→Pooling→ReLU and the convolution stride was set to 1 for all layers. For the decoder NN^{dec} defined in Equation (3), we set it as a Fully-Connected network (FC), where the

ReLU was chosen as its activation function for each hidden layer. We also set the object scale coefficients $\eta^x = \eta^y = 0.4$. For the renderer, we set the image layer $I = 3$ (except for UMOD-MRAN-noOcc and AIR where $I = 1$). For the loss defined in Equation (9), we set $\lambda = 1$. To train the model, we minimized the averaged loss on the training set with respect to all network parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^{enc}, \boldsymbol{\theta}^{rec}, \boldsymbol{\theta}^{dec}\}$ using Adam [27] with a learning rate of 5×10^{-4}. Early stopping was used to terminate training.

4.1. Sprites

As a toy example, we wanted to see whether the model could robustly handle occlusion and infer the object existence, position, scale, shape, and appearance, thereby generating accurate object bounding boxes. Therefore, we created a new Sprites dataset composed of 1 M color images, each of which is of size $128 \times 128 \times 3$, comprising a black background and 0–3 sprites which could occlude each other. Each sprite is a $21 \times 21 \times$ three-color patch with a random scale, position, shape (diamond/rectangle/triangle/circle), and color (cyan/magenta/yellow/blue/green/red).

To deal with the task, for the UMOD configuration's we set the detection step $T = 4$ and the background $\hat{\boldsymbol{X}}_t^{(0)} = \mathbf{0}$. Please refer to Table 1 for other configurations. The mini-batch size was set to 128 for training.

Table 1. Model hyper-parameters for Sprites and DukeMTMC.

Hyper-parameter	**Sprites**		**DukeMTMC**	
$[H, W, D]$	[128, 128, 3]		[108, 192, 3]	
NN^{enc} (FCN)	Kernel size	Layer size	Kernel size	Layer size
	[5, 5]	[64, 64, 32]	[5, 5]	[108, 192, 32]
	[3, 3]	[32, 32, 64]	[5, 3]	[36, 64, 128]
	[1, 1]	[16, 16, 128]	[5, 3]	[18, 32, 256]
	[3, 3]	[8, 8, 256]	[3, 1]	[9, 16, 512]
	[1, 1]	[8, 8, 20]	[1, 1]	[9, 16, 200]
$[M, N, S]$	[8, 8, 20]		[9, 16, 200]	
R	40		400	
NN^{dec} (FC)	$40 \rightarrow 266 \rightarrow 1772$		$400 \rightarrow 578 \rightarrow 836$	
$[U, V, D]$	[21, 21, 3]		[9, 23, 3]	

Figure 4 shows the training curves. UMOD-MRAN and UMOD-MRAN-noOcc converge significantly faster than UMOD-RNN, indicating that, with MRAN, UMOD can be trained more easily. However, for the final validation losses, UMOD-MRAN and UMOD-RNN are slightly better than UMOD-MRAN-noOcc, meaning that, without layered representation which can model occlusion, the input images could not be well-constructed.

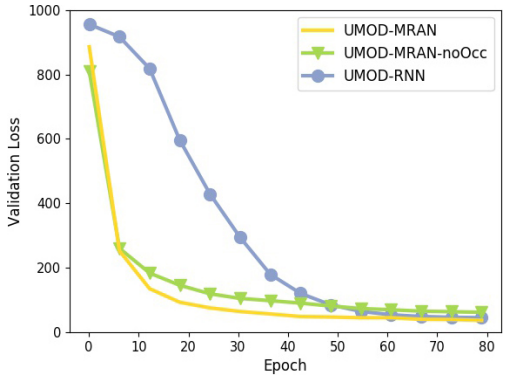

Figure 4. Training curves of different configurations on Sprites (Note that we do not compare the loss of AIR since it uses a different training objective).

To visualize the detection performance, UMOD-MRAN was compared against other configurations on sampled images. Qualitative results are shown in Figure 5. We can see that UMOD-MRAN performs well and can robustly infer the existence, layer, position, scale, shape, and appearance of the object. UMOD-RNN performs slightly worse than UMOD-MRAN since it sometimes fails to recover the occlusion order (Columns 2, 5, and 7). However, with a layer number $I = 1$, UMOD-MRAN-noOcc and AIR perform even worse since they could handle occlusion (Columns 3, 5, 6, and 9), sometimes losing detection (Columns 1, 2, 7, and 8)—we conjecture that the model has learned to suppress the occluded outputs, as adding their pixel values to a single layer probably causes a high reconstruction error.

