Signal, Image and Video Processing (2019) 13:1629–1637
https://doi.org/10.1007/s11760-019-01506-4

ORIGINAL PAPER

Target tracking and classification using compressive sensing camera for SWIR videos

Chiman Kwan[1] · Bryan Chou[1] · Jonathan Yang[1] · Akshay Rangamani[2] · Trac Tran[2] · Jack Zhang[3] · Ralph Etienne-Cummings[2]

Received: 26 November 2018 / Revised: 29 January 2019 / Accepted: 27 May 2019 / Published online: 7 June 2019

Abstract
The pixel-wise code exposure (PCE) camera is a compressive sensing camera that has several advantages, such as low power consumption and high compression ratio. Moreover, one notable advantage is the capability to control individual pixel exposure time. Conventional approaches of using PCE cameras involve a time-consuming and lossy process to reconstruct the original frames and then use those frames for target tracking and classification. Otherwise, conventional approaches will fail if compressive measurements are used. In this paper, we present a deep learning approach that directly performs target tracking and classification in the compressive measurement domain without any frame reconstruction. Our approach has two parts: tracking and classification. The tracking has been done via detection using You Only Look Once (YOLO), and the classification is achieved using residual network (ResNet). Extensive simulations using short-wave infrared (SWIR) videos demonstrated the efficacy of our proposed approach.

Keywords Compressive measurement · Pixel-wise code exposure (PCE) camera · Multi-target tracking and classification · SWIR

1 Introduction

Target tracking using radar [1, 2], optical [3–10], and infrared sensors [11] has found widespread usage in many applications. In the above applications, the sensors are normally in their original format. In the past decade, compressive sensing has gained popularity in various applications. Compressive measurements [12] are normally collected by multiplying the original vectorized image with a Gaussian random matrix. Each measurement contains a scalar value, and the measurement is repeated M times where M is much fewer than N (the number of pixels). To track a target using compressive measurements, it is normally done by reconstructing the image scene and then conventional trackers are then applied.

Recently, a new compressive sensing device known as the pixel-wise code exposure (PCE) camera was proposed [13]. A hardware prototype was developed, and performance in terms of power consumption and compression ratio was demonstrated. In [13], the original frames were reconstructed using L_1 [14] or L_0 [15–17] sparsity-based algorithms. One problem with the reconstruction-based approach is that it is extremely time-consuming to reconstruct the original frames, and hence, this may prohibit real-time applications. Moreover, information may be lost in the reconstruction process [18]. For target tracking and classification applications, it will be ideal if one can carry out target tracking and classification directly in the compressive measurement domain. Although there are some tracking papers [19] in the literature that appear to be using compressive measurements, they are actually still using the original video frames for tracking. A target tracking algorithm that truly uses compressive measurement directly is the paper [20], which uses the subsampling operator to generate the compressive measurements. ResNet was used for both tracking and classification. One limitation is that the initial target bounding boxes are still needed.

In this paper, we propose a target tracking and classification approach in the compressive measurement domain. First, a YOLO tracker is used for target tracking, which is done by object detection. The training of YOLO tracker is

✉ Chiman Kwan
chiman.kwan@arllc.net

1 Applied Research LLC, Rockville, MD, USA
2 Johns Hopkins University, Baltimore, MD, USA
3 MIT, Cambridge, MA, USA

very simple, which requires image frames with known target locations. Although YOLO can also perform classification, the performance is not good as we have very limited number of video frames for training. So, in the second step of target classification, we decided to use ResNet for classification. That is, the object locations detected by YOLO are fed into the ResNet for classification. We chose ResNet because it allows us to perform customized training by augmenting the data from the limited video frames. Our proposed approach was demonstrated using two short-wave infrared (SWIR) videos. The tracking and classification results are reasonable. This is a big improvement over conventional trackers [4, 5], which do not work well in the compressive measurement domain.

There are two key contributions of our paper. First, we are the first ones to apply latest deep learning techniques to target tracking and classification *directly in the compressive measurement domain*, which does not require any time-consuming image reconstruction. This will allow fast and near real-time target tracking using compressive measurements. Actually, real-time experiments have been carried out and we will report the results in another paper. Second, we improved the target classification performance with a customized ResNet, which yielded much better performance than that of the built-in classifier in YOLO.

This paper is organized as follows. In Sect. 2, we describe some background materials, including the PCE camera, YOLO, and ResNet. In Sect. 3, we summarize the tracking and classification results using SWIR videos. Finally, we conclude our paper with some remarks for future research.

2 Background

2.1 PCE imaging and coded aperture

In this paper, we employ a sensing scheme based on PCE, also known as coded aperture (CA) video frames, as described in [13]. Figure 1 illustrates the differences between a conventional video sensing scheme and PCE, where random spatial pixel activation is combined with fixed temporal exposure duration. First, conventional cameras capture frames at certain frame rates, such as 30 frames per second. In contrast, the PCE camera captures a compressed frame called motion coded image over a fixed period of time (T_v). For example, a user can compress 30 conventional frames into a single motion coded frame. This will yield significant data compression ratio. Second, the PCE camera allows a user to use different exposure times for different pixel locations. For low lighting regions, more exposure times can be used and for strong light areas, short exposure can be exerted. This will allow high dynamic range. Moreover, power can also be saved via low sampling rate in the data acquisition process. As shown in Fig. 1, one conventional approach to using the motion coded images is to apply sparse reconstruction to reconstruct the original frames and this process may be very time-consuming.

