Method Article

Video Surveillance Processing Algorithms utilizing Artificial Intelligent (AI) for Unmanned Autonomous Vehicles (UAVs)

Minh T. Nguyen\textsuperscript{a,∗}, Linh H. Truong\textsuperscript{b}, Trang T.H. Le\textsuperscript{a}

\textsuperscript{a} Thai Nguyen University of Technology, Viet Nam
\textsuperscript{b} National Tsing Hua University, Taiwan

\textbf{A B S T R A C T}

With the advancement of science and technology, the combination of the unmanned aerial vehicle (UAV) and camera surveillance systems (CSS) is currently a promising solution for practical applications related to security and surveillance operations. However, one of the biggest risks and challenges for the UAV-CSS is analysis, process, and transmission data, especially, the limitations of computational capacity, storage and overloading the transmission bandwidth. Regard to conventional methods, almost the data collected from UAVs is processed and transmitted that cost huge energy. A certain amount of data is redundant and not necessary to be processed or transmitted. This paper proposes an efficient algorithm to optimize the transmission and reception of data in UAV-CSS systems, based on the platforms of artificial intelligence (AI) for data processing. The algorithm creates an initial background frame and update to the complete background which is sent to server. It splits the region of interest (moving objects) in the scene and then sends only the changes. This supports the CSS to reduce significantly either data storage or data transmission. In addition, the complexity of the systems could be significantly reduced. The main contributions of the algorithm can be listed as follows;

- The developed solution can reduce data transmission significantly.
- The solution can empower smart manufacturing via camera surveillance.
- Simulation results have validated practical viability of this approach.

The experimental method results show that reducing up to 80% of storage capacity and transmission data.

This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/)

\textbf{A R T I C L E  I N F O}

\textit{Method name:} D-CNN object classification

\textit{Keywords:} Artificial Intelligence, Background Modeling, Region of Interest (RoI), Convolution Neural Networks (CNN), Video Surveillance (VS), Unmanned aerial vehicles (UAVs)

\textit{Article history:} Received 7 November 2020; Accepted 26 July 2021; Available online 27 July 2021

\textsuperscript{∗} Corresponding author.

E-mail addresses: nguyentuanminh@tnut.edu.vn, tuanminh.nguyen@okstate.edu (M.T. Nguyen).

DOI of original article: 10.1016/j.cie.2020.106671

E-mail addresses: nguyentuanminh@tnut.edu.vn, tuanminh.nguyen@okstate.edu (M.T. Nguyen).

https://doi.org/10.1016/j.mex.2021.101472

2215-0161/© 2021 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/)
Specification table

| Subject Area | Computer Science, Data processing, Data communication |
|--------------|--------------------------------------------------------|
| More specific subject area | Data Surveillance, Object classification and recognition |
| Method name | D-CNN object classification |
| Name and reference of original method | Minh T. Nguyen, Linh H. Truong, Trang T. Tran, Chen-Fu Chien, “Artificial intelligence based data processing algorithm for video surveillance to empower industry 3.5”, Computers & Industrial Engineering, Volume 148, 2020, 106671. |
| Resource availability | https://doi.org/10.1016/j.cie.2020.106671 |

Method details

Introduction

Over the decades, unmanned aerial vehicles (UAVs) are considered to be an alternative solution, aiming to create the safest working environments to humans from dangerous areas or high-risk missions [1,2]. With the ability to remotely real-time surveillance, UAVs equipped with cameras can capture images or videos to track targets such as people, vehicles or specific areas.

Currently, UAV surveillance has been upgraded with many more features of self-controlling, analysis and data processing by integrating UAVs with artificial intelligence (AI) [3]. Using these technologies, UAVs can be trained to perform particular tasks by processing large amounts of images and videos and simultaneously identifying presence region of interests (RoIs) in frames such as rebar detection bridge deck inspection [4], perform monitoring tasks for early warning of natural disasters [5] or crack detection [6]. Indeed, the integration between UAV with AI technology could help UAV in performing complex tasks rather than surveillance. In addition, AI technology also improve existing limitations of UAV surveillance systems such as storage capacity, processing capability, transmission bandwidth, thus help transmitting data continuously, reducing computational cost and increase the accuracy of the RoIs detection [7,8].