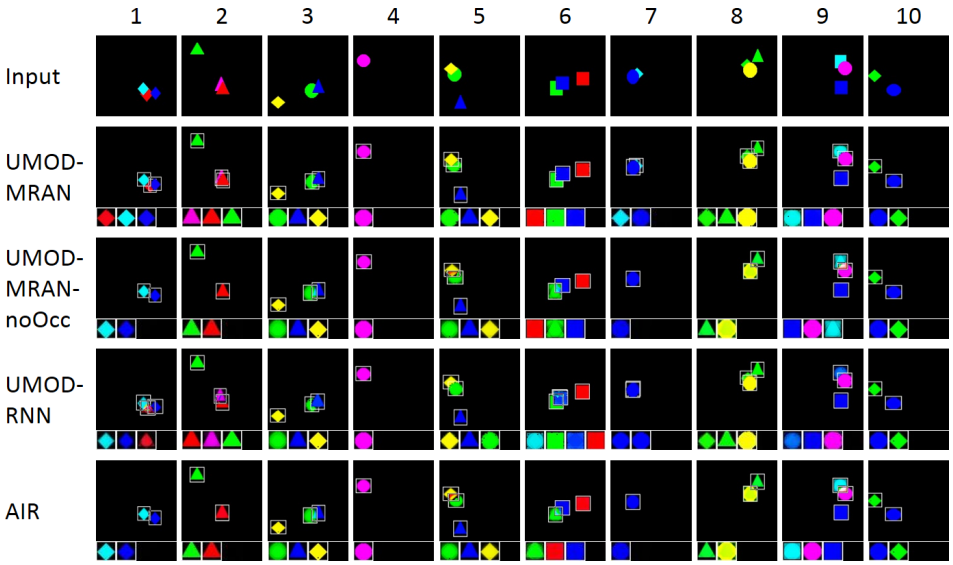

Figure 5. Qualitative results of different configurations on Sprites. For each configuration, the reconstructed images are shown, with the detector outputs on the bottom (produced at detection Steps 1–4 from left to right).

To quantitatively assess the model, we also evaluated different configurations with the commonly used MOD metrics, including the Average Precision (AP) [9], Multi-Object Detection Accuracy (MODA), Multi-Object Detection Precision (MODP) [28], average False Alarm number per Frame (FAF), total True Positive number (TF), total False Positive number (FP), total False Negative number (FN), Precision ($TP/(TP + FP)$), and Recall ($TP/(TP + FN)$). Results are presented in Table 2. UMOD-MRAN outperforms all other configurations with respect to all metrics. Without layered representation, the performances of UMOD-MRAN-noOcc and AIR are largely affected by higher

FNs (1702 and 1964, respectively), which again suggests that the detector outputs are suppressed when only a single image layer is used. Moreover, when the attention and external memory are not explicitly modeled, UMOD-RNN and AIR perform slightly worse than UMOD-MRAN and UMOD-MRAN-noOcc (in all metrics), respectively, which means incorporating these two prior knowledge can well regularize the model so that it can learn to extract more desired outputs.

Table 2. Detection performances of different configurations on Sprites.

Configuration	AP↑	MODA↑	MODP↑	FAF↓	TP↑	FP↓	FN↓	Precision↑	Recall↑
UMOD-MRAN	96.8	95.3	91.5	0.02	21,016	234	807	98.9	96.3
UMOD-MRAN-noOcc	92.7	90.3	90.3	0.04	20,121	415	1702	98.1	92.2
UMOD-RNN	94.5	94.1	90.6	0.03	20,819	284	1004	98.7	95.4
AIR	90.5	88.2	88.6	0.06	19,859	611	1964	97.2	91.0

4.2. DukeMTMC

To examine the performance of our model when applied to real-world data that are highly flexible and complex, we assessed the UMOD-MRAN on the challenging DukeMTMC dataset [17]. It is a video surveillance dataset comprising eight videos with 60 fps and a resolution of 1080 × 1920, which were collected from eight fixed cameras that record people's movements at different places in Duke university. Each video is divided into a training set (50 min), a hard test set (10 min), and an easy test (25 min).