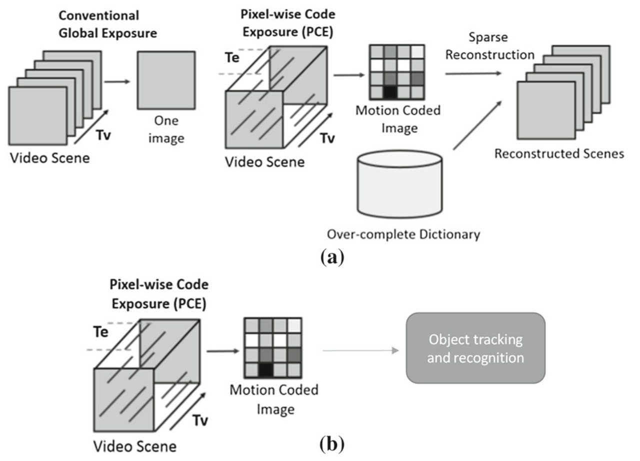


Fig. 1 **a** Conventional camera vs. pixel-wise coded exposure (PCE) compressed image/video sensor [13]; **b** object tracking and recognition directly using motion coded images

Suppose the video scene is contained in a data cube $\mathbf{X} \in \mathbf{R}^{M \times N \times T}$ where $M \times N$ is the image size and T is the number of frames. A sensing data cube is defined by $\mathbf{S} \in \mathbf{R}^{M \times N \times T}$ which contains the exposure times for pixel located at (m, n, t). The value of $\mathbf{S}(m,n,t)$ is 1 for frames $t \in \leftarrow [tstart, t_{\text{end}}]$ and 0 otherwise. $[t_{\text{start}}, t_{\text{end}}]$ denotes the start and end frame numbers for a particular pixel.

The measured coded aperture image $\mathbf{Y} \in \mathbf{R}^{M \times N}$ is obtained by

$$\mathbf{Y}(m, n) = \sum_{t=1}^{T} \mathbf{S}(m, n, t) \cdot \mathbf{X}(m, n, t) \tag{1}$$

The original video scene $\mathbf{X} \in \mathbf{R}^{M \times N \times T}$ can be reconstructed via sparsity methods (L_1 or L_0). Details can be found in [13].

Instead of doing sparse reconstruction on PCE images or frames, our scheme directly acts on the PCE or coded aperture images, which contain raw sensing measurements without the need for any reconstruction effort. Utilizing raw measurements has several challenges. First, moving targets may be smeared if the exposure times are long. Second, there are also missing pixels in the raw measurements because not all pixels are activated during the data collection process. Third, there are much fewer frames in the raw video because many original frames are compressed into a single coded frame.

In this study, we have focused our effort into simulating EXACTLY the measurements that should be produced by the PCE-based compressive sensing (CS) sensor. We then proceed to show that detecting, tracking, and even classifying moving objects of interest in the scene is entirely feasible with a minor sacrifice in discrimination accuracy. We carried out multiple experiments with three diverse sensing models: *PCE/CA full*, *PCE/CA 50%*, and *PCE/CA 25%*.

The PCE full model (*PCE full* or *CA full*) is quite similar to a conventional video sensor: every pixel in the spatial scene is exposed for exactly the same duration of one second. Obviously, all activities within that 1-second window are captured in our PCE image/frame. However, motion is expected to be blurred significantly. This simple model still produces a compression ratio of 30:1. However, there is not much saving in sensing power since all pixels are exposed at all times.

Next, we set the sensing model labeled as *PCE 50%* or *CA 50%* using the following set of parameters. For each frame, there are roughly 1.85% pixels being activated. The exposure time is $T_e = 133.3$ ms. Therefore, each exposed pixel stays continuously active for 4-frame duration. In short, we output ONE coded aperture image for every group of 30 frames, resulting in a temporal sensing ratio of 1 frame per second (fps) or equivalently 30:1 compression ratio in terms of frame rate. In every frame, a new set of pixels that have not been activated yet would be selected for activation. Once activated, each pixel would have exactly the same exposure duration. This scheme results in 50% of the pixels locations being captured in various time-stamps within one sensing period (1 s), resulting in a single coded aperture image or PCE frame with 50% activated pixels for every 30 conventional video frames. The PCE 50% Model yields a data saving ratio of $\frac{1}{30} \times \frac{1}{2} = \frac{1}{60}$ and a power saving ratio of $\frac{1}{60} \times 4 = \frac{1}{15}$.

For the *PCE 25%* or *CA 25%* model, we further decrease the percentage of pixels activated per frame so that the final output PCE frame contains only 25% of randomly activated pixel locations. The exposure duration is still set at the same conventional 4-frame duration. A simple way to generate PCE 25% data is to randomly ignored half of the measurements collected from the PCE 50% Model. The PCE 25% model yields a data saving ratio of $\frac{1}{30} \times \frac{1}{2} = \frac{1}{120}$ and a power saving ratio of $\frac{1}{120} \times 4 = \frac{1}{30}$. Note that we can easily reduce the sensing power by limit ourselves to much shorter exposure duration. This might be advantageous for tracking fast-moving objects at the expense of noisier measurements at low-light conditions.