Following to the realistic needs, this study introduces a novel method aiming to distribute more appropriate in the data processing process on both UAVs and Server sides of the UAV surveillance system. The block diagram in Fig. 1 describes the operation steps of the video processing on the UAV surveillance side before sending data to the server-side. Each frame of the input video is used for two purposes, which is performing backgrounds, Bf and region of interest, RoIs. The Bf is referred to the background which is updated if there are static changes over previous frames. In the other hand, RoIs contain moving objects which are obtained by applying background subtraction technique. A special feature is that only the RoIs will be sent to the server. Meanwhile, the background through the updating process from Bi into Bf, the final background. The Bf is then sent to the server-side. Merging Bf and ROI (moving objects) on the server side would generate frames with full background.

![Block diagram](image)
and foreground. Also, this technique applied on the camera side helps the server on the server side only need to focus ROIs to process further tasks. Thereby reducing the computational burden on the server side.

**Data Processing at camera surveillance side**

Each frame of original input video with the size of width, \( W \) and height, \( H \) has static blocks and might be contained moving blocks. In order to model the background without moving objects, an algorithm is shown in Fig. 2 is proposed. The background is modeled following the \( t \)-consecutive video frames, \( t \) can be varied based on specific scenes. The frames at time \( t=1 \) and time \( t+1 \) are denoted by \( F^t \) and \( F^{t+1} \), respectively. At the first frame of the input video \( F^1 \) or \( t = 1 \), the initial background frame \( B^1 \) is defined as undefined and denotes as “black” blocks. For following frames, the frame \( t \) is compared with the previous frame \( t - 1 \) by a block matching algorithm to determine moving areas that can be referred as area contained moving objects and create motion maps, \( R^t \). Block matching algorithm calculates the cost function at each possible location in the search window. This leads to the possible match of the macro-block in the reference frame with a block in another frame. The background, \( B^t \) are updated based on the previous initial background \( B^{t-1} \) and motion frames \( R^t \). The background updating procedure is performed until all undefined blocks are filled by background blocks.

As shown in Fig. 3, there are original frames \( F^t \), motion frame \( R^t \) in which moving blocks are denoted as “black” blocks and “white” blocks as static blocks, and background initials at different frames. At the first frame, \( t = 1 \), the background initial, \( B^1 \) is modeled with all “black” blocks, as shown in Fig. 3e. From \( t = 2 \), by applying block matching algorithm, if a frame includes moving objects, the motion map, \( R^t \) blocks will be marked with “black” blocks, otherwise will be a “white” block. The background modeling process is performed via \( R^t \) and the previous background, \( B^{t-1} \). In addition, at this frame, if block in \( R^t \) that is “white” block and the corresponding block in the \( B^{t-1} \) initial is “black” then those blocks in the background initial will be duplicated from the frame. Similarly, after all “black” blocks in background initial are replace by “white” blocks in \( R^t \), the background initial is performed. Since then, the background initial will participate in the process of updating the background.

Each block in \( F^t \) could be classified into four different categories as: “background”, “still object”, “environmental change”, and “moving object”. Motion map for each frame, \( R^t \) exits two states which are background marked in “white” block and movement marked in “black” block. Correlation coefficient \( C_B \), as shown in Eq. 1, is compared with threshold to determine the category of each block in two consecutive frames \( F^t, F^{t-1} \). The block representation of video frame \( F^t \) is classified as “background” if the block represents of \( R^t \) is “background”, so if \( C_B > TH_{CB} \) or “still object” if \( C_B < TH_{CB} \). Otherwise, for \( R^t \), if \( C_B > TH_{CB} \) the frame block represent is “environmental change”, else is “moving object”.

![Fig. 2. The flowchart of the background initialization](image-url)
Correlation coefficient:

\[ C_{B(m,n)} = \frac{\sum |b_{m,n}^t - \mu b_{m,n}^t| \times |\hat{b}_{m,n}^t - \mu \hat{b}_{m,n}^t|}{\sqrt{\sum |b_{m,n}^t - \mu b_{m,n}^t|^2} \times \sqrt{\sum |\hat{b}_{m,n}^t - \mu \hat{b}_{m,n}^t|^2}} \]  \hspace{1cm} (1)

Background and environment change updating

\[ \hat{b}_{m,n}^t = \alpha \hat{b}_{m,n}^t + (1 - \alpha) b_{m,n}^t \]  \hspace{1cm} (2)

Still object updating:

\[ \tilde{b}_{m,n}^t = \begin{cases} b_{m,n}^t & \text{Count}_{(m,n)} \geq \text{TH}_{\text{still}} \\ \hat{b}_{m,n}^t & \text{otherwise} \end{cases} \]  \hspace{1cm} (3)