For UMOD-MRAN, the detection step T was set to 10 and the IMBS algorithm [29] was used for extracting the background $\hat{\boldsymbol{X}}_t^{(0)}$. Please see Table 1 for other configurations. For training, we set the mini-batch size to 32. To ease processing, we resized the input images to 108 × 192. We trained a single model and evaluated it on the easy test sets of all scenarios. Note that we did not evaluate our model on the hard test sets as they contain very different data statistics from the training sets.

Qualitative results are shown in Figure 6. We can see that UMOD-MRAN performs well under various scenarios: (i) a few people (Column 4 in Rows 1–3); (ii) many people (Column 5 ins Row 1 and 2); (iii) occluded people (Column 2 in Row 2); (iv) people that are near to the camera (Column 4 in Row 1); (v) people that are far from the camera (Column 6 in Row 2); (vi) people with different shapes/appearances (Column 8 in Row 1); and (vii) people that are hard to be distinguished from the background (Column 6 in Row 1).

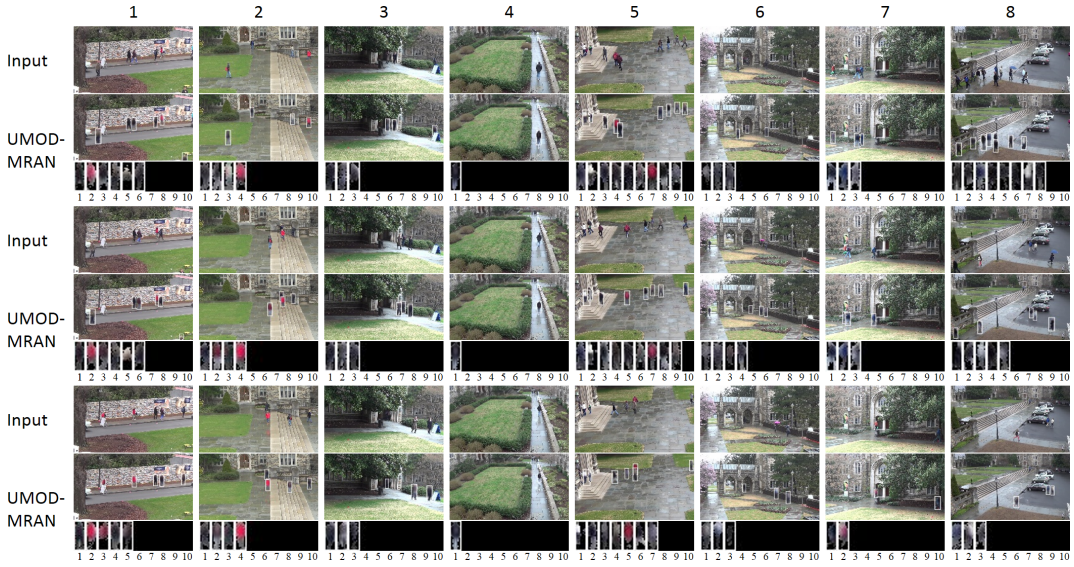

Figure 6. Qualitative results in different scenarios on DukeMTMC.

Table 3 reports the quantitative results. UMOD-MRAN outperforms the DPM [1] with respect to all metrics. It reaches an AP of 87.2%, which is significantly higher than that of the DPM (79.3% AP), and is also very competitive to the recently proposed Faster R-CNN [5] (89.7% AP). Although the CRAFT [30,31] perform the best (with 91.1% and 92.0% APs, respectively), our model is the first one free of any training labels or extracted features.

Table 3. Detection performance of the UMOD-MRAN compared with those of the state-of-the-art methods on DukeMTMC.

Method	AP↑	MODA↑	MODP↑	FAF↓	TP↑	FP↓	FN↓	Precision↑	Recall↑
DPM [1]	79.3	66.9	83.4	0.32	108,050	22,445	19,820	82.8	84.5
Faster R-CNN [5]	89.7	82.1	87.5	0.13	114,443	9,413	13,427	92.4	89.5
CRAFT [30]	91.1	83.4	89.8	0.12	115,850	8,586	12,020	93.1	90.6
RRC [31]	92.0	84.3	90.7	0.13	116,873	9,068	10,997	92.8	91.4
UMOD-MRAN (ours)	87.2	78.7	85.3	0.15	111,247	10,601	16,623	91.3	87.0