Table 1 summarizes the comparison between the three sensing models.

A small portion of the sensing mask in 3-dimensional spatiotemporal space for the PCE 50% model is shown in Fig. 2. Colored dots denote nonzero entries (activated pixels being exposed), whereas white part of the spatiotemporal cube is all zero (these pixels are staying dormant). The horizontal axis is the time domain, and the reader is reminded that each exposed pixel stays active for an equivalent duration of 4 continuous frames.

Table 1 Comparison in data compression ratio and power saving ratio between three sensing models

	PCE full/CA full	PCE 50%/CA 50%	PCE 25%/CA 25%
Data saving ratio	30:1	60:1	120:1
Power Saving ratio	1:1	15:1	30:1

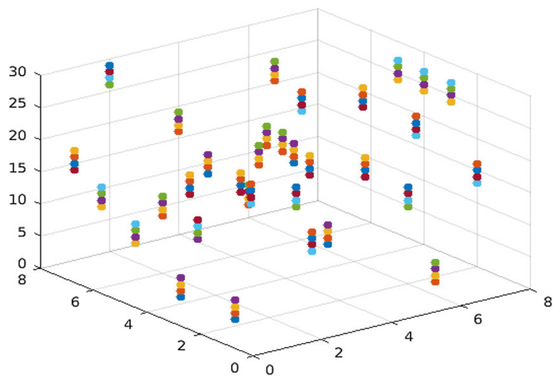


Fig. 2 Example of part of sensing mask. Colored dots denote nonzero entries (activated pixels), whereas white part of the cube is all zero (dormant pixels)

2.2 YOLO tracker

We used the so-called tracking by detection approach. In the target tracking literature, there are several ways to tracking. Some trackers such as STAPLE [4] or GMM [5] require an operator to put a bounding box on a specific target, and then, the trackers will try to track this initial target in subsequent frames. The limitation of this type of trackers is that they can track one target at a time. Another limitation is that they cannot track multiple targets simultaneously. Other trackers such as YOLO and Faster R-CNN do not require initial bounding boxes and can simultaneously detect objects. We can call the second type of trackers: tracking by detection. That is, based on detection results, we determine the vehicle locations in all the frames.

The YOLO tracker [21] is fast and has similar performance to the Faster R-CNN [22]. The input image is resized to 448 × 448. Figure 3 shows the architecture of YOLO version 1. There are 24 convolutional layers and 2 fully connected layers. The output is 7 × 7 × 30. We have used YOLOv2 because it is more accurate than YOLO version 1. The training of YOLO is quite simple. Images with ground-truth target locations are needed. The bounding box for each vehicle was manually determined using tools in MATLAB. For YOLO,

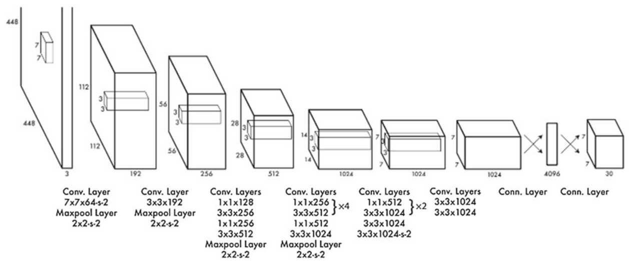

Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

Fig. 3 24 convolutional layers followed by 2 fully connected layers for YOLO version 1 [21]

the last layer of the deep learning model was re-trained. We did not change any of the activation functions. YOLO took approximately 2000 epochs to train.

YOLO also comes with a classification module. However, based on our evaluations, the classification accuracy using YOLO is not good as can be seen in Sect. 3. This is perhaps due to a lack of training data.

2.3 ResNet classifier

As mentioned earlier, YOLO's built-in classifier did not perform well, which is probably because we have limited training data. Moreover, we think that, although YOLO is good for object detection, its built-in classifier is probably more suitable for inter-class (humans, vehicles, bikes, etc.) discrimination and not good for intra-class (Frontier vs. Ram) discrimination. The ResNet-18 model is an 18-layer convolutional neural network (CNN) that has the advantage of avoiding performance saturation and/or degradation when training deeper layers, which is a common problem among other CNN architectures. The ResNet-18 model avoids the performance saturation by implementing an identity shortcut connection, which skips one or more layers and learns the residual mapping of the layer rather than the original mapping. Figure 4 shows the architecture of an 18-layer ResNet.

It is necessary to explain the relationship between YOLO and ResNet. YOLO was used to determine where, in each frame, the trucks were located. YOLO generated bounding boxes for those trucks and that data were used to crop the trucks from the image. The cropped trucks would be fed into the ResNet-18 for classification, and classification results were generated. To be more specific, ResNet-18 is used directly after bounding box information is obtained from YOLO.