Moving object updating:

\[ \tilde{b}_{m,n}^t = \begin{cases} b_{m,n}^t & \text{SM}(b_{m,n}^t) < \text{SM}(\hat{b}_{m,n}^t) \\ \hat{b}_{m,n}^t & \text{otherwise} \end{cases} \]  \hspace{1cm} (4)
Where \( b_{m,n}^t = \{ F_{(mN+i,nN+j);i,j=0,1,2,...,N-1}^t \} \) denotes the block index of frame \( t \), \( F^t \), \( b_{m,n}^{t-1} = \{ F_{(mN+i,nN+j);i,j=0,1,2,...,N-1}^{t-1} \} \) denote the block index of frame \( t-1 \), \( F^{t-1} \) and \( \hat{b}_{m,n}^{t} = \{ \hat{F}_{(mN+i,nN+j);i,j=0,1,2,...,N-1}^{t} \} \) is denote the block index of the background \( \hat{B}^{t} \), \( \hat{b}_{m,n}^{t} \) denote the block index of the initial background frames, \( \mu_b \) is the mean of the pixel values in block \( \hat{b}^{t} \). Additionally, \( SM(\hat{b}_{m,n}^t) \) and \( SM(\hat{b}_{m,n}^{t-1}) \) denote the side-match measures for block \( \hat{b}_{m,n}^t \) from \( F^t \) embedded in \( \hat{F}^t \) and that for block \( \hat{b}_{m,n}^{t-1} \) embedded in \( \hat{F}^{t-1} \), respectively. Note that if a block in \( \hat{F}^t \) is determined as a “moving object” block two or more times continuously.

The next step is extracting moving objects in each frame and transmit to the server side. The metrics is applied for the model performance evaluation of ROIs extraction and D-CNN process: F-Measure, is defined as:

\[
F - \text{measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]

Where \( \text{precision} = \frac{TP}{TP+FP} \) and \( \text{recall} = \frac{TP}{TP+FN} \). The range of F-measure is between 0 and 1 where if the result of F-measure = 1, that mean the predicted value is totally match the ground truth.

The videos used in this paper is taken from the ATON dataset which contains several videos captured from camera surveillance. By using this method, the time to create the background initial is approximate 30 frames, then 170 next frames for background updating that the blocks are “still object” or “illumination change” is updated. Finally, the updated background will be sent to the server side. Otherwise, it will use the previous background. TH_{still} is a rate to update the background and it can vary for different scenes. The experimental results show that if setting \( TH_{still} = 20 \), the background will be updated faster (at frame = 97 after has initialized “background initial”) and vice versa with \( TH_{still} = 40 \) (at frame 173), however, the accuracy of \( TH_{still} = 40 \) is higher as shown in Fig. 4.

After having background model, the process which is shown in Fig. 5, is separate the foreground from the original frame. Only these foregrounds are sent to the server side. Our network is trained to be able to map raw pixel values from video frames to a set of binary values, between 0 and 1.

Based on the image resolution, it will see \( W \times H \times 3 \) (\( W = \text{Width}, H = \text{Height}, 3 = \text{RGB color channels} \)). The CNNs models will receive each input frame to train and test. Table 1 depicts the processed steps of network architecture including two stages with many convolutional layers. At the first stage, each frame separated from the video is directly received by a modified CNN network. The 16 feature maps having the kernel size and the stride of 1 at the end of the first convolutional block are transformed from this frame. Next step, the output of the second block is the 32 feature maps of size \( W/2 \times H/2 \). These feature maps are made from the 16 feature maps at the end of the first convolutional block down sampling a max-pooling layer with a stride of 2. The following steps are repeated the same which present in Table 1.

Transposed convolution (TC) uses some learnable parameters to unsampled the input feature map into a desired output feature map. In this case, the input feature map of the TC is the output of the Edge Modified CNN. It is taken to the Transposed CNN (TCNN) to learn the weights for decoding the feature maps. Following the input of TCNN have the size of 16 times smaller than the original size and consists of 256 feature maps with the size of \( \frac{W}{16} \times \frac{H}{16} \times 256 \). With respect to the block 6 of TCNN stage, the output of modified CNN is projected into \( \frac{W}{8} \times \frac{H}{8} \times 128 \) by using \( 2 \times 2 \) transposed convolution filter with stride of 2.

By applying sigmoid function and threshold the pixel values at the last layer, the output of the transpose CNN is the probability mask for each pixel with the value for noticed objects of 1 and background of 0.