4.3. Visualizing the UMOD-MRAN

To further understand the model, we visualize the inner working of the UMOD-MRAN on Sprites, as shown in Figure 7. Both the memory $\boldsymbol{C}_t$ and the attention weight $\boldsymbol{W}_t$ are visualized as $M \times N$ (8×8) matrices (brighter pixels indicate higher values), where for $\boldsymbol{C}_t$ the matrix consists of its mean values along the last dimension. The detector output $\mathcal{Y}_t$ is visualized as $(y_t^c \boldsymbol{Y}_t^s \odot \boldsymbol{Y}_t^a) \in [0,1]^{U \times V \times D}$. At detection step t, the memory $\boldsymbol{C}_{t-1}$ produces an attention weight $\boldsymbol{W}_t$, through which the detector first reads from $\boldsymbol{C}_{t-1}$ and then writes to $\boldsymbol{C}_t$. We can find that at each detection step, the memory content (bright region on $\boldsymbol{C}_{t-1}$) related to the associated object ($\mathcal{Y}_t$) is *erased* (becomes dark) by the write operation, thereby preventing the detector from reading it again in the next detection step.

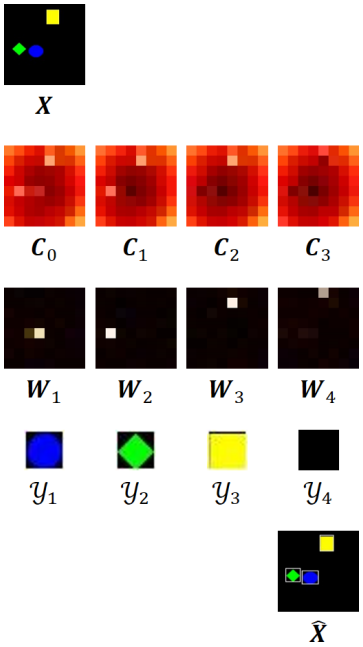

Figure 7. Visualization of the UMOD-MRAN on Sprites, where the detection step $T = 4$.

5. Related Work

Recently, unsupervised learning has been used in some works to extract desired patterns from images. For example, Kulkarni et al. [32], Chen et al. [33] and Rolfe [34] focused on finding lower-level disentangled factors; Le Roux et al. [35], Moreno et al. [36] and Huang and Murphy [37] focused on extracting mid-level semantics; and Eslami et al. [26], Yan et al. [38], Rezende et al. [39], Stewart and

Ermon [40] and Wu et al. [41] focused on discovering higher-level semantics. However, unlike these methods, the proposed UMOD-MRAN focuses on MOD tasks. It uses a novel rendering scheme to handle occlusion and integrates memory and attention to improve the efficiency, being well-suited for real applications such as video surveillance.

6. Conclusions

In this paper, we propose a novel UMOD framework to tackle the MOD task for video surveillance. The main advantage of our model over other popular methods is that it is free of any training labels or extracted features. Another important advantage of our model is that the MRAN module can largely improve the detection efficiency and ease model training. The proposed model was evaluated on both synthetic and real datasets, exhibiting its superiority and practicality.

For future work, we would like to extend our model in two aspects. First, it is useful to incorporate the idea of "adaptive computation time" [42] into our framework so that the recurrent object detector can adaptively choose an appropriate detection step T for efficiency. Second, it is intriguing to model the object dynamics by employing temporal RNNs so that our model can directly deal with multi-object tracking problems for video surveillance.

Author Contributions: Methodology, Z.H. and H.H.; Software, Z.H.; Validation, Z.H.; Investigation, Z.H.; Writing—Original Draft Preparation, Z.H.; Writing—Review & Editing, Z.H. and H.H.; Visualization, Z.H.; Supervision, H.H.; Project Administration, Z.H. and H.H.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Felzenszwalb, P.F.; Girshick, R.B.; McAllester, D.; Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 1627–1645. [CrossRef] [PubMed]
2. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
3. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
4. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Washington, DC, USA, 7–13 December 2015; pp. 1440–1448
5. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in neural information processing systems, Montreal, DC, Canada, 7–12 December 2015; pp. 91–99.
6. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2980–2988.
7. Enzweiler, M.; Gavrila, D.M. Monocular pedestrian detection: Survey and experiments. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 2179–2195. [CrossRef] [PubMed]
8. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F.-F. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009.
9. Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
10. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The KITTI dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [CrossRef]
11. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2014; pp. 740–755.
12. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation applied to handwritten zip code recognition. *Neural Comput.* **1989**, *1*, 541–551. [CrossRef]

13. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
14. Huang, L.; Yang, Y.; Deng, Y.; Yu, Y. Densebox: Unifying landmark localization with end to end object detection. *arXiv* **2015**, arXiv:1509.04874.
15. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
16. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2016; pp. 21–37.
17. Ristani, E.; Solera, F.; Zou, R.; Cucchiara, R.; Tomasi, C. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2016; pp. 17–35.
18. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
19. Gers, F.A.; Schmidhuber, J.; Cummins, F. Learning to forget: Continual prediction with LSTM. *Neural Comput.* **2000**, *12*, 2451–2471. [CrossRef] [PubMed]
20. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
21. Jang, E.; Gu, S.; Poole, B. Categorical Reparameterization with Gumbel-Softmax. *arXiv* **2016**, arXiv:1611.01144.
22. Jaderberg, M.; Simonyan, K.; Zisserman, A. Spatial transformer networks. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 2017–2025.
23. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
24. Graves, A.; Wayne, G.; Danihelka, I. Neural turing machines. *arXiv* **2014**, arXiv:1410.5401.
25. Graves, A.; Wayne, G.; Reynolds, M.; Harley, T.; Danihelka, I.; Grabska-Barwińska, A.; Colmenarejo, S.G.; Grefenstette, E.; Ramalho, T.; Agapiou, J.; et al. Hybrid computing using a neural network with dynamic external memory. *Nature* **2016**, *538*, 471–476. [CrossRef] [PubMed]
26. Eslami, S.A.; Heess, N.; Weber, T.; Tassa, Y.; Szepesvari, D.; Hinton, G.E. Attend, infer, repeat: Fast scene understanding with generative models. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 3225–3233.
27. Kingma, D.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
28. Stiefelhagen, R.; Bernardin, K.; Bowers, R.; Garofolo, J.; Mostefa, D.; Soundararajan, P. The CLEAR 2006 evaluation. In *International Evaluation Workshop on Classification of Events, Activities and Relationships*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 1–44.
29. Bloisi, D.; Iocchi, L. Independent multimodal background subtraction. In *CompIMAGE*; CRC Press: Boca Raton, FL, USA, 2012.
30. Yang, B.; Yan, J.; Lei, Z.; Li, S.Z. Craft objects from images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 6043–6051.
31. Ren, J.; Chen, X.; Liu, J.; Sun, W.; Pang, J.; Yan, Q.; Tai, Y.-W.; Xu, L. Accurate single stage detector using recurrent rolling convolution. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
32. Kulkarni, T.D.; Whitney, W.F.; Kohli, P.; Tenenbaum, J. Deep convolutional inverse graphics network. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 2539–2547.
33. Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 2172–2180.
34. Rolfe, J.T. Discrete variational autoencoders. *arXiv* **2016**, arXiv:1609.02200.
35. Le Roux, N.; Heess, N.; Shotton, J.; Winn, J. Learning a generative model of images by factoring appearance and shape. *Neural Comput.* **2011**, *23*, 593–650. [CrossRef] [PubMed]
36. Moreno, P.; Williams, C.K.; Nash, C.; Kohli, P. Overcoming occlusion with inverse graphics. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 170–185.

37. Huang, J.; Murphy, K. Efficient inference in occlusion-aware generative models of images. *arXiv* **2015**, arXiv:1511.06362.
38. Yan, X.; Yang, J.; Yumer, E.; Guo, Y.; Lee, H. Perspective transformer nets: Learning single-view 3d object reconstruction without 3D supervision. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 1696–1704.
39. Rezende, D.J.; Eslami, S.A.; Mohamed, S.; Battaglia, P.; Jaderberg, M.; Heess, N. Unsupervised learning of 3D structure from images. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 4996–5004.
40. Stewart, R.; Ermon, S. Label-free supervision of neural networks with physics and domain knowledge. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
41. Wu, J.; Tenenbaum, J.B.; Kohli, P. Neural scene de-rendering. In Proceedings of the Computer Vision Foundation, Honolulu, HI, USA, 21–26 July 2017.
42. Graves, A. Adaptive computation time for recurrent neural networks. *arXiv* **2016**, arXiv:1603.08983.