Training of ResNet requires target patches. The targets are cropped from training videos. Mirror images are then created. We then perform data augmentation using scaling (larger and smaller), rotation (every 45 degrees), and illumination (brighter and dimmer) to create more training data. For each cropped target, we are able to create a data set with 64

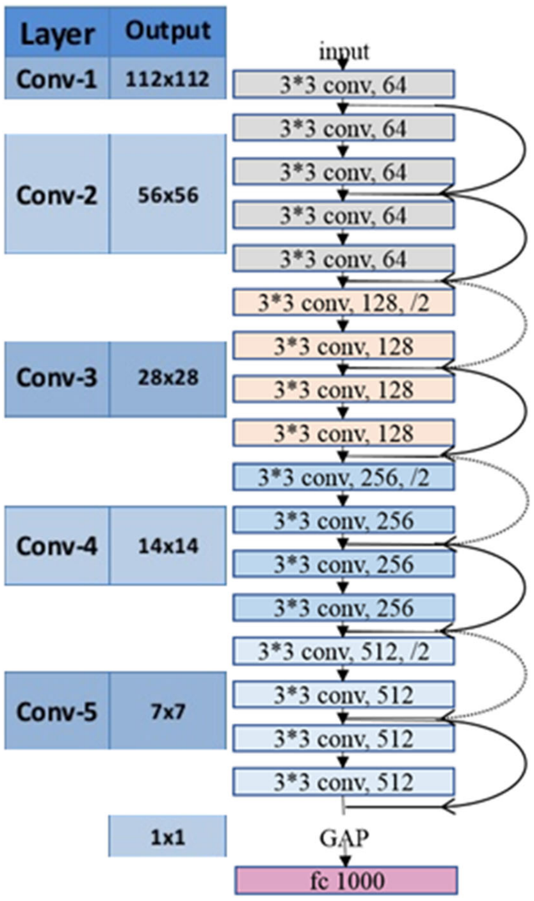


Fig. 4 Architecture of ResNet-18 [23]

more images. For ResNet, the last layer of the deep learning model was re-trained. The ResNet model was trained until the validation score plateaued.

3 Tracking and classification results using SWIR videos

This study focuses on the case of tracking and classification using a combination of YOLO and ResNet for coded aperture cameras. There are three cases. PCE full refers the compression of 30 frames to 1 with no missing pixels. PCE 50 is the case where we compress 30 frames to 1 and at the same time, only 50% of pixels are activated for a length of 4/30 s. PCE 25 is similar to PCE 50 except that only 25% of the pixels are activated for 4/30 s.

We have two SWIR videos. Each one has close to 3000 frames. Each frame has a dimension of 512 by 640. One video (Video 4) starts with vehicles (Ram, Frontier, and Silverado) leaving a parking lot and moves on to a remote location. Another video (Video 5) is just the opposite. The two videos are difficult for tracking and classification because 1) the camera moves to follow the targets; 2) the target sizes change; 3) the target orientations also vary significantly; 4) the illuminations in different videos are also different.

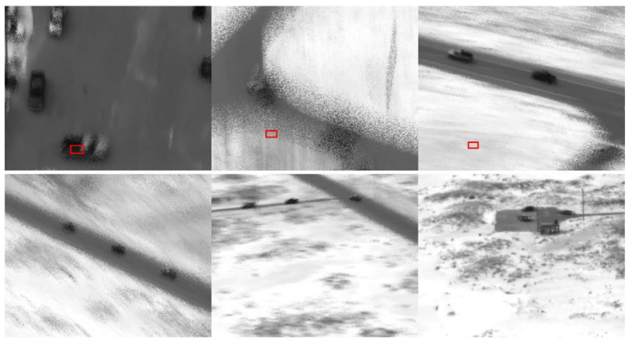
3.1 Tracking results

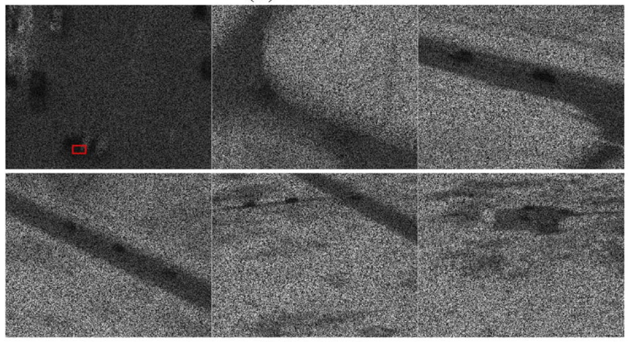
3.1.1 Conventional tracker results

We used the following metrics for evaluating the YOLO tracker performance:

- *Center location error (CLE)* It is the error between the center of the bounding box and the ground-truth bounding box.
- *Distance precision (DP)* It is the percentage of frames where the centroids of detected bounding boxes are within 20 pixels of the centroid of ground-truth bounding boxes.
- *EinGT* It is the percentage of the frames where the centroids of the detected bounding boxes are inside the ground-truth bounding boxes.
- *Number of frames with detection* This is the total number of frames that have detection.
- *Mean average precision (mAP)* Following the definition in [24], it is the mean precision for all frames.
- *Frames per second (FPS)* It is the total number of frames divided by the total execution time. We used a PC with Intel i7-4790 and a GeForce GTX Titan GPU.