The region of interest extraction with findContour is shown in Fig. 6.

About 50 to 200 training examples are employed in this proposed model. From the SBI dataset as training examples, the experiments select random 100 images. There are around 1.2 million parameters in total. The true label compares with the predicted value by using a binary cross-entropy loss function. The RMSProp optimizer with a batch size of 5, 100 epochs are applied to aim with training the network. Based on the description in Fig 7, the model checkpoint helps to improve the network performance and increase the precision of the model. This point does not update until
training accuracy increases, or training losses decrease. Hence, this strategy helps the network keep training results better.

**Data Processing at Server Side**

The server side receives the RoIs to train their own D-CNN network, as shown in Fig. 8. When the network bases on from those actual data, accuracy and relevance will significantly increase. The first default input size of RoIs is 64 × 64 pixels of RGB to meet the input image requirements from the
The Edge DCNN network configuration at the camera side. A modified CNNs (based on U-net network architecture [9]) is from block 1 to 4, where block 0 is RGB input image. In the other side, the edge Transpose-CNN is from block 5 to 9, where block 9 is the output probability mask from the network.

| Table 1 |
|-----------------------------------------------|
| Edge modified-CNN | Edge Transposed CNN |
| 0 | $W \times H \times 3$, RGB image |
| 1 | (Conv) $W \times H \times 16$, $F = 3 \times 3, S = 1$ |
| | Max-pooling, $F = 2 \times 2, S = 2$ |
| | Dropout rate = 0.1 |
| 2 | (Conv) $W / 2 \times H / 2 \times 32$, $F = 3 \times 3, S = 1$ |
| | Max-pooling, $F = 2 \times 2, S = 2$ |
| | Dropout rate = 0.1 |
| 3 | (Conv) $W / 4 \times H / 4 \times 64$, $F = 3 \times 3, S = 1$ |
| | Max-pooling, $F = 2 \times 2, S = 2$ |
| | Dropout rate = 0.1 |
| 4 | (Conv) $W / 8 \times H / 8 \times 128$, $F = 3 \times 3, S = 1$ |
| | Max-pooling, $F = 2 \times 2, S = 2$ |
| | Dropout rate = 0.1 |
| 5 | (Conv) $W / 16 \times H / 16 \times 256$, $F = 3 \times 3, S = 1$ |
| | Max-pooling, $F = 2 \times 2, S = 2$ |
| | Dropout rate = 0.1 |

| Table 2 |
|-----------------------------------------------|
| The D-CNN network configuration at the server side. |
| 1 | Input layer: 64×64 pixels |
| 2 | Convolution layer: $W \times H \times 256$, $F = 3 \times 3, S = 1$ |
| | Active function: ReLU |
| | Max-pooling layer: $W / 2 \times H / 2 \times 256$, $F = 2 \times 2, S = 1$ |
| 3 | Convolution layer: $W / 4 \times H / 4 \times 256$, $F = 2 \times 2, S = 2$ |
| | Active function: ReLU |
| 4 | Flatten layer |
| 5 | Fully connected layer: Dense 64 |
| 6 | Fully connected layer: Dense 1 |
| 7 | Active function: sigmoid |

various ROI size separated from the previous step. In addition, about 200 RoIs is deployed to train DCNN at the server side. This model system needs to go through several experiments to achieve the best performance with the optimal size. When the server receives well-defined RoIs, the system only needs to focus RoIs with the small size and no need for complex classification [11]. All RoIs with the default size selected manually classifies into two classes as “car” and “truck”. The classification system consists of two convolution layers, two pooling layers arranged alternately, two fully connected layer and finally the output layer. The rectifier linear units (ReLU) in the final layer plays a role as the active function. The detailed custom classification of the network is given in Table 2.
Fig. 6. ROIs extraction with findContour at random frames.

The number of ROIs is divided following two classes 100 ROIs chosen as cars and 100 ROIs as trucks, all extracted from the video in the ATON dataset- 33 seconds of length with 25 frames per second, thus total frames are 825 frames (http://cvrr.ucsd.edu/aton/shad).

To achieve the highest accuracy of the training process, the network need has experimented for several epochs, with an era range from 5 to 20. The best result got an epoch equaling 12 in which the trade-off between training Creating time and accuracy are the same. Fig. 9 shows that at epoch is 12, our DCNN took 13.18 seconds to finish with a loss: 0.0544 - acc: 0.9841 - lost val: 0.1650 - val acc: 0.8571.