We first present some tracking results using a conventional tracker known as STAPLE [4]. STAPLE requires the target location to be known in the first frame. After that, STAPLE learns the target model online and tracks the target. However, in all three cases (PCE full, PCE 50, and PCE 25) as shown in Fig. 5, STAPLE was not able to track any targets in subsequent frames. This shows the difficulty of target tracking using PCE cameras. In order to compare with the YOLO tracker results later, we include Tables 2 and 3, which contain various tracking metrics for the STAPLE tracker. We have the following observations from the two tables. First, DP, EinGT, and mAP drop with higher compression rates for Video 5 and those metrics also have low scores for Video 4. Second, the CLE values are very large as compared to those YOLO results in Tables 4 and 5. For instance, some CLE values are 152, 299, etc., which mean the bounding boxes are mostly outside the frames and tracking performance is very poor. Third, the FPS numbers are almost the same.


(a) PCE full


(b) PCE 50

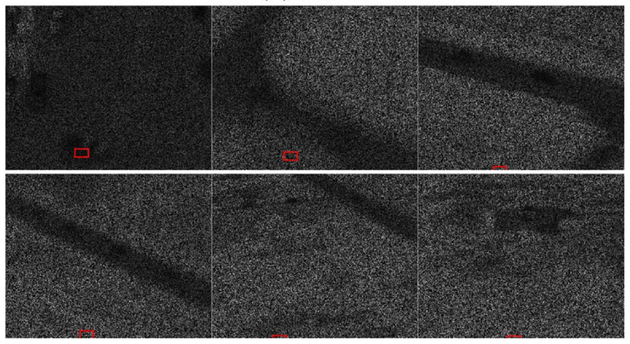

(c) PCE 25

Fig. 5 STAPLE tracking results for the PCE full case. Frames: 10, 30, 50, 70, 90, 110 are shown here. Video 4

when image compression increases, the detection performance drops accordingly. This can be corroborated in the snapshots shown in Fig. 6 where we can see that some targets do not have bounding boxes around them in the high compression cases. The FPS value for PCE full is 18.43, but drops to low values for high compression cases. The reason is because there are more missing pixels in PCE 50 and PCE 25 and YOLO needs more time to generate decisions when there are missing pixels in the images.

3.1.2 YOLO results: train using video 4 and test using video 5

Table 4 shows the tracking results for PCE full, PCE 50, and PCE 25, respectively. In each table, the last column is the number of frames with detection. The trend is that

3.1.3 YOLO results: train using video 5 and test using video 4

Table 5 shows the tracking results for PCE full, PCE 50, and PCE 25, respectively. The trend is that when the image compression increases, the performance drops accordingly.

Table 2 STAPLE tracking metrics for PCE full, PCE 50, and PCE 25. Video 5

	CLE	DP	EinGT	mAP	FPS
	(a) PCE full				
Ram	59.30	0.48	0.37	0.63	8.07
Frontier	82.87	0.54	0.27	0.61	8.14
Silverado	92.09	0.40	0.25	0.58	7.98
Average	78.09	0.47	0.30	0.61	8.06
	(b) PCE 50				
Ram	106.71	0.30	0.10	0.34	7.82
Frontier	84.38	0.33	0.08	0.38	7.80
Silverado	232.63	0.12	0.04	0.11	7.94
Average	141.24	0.25	0.07	0.28	7.85
	(c) PCE 25				
Ram	121.58	0.29	0.06	0.27	7.76
Frontier	132.63	0.18	0.10	0.24	7.98
Silverado	202.98	0.13	0.12	0.12	7.90
Average	152.40	0.20	0.09	0.21	7.88

Table 3 STAPLE tracking metrics for PCE full, PCE 50, and PCE 25. Video 4

	CLE	DP	EinGT	mAP	FPS
	(a) PCE full				
Ram	1053.50	0.09	0.05	0.13	7.75
Frontier	774.61	0.15	0.11	0.25	7.80
Silverado	276.41	0.09	0.04	0.12	8.01
Average	701.51	0.11	0.07	0.17	7.85
	(b) PCE 50				
Ram	348.70	0.08	0.08	0.12	7.29
Frontier	330.72	0.12	0.04	0.13	7.32
Silverado	218.43	0.09	0.03	0.12	7.49
Average	299.28	0.10	0.05	0.12	7.37
	(c) PCE 25				
Ram	330.83	0.08	0.03	0.12	7.35
Frontier	228.76	0.13	0.06	0.16	7.47
Silverado	204.16	0.09	0.03	0.10	7.28
Average	254.58	0.10	0.04	0.13	7.37

Table 4 YOLO tracking metrics for PCE full, PCE 50, and PCE 25. Train using Video 4 and test using Video 5

	CLE	DP	EinGT	mAP	FPS	Frames with detection
	(a) PCE full					
Ram	5.59	1	1	0.71	18.43	62/89
Frontier	5.77	1	1	0.73	18.43	75/89
Silverado	4.89	1	1	0.71	18.43	55/89
Average	5.42	1	1	0.72	18.43	64/89
	(b) PCE 50					
Ram	6.13	0.98	1	0.63	5.47	62/89
Frontier	7.07	1	1	0.57	5.47	72/89
Silverado	5.82	1	1	0.58	5.47	53/89
Average	6.34	0.99	1	0.59	5.47	62/89
	(c) PCE 25					
Ram	6.68	0.98	1	0.73	5.91	51/89
Frontier	7.28	0.98	1	0.59	5.91	63/89
Silverado	6.69	1	0.93	0.5	5.91	28/89
Average	6.88	0.99	0.98	0.61	5.91	47/89

Table 5 Tracking metrics for PCE full. Train using Video 5 and test using Video 4

	CLE	DP	EinGT	mAP	FPS	Frames with detection
	(a) PCE full					
Ram	5	1	1	0.71	34.17	90/110
Frontier	5	1	1	0.8	34.17	101/110
Silverado	5.01	1	1	0.86	34.17	57/110
Average	5.00	1	1	0.79	34.17	83/110
	(b) PCE 50					
Ram	5.86	1	1	0.57	8.62	93/110
Frontier	5.69	1	1	0.69	8.62	102/110
Silverado	5.09	1	1	0.76	8.62	54/110
Average	5.55	1	1	0.67	8.62	83/110
	(c) PCE 25					
Ram	7.2	0.97	1	0.51	9.07	79/110
Frontier	6.28	1	1	0.61	9.07	94/110
Silverado	5.32	1	1	0.59	9.07	44/110
Average	6.27	0.99	1	0.57	9.07	72/110

For instance, the averaged mAP values drop from 0.79 in the PCE full case to 0.57 in the PCE 25 case. This can be confirmed in the snapshots shown in Fig. 7 where we can see that some targets do not have bounding boxes around them in the high compression cases.

It should be noted that the YOLO results such as CLE values are much better than those STAPLE results shown in Tables 2 and 3.

3.2 Classification results

Here, we want to compare two classifiers: YOLO and ResNet. It should be noted that classification is performed only when there are good detection results from the YOLO tracker. For some frames in the PCE 50 and PCE 25, there may not be positive detection results and, for those frames, we do not generate any classification results.

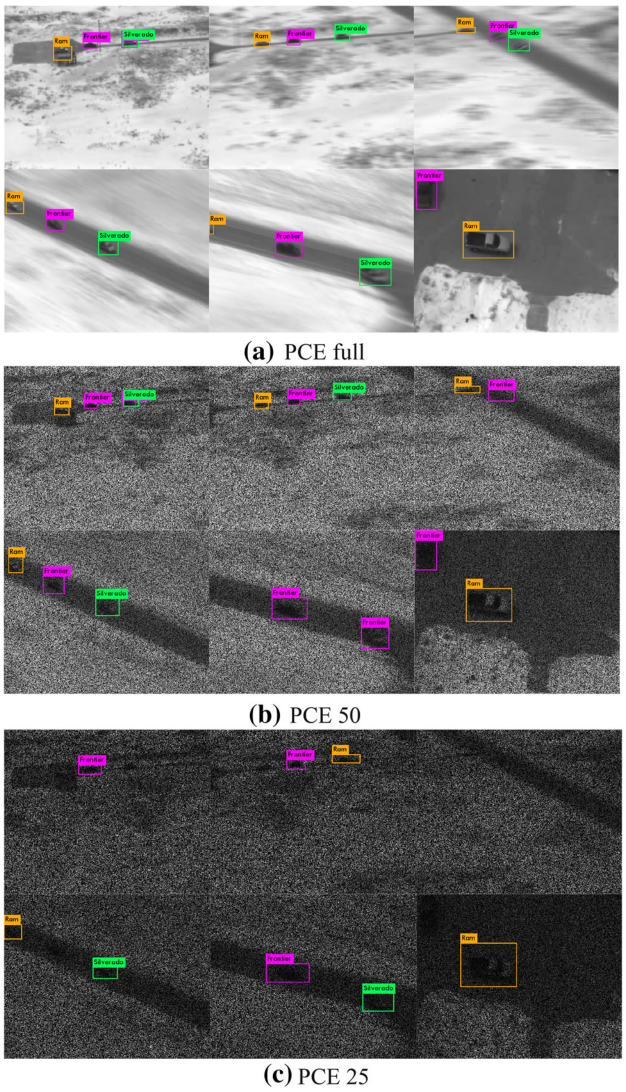


Fig. 6 Tracking results for frames 1, 16, 31, 45, 60, and 89. PCE full case. Train using Video 4 and test using Video 5