In order to calculate the complexity, the number of parameters of each layer has been calculated, that is convolution layer as follow:

\[ \text{Number of parameters} = F^2 \times C \times K \]  

(6)
Fig. 7. Evaluate training performance at edge side.

Fig. 8. The D-CNN object classification.

Fig. 9. Evaluate training performance of the DCNN at the server side.
Table 3
Complexity and memory usage of approaching and a state-of-the-art.

|                          | D-CNN model | YOLO model |
|--------------------------|-------------|------------|
| Input size               | 64 × 64     | 224 × 224  |
| # of conv layer          | 2           | 24         |
| # of parameters          | 209K        | 66,000K    |
| Memory usage/image       | 1690KB      | 53,606KB   |

Algorithm 1
Background updating algorithm.

- Background initialization process
- Block matching algorithms to determine different blocks or moving objects

for each block in frames do
    - Calculate the correlation coefficient $C_B(m, n)$ as Eq. 1 to classify categories of each blocks in frames.

    if blocks is classified as “moving objects” or labeled in “black” blocks then
        if $C_B(m, n) >$ threshold then
            update as “environmental change” type as Eq. 2
        else
            update as “moving object” type as Eq. 4
        end
    else
        if $C_B(m, n) >$ threshold then
            update as “backgrounds”,” type as Eq. 2
        else
            update as “still objects” type as Eq. 3
        end
    end
end

Where $F$ is the filter size, $C$ is the number of input channels, and $K$ is the number of filters. As shown in Table 2, the number of parameters of the custom CNN on the server side can significantly reduce in the network compared to existing object classification and detection methods. At the server edges, the parameters in CNN customized has 209K compared to 66 million or approximately 0.3% of the YOLO method's parameters. In addition, the memory usage per image/frame is 1.7 MB compared to 53.1 MB of the YOLO method.

The classification of the noticed objects bases on evaluating the performance of the custom CNN. In addition, F-measure is also implemented as the measurement. The precision, recall, and F-measure are detailly calculated, as shown in Table 3. With respect to each input image size, the experiment was repeated 10 times using random sub-sampling. Because the size and volume of the data set is smaller,
the performance of their system is not high. In the future, when the amount of data is enhanced, this performance will certainly improve. The measure here is the average of 10 experiments with accuracy, precision and recall. The proposed network shows very good processing results with a smaller number of operations in both training and classification.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to thank Thai Nguyen University of Technology, Viet Nam for the support. Thanks to Professor Chen-Fu Chien - National Tsing Hua University for the valuable comments to improve the paper.

References

[1] A. Masaracchia, D.B. Da Costa, T.Q. Duong, M. Nguyen, M.T. Nguyen, H.T. Nguyen, T.V. Quyen, C.V. Nguyen, A.M. Le, H.T. Tran, M.T. Nguyen (Eds.), A pso-based approach for user-pairing schemes in noma systems: theory and applications, IEEE Access 7 (2019) 90550–90564. and “Control algorithms for UAVs: A comprehensive survey,” EAI Endorsed Transactions on Industrial Networks and Intelligent Systems, vol. 7, 5 2020.

[2] C. Van Nguyen, T. Van Quyen, A.M. Le, L.H. Truong, M.T. Nguyen, Advanced hybrid energy harvesting systems for unmanned aerial vehicles (UAVs),” advances in science, Technol. Eng. Syst. J. 5 (2020) 34–39 March.

[3] A. Singh, D. Patil, S. Omkar, Eye in the sky: real-time drone surveillance system (DSS) for violent individuals identification using scatternet hybrid deep learning network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 1629–1637.

[4] H. Ahmed, H.M. La, K. Tran, Rebar detection and localization for bridge deck inspection and evaluation using deep residual networks, Autom. Constr. 120 (2020) 103393.

[5] B. Mishra, D. Garg, P. Narang, V. Mishra, Drone-surveillance for search and rescue in natural disaster, Comp. Commun. (2020).

[6] U.H. Billah, H.M. La, A. Tavakkoli, Deep learning-based feature silencing for accurate concrete crack detection, Sensors 20 (16) (2020) 4403.

[7] V. Badrinarayanan, A. Handa, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. arXiv preprint arXiv:1505.07293 (2015).

[8] M.T. Nguyen, L.H. Truong, T.T. Tran, C.F. Chien, Artificial intelligence based data processing algorithm for video surveillance to empower industry 3.5”, Comp. Ind. Eng. 148 (2020) 106671 Volume.

[9] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.