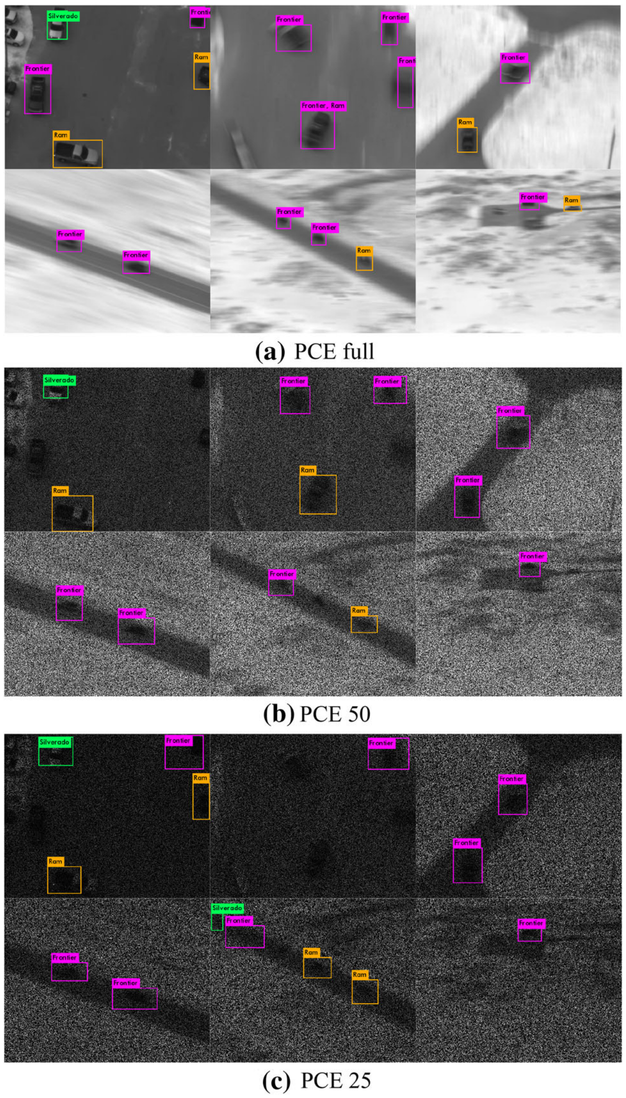


Fig. 7 Tracking results for frames 1, 16, 31, 45, 60, and 89. PCE full case. Train using Video 5 and test using Video 4

3.2.1 Training using video 4 and testing using video 5

Tables 6, 7, and 8 show the classification results using YOLO and ResNet. In each table, the second to fourth columns contain the confusion matrix and the last column is the classification rate. The first observation is that the ResNet performance is better than that of YOLO in most cases. The second observation is that the classification performance deteriorates with high missing rates. The third observation is that Ram and Silverado have lower classification rates. This is because Ram and Silverado have similar appearances. A fourth observation is that the results in Table 8 appear to be better than other cases. This may be misleading, as the classification is done only for frames with good detection.

3.2.2 Training using video 5 and testing using video 4

The observations are similar to the earlier case as shown in Tables 9, 10, 11. That is, ResNet is better than YOLO and classification performance drops with high compression rates.

3.3 Discussions

If one is interested in only target tracking, then, even for the PCE 25 camera (120 times compression), the YOLO tracker can achieve 53% (47/89) and 65% (72/110) of correct detection for Train 4 Test 5 (Table 4c) and Train 5 Test 4 cases (Table 5c), respectively. This is a decent result.

However, if one is more interested in target classification, then PCE 25 is not good enough even using ResNet because

Table 6 Classification results for PCE full case. Video 4 for training and Video 5 for testing

	Ram	Frontier	Silverado	Classification accuracy
	(a) YOLO classifier outputs			
Ram	14	2	45	0.2295
Frontier	13	60	2	0.8
Silverado	53	0	2	0.0364
Average classification accuracy				0.3553
	(b) ResNet classifier outputs			
Ram	38	16	8	0.6129
Frontier	7	68	0	0.9067
Silverado	12	8	35	0.6364
Average classification accuracy				0.7187

Table 7 Classification results for PCE 50 case. Video 4 for training and Video 5 for testing

	Ram	Frontier	Silverado	Classification accuracy
	(a) YOLO classifier outputs			
Ram	11	1	50	0.1774
Frontier	14	47	8	0.6812
Silverado	43	2	4	0.0816
Average classification accuracy				0.3134
	(b) ResNet classifier outputs			
Ram	16	2	44	0.2581
Frontier	21	36	15	0.5
Silverado	8	1	44	0.8302
Average classification accuracy				0.5294

the average classification accuracies are 44% for Train 4 Test 5 (Table 8b) and Train 5 Test 4 (Table 9b) cases. We will need to use PCE full (30 to 1 compression), which can give 71% (Table 6b) and 54% (Table 10b) of classification accuracy for Train 4 Test 5 and Train 5 Test 4 cases, respectively.

4 Conclusions

In this paper, we present a high-performance approach to target tracking and classification directly in the compressive sensing domain. To the best of our knowledge, this is the first work in this area. Skipping the time-consuming reconstruction step will allow us to perform real-time target tracking and classification. The proposed approach is based on a combination of two deep learning schemes: YOLO for tracking and ResNet for classification. The proposed approach is suitable

Table 8 Classification results for PCE 25 case. Video 4 for training and Video 5 for testing

	Ram	Frontier	Silverado	Classification accuracy
	(a) YOLO classifier outputs			
Ram	7	9	35	0.1373
Frontier	20	35	7	0.5645
Silverado	20	4	3	0.1111
Average classification accuracy				0.2710
	(b) ResNet classifier outputs			
Ram	15	1	35	0.2941
Frontier	22	27	14	0.4286
Silverado	10	1	17	0.6071
Average classification accuracy				0.4433

Table 9 Classification results for PCE 25 case. Video 5 for training and Video 4 for testing

	Ram	Frontier	Silverado	Classification accuracy
	(a) YOLO classifier outputs			
Ram	11	25	42	0.141
Frontier	34	56	0	0.6222
Silverado	32	0	12	0.2727
Average classification accuracy				0.3453
	(b) ResNet classifier outputs			
Ram	17	28	34	0.2152
Frontier	15	67	12	0.7128
Silverado	19	7	18	0.4091
Average classification accuracy				0.4457

Table 10 Classification results for PCE full case. Video 5 for training and Video 4 for testing

	Ram	Frontier	Silverado	Classification accuracy
	(a) YOLO classifier outputs			
Ram	16	56	18	0.1778
Frontier	1	99	0	0.99
Silverado	43	0	14	0.2456
Average classification accuracy				0.4711
	(b) ResNet classifier outputs			
Ram	32	25	33	0.3556
Frontier	4	78	19	0.7723
Silverado	22	7	28	0.4912
Average classification accuracy				0.5397

Table 11 Classification results for PCE 50 case. Video 5 for training and Video 4 for testing

	Ram	Frontier	Silverado	Classification accuracy
	(a) YOLO classifier outputs			
Ram	20	29	41	0.2222
Frontier	21	76	0	0.7835
Silverado	41	0	13	0.2407
Average classification accuracy				0.4155
	(b) ResNet classifier outputs			
Ram	26	27	40	0.2796
Frontier	17	75	10	0.7353
Silverado	15	6	33	0.6111
Average classification accuracy				0.5420

for applications where limited training data are available. Experiments using SWIR videos clearly demonstrated the performance.

One potential direction is to integrate our proposed approach with real hardware to perform real-time target tracking and classification directly in the compressive sensing domain.

Acknowledgements This research was supported by the US Air Force under contract FA8651-17-C-0017. The views, opinions, and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the US Government. DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

References

1. Zhao, Z., Chen, H., Chen, G., Kwan, C., Li, X. R.: Comparison of several ballistic target tracking filters. In: Proceedings of American Control Conference, pp. 2197–2202 (2006)
2. Zhao, Z., Chen, H., Chen, G., Kwan, C., Li, X. R.: IMM-LMMSE filtering algorithm for ballistic target tracking with unknown ballistic coefficient. In: Proceedings of SPIE, Volume 6236, Signal and Data Processing of Small Targets (2006)
3. Zhou, J., Kwan, C.: Anomaly detection in low quality traffic monitoring videos using optical flow. In: Proceedings of SPIE 10649, Pattern Recognition and Tracking XXIX (2018)
4. Bertinetto, L., Valmadre, J., Golodetz, S., Miksik, O., Torr, P.: Staple: complementary learners for real-time tracking. In: Conference on Computer Vision and Pattern Recognition (2016)
5. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. **2**, 2246–2252 (1999)
6. Kwan, C., Zhou, J., Wang, Z., Li, B.: Efficient anomaly detection algorithms for summarizing low quality videos. In: Proceedings of SPIE 10649, Pattern Recognition and Tracking XXIX, vol. 1064906 (2018)
7. Kwan, C., Yin, J., Zhou, J.: The development of a video browsing and video summary review tool. In: Proceedings of SPIE 10649, Pattern Recognition and Tracking XXIX, vol. 1064907 (2018)
8. Kandylakis, Z., et al.: Multimodal data fusion for effective surveillance of critical infrastructure. In: ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, pp. 87–93 (2017)
9. Berg, A.: Detection and Tracking in Thermal Infrared Imagery. Linköping University Electronic Press, Diss (2016)
10. Zhou, J., Kwan, C.: Tracking of multiple pixel targets using multiple cameras. In: 15th International Symposium on Neural Networks (2018)
11. Kwan, C., Chou, B., Kwan, L. M.: A comparative study of conventional and deep learning target tracking algorithms for low quality videos. In: 15th International Symposium on Neural Networks (2018)
12. Candes, E.J., Wakin, M.B.: An introduction to compressive sampling. IEEE Signal Proc. Mag. **25**, 21–30 (2008)
13. Zhang, J., Xiong, T., Tran, T., Chin, S., Etienne-Cummings, R.: Compact all-CMOS spatio-temporal compressive sensing video camera with pixel-wise coded exposure. Opt. Express **24**(8), 9013–9024 (2016)
14. Yang, J., Zhang, Y.: Alternating direction algorithms for l_1-problems in compressive sensing. SIAM J. Sci. Comput. **33**, 250–278 (2011)
15. Tropp, J.A.: Greed is good: algorithmic results for sparse approximation. IEEE Trans. Inf. Theory **50**(10), 2231–2242 (2004)
16. Dao, M., Kwan, C., Koperski, K., Marchisio, G.: A joint sparsity approach to tunnel activity monitoring using high resolution satellite images. In: IEEE Ubiquitous Computing, Electronics & Mobile Communication Conference, pp. 322–328, New York City (2017)
17. Zhou, J., Ayhan, B., Kwan, C., Tran, T.: ATR performance improvement using images with corrupted or missing pixels. In: Proceedings of SPIE 10649, Pattern Recognition and Tracking XXIX, vol. 106490E (2018)
18. Applied Research LLC, Phase 1 Final report, August 2016
19. Yang, M.H., Zhang, K., Zhang, L.: Real-time compressive tracking. In: European Conference on Computer Vision (2012)
20. Kwan, C., Chou, B., Echavarren, A., Budavari, B., Li, J., Tran, T.: Compressive vehicle tracking using deep learning. In: IEEE Ubiquitous Computing, Electronics & Mobile Communication Conference, New York City (2018)
21. Redmon, J., Farhadi, A.: YOLOv3: An Incremental Improvement (2018). arXiv:1804.02767
22. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems (2015)
23. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Conference on Computer Vision and Pattern Recognition (2016)
24. Definition of mAP, https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173. Accessed 10 Jan 2019