LinkStream: A Liquidity Modeling System on Large-Scale Video Stream in Oilfield

Hao Yuan, Qiang Ma*, Zhe Hu, Xiaoxiang Li, Xu Wang, Zheng Yang
School of Software, Tsinghua University, Beijing, China
{yuanh19, huz19, lixx21}@mails.tsinghua.edu.cn, {wangxu2020, yangzheng}@tsinghua.edu.cn
thumaq@mail.tsinghua.edu.cn, *Corresponding Author

2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS) | 978-1-6654-0878-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICPADS53394.2021.00103

***Abstract*—This article introduces LinkStream, a liquidity modeling system based on multiple video streams designed and implemented for oilfield. LinkStream combines a variety of technologies to solve several problems in computing power and network latency. First, the system adopts an edge-central architecture and tailoring based on spatio-temporal correlation, which greatly reduces computing power requirements and network costs, and enables real-time analysis of large-scale video stream on limited edge devices. Second, it designed a set of liquidity models to describe the liquidity status in the oilfield. Finally, it uses object tracking technology to design a counting algorithm for the unique tubing object in the oilfield. We have deployed LinkStream in an oilfield in Iraq. LinkStream can perform real-time inference on over 200 video streams with acceptable resource overhead.**

***Index Terms*—Liquidity Modeling, Edge Computing, Object Counting, Cross-Camera Tracking**

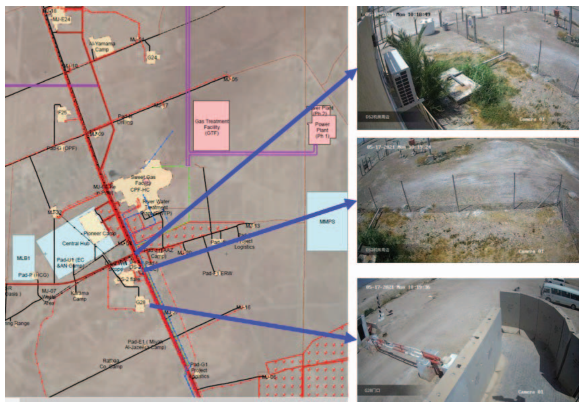
Fig. 1. Deployment of Three Representative Cameras in Majnoon Oilfield

I. Introduction

The rapid development of the Internet of Things (IoTs) and Deep Learning has made it possible to build intelligent analysis systems based on large-scale video stream. A series of applications designed for different scenarios were born, such as industrial security, home automation, and smart city. This work, which aimed at the oilfield industrial zone, has designed and implemented a system based on large-scale cameras for object tracking and liquidity monitoring.

This system was deployed in Majnoon oilfield, which is one of the largest oilfields in the world. It is located in the southeast of Basra, Iraq, with an oilfield area of 626 square kilometers. In the oilfield, data transmission is required to be a high priority for oil and gasoline industry especially production and environmental data. To cover core facilities and well pads, we plan to deploy over 300 wireless Mesh nodes for the comprehensive backbone communication system in Majnoon oilfield. Based on the Backbone Communication Network, smart video analysis applications shall be actively monitored to assist personnel in protecting oilfield public property by detecting and preventing crime. These applications also need to detect intruders and automatically notify guards or broadcast messages through loudspeakers. Edge detection applications need to be scalable and help reduce bandwidth and storage usage by sending and recording only related videos.

Figure 1 shows the schematic diagram of the camera deployment position and several camera video images. Hundreds of surveillance cameras are deployed at the entrances, crossroads, industrial regions, and warehouses of the industrial zone. In addition, there are security personnel carrying video capture equipment for patrol monitoring. Different from other industrial areas, the data collected by the monitoring equipment deployed in the oilfield includes not only general objects such as personnel and vehicles, but also special objects such as oil tubing and oil tanks. The use of video information provided by multiple devices to model and analyze the liquidity information of these objects in the oilfield is of great significance to the establishment of a higher-level intelligent industrial management system.

While the large-scale video stream provides rich information, it also brings huge challenges to the system design. Specifically, in order to meet the needs of real scenes, we need to pay attention to the following problems:

- Large-scale video stream (up to hundreds of channels) puts a huge demand on the computing power of the system. It is unrealistic to use cloud servers for model inference while the network in oilfiled is poor with unacceptable overhead and delay. At the same time, cross-camera analysis will amplify the computational load caused by the scale of the data stream, because it needs to cross the camera and time to discover the correlation between different objects.
- In an oilfield with a large area, both fixed-position cameras and mobile cameras are needed. How to construct a spatio-temporal correlation model to describe liquidity

information based on the data analysis results of several cameras is another issue worthy of attention.
- Object counting task is an important part of analyzing liquidity data. For general objects such as person and vehicles, traditional cross-camera object tracking has achieved good results [1], [2], [3]. However, how to count the oil tubing that are densely stacked is still a problem to be solved.

In this work, we propose LinkStream, a liquidity modeling system based on multiple video streams. This system integrates multiple technologies to solve these challenges.

Edge-Central Architecture Design. We use an edge-central computing architecture that offloads some tasks to edge nodes to save computing power on edge servers. The design consists of two parts: the frame filtering of the fixed camera and the independent processing of the mobile camera.

Spatio-Temporal Correlation. By constructing a spatio-temporal correlation model, we predict the probability of the object moving between two specific cameras with a certain time interval to tailor the search space of the ReID task. It will reduce the cost of cross-camera analysis.

Counting Based on Object Tracking. Aiming at the tubing object, we propose a counting algorithm based on ”Detection + Tracking”. The algorithm uses the improved YOLOv3 model and DeepSORT algorithm to obtain preliminary results and corrects the results based on geometric relations.

Liquidity Model. We divide the oilfield into several sub-areas that do not overlap with each other according to geographic location, and associate the camera with the sub-areas. Based on single-camera counting and cross-cameras tracking, we can obtain the object liquidity relationsh between the regions.

We deploy the system in a oilfield and use real data stream to inspect this system. On this basis, we separately inspected the different units of the system, and evaluated the system design in terms of resource overhead, system delay, and accuracy performance. Evaluation result shows that LinkStream can save 30% to 60% in various resource indicators compared to the unoptimized version. Secondly, LinkStream can perform real-time inference on over 200 video streams with a delay of no more than 2s. Besides, the tubing counting algorithm can count tubing at a speed of 15fps, while the error rate does not exceed 4%. Finally, the liquidity information described by LinkStream can be used in other parts of the system, such as real-time monitoring and alarms.

We will show a brief overview of the design framework of the whole system in Section II-A, and describe the technical details of the system components mentioned in overview in detail in the remainder of Section II, including the division of labor design between camera and server (Section II-B), the object counting algorithms (Section II-D), and the liquidity model (Section II-E). Then, we will show our detailed deployment configuration and some experimental results on the system in Section III. Finally, we will introduce some existing related work in Section IV .

II. System Design

A. Overview

Figure 2 shows the architecture design of LinkStream, which composed of a two-tier structure of Edge Device - Central Server. The edge device refers to the multi-channel cameras used to capture video stream, and the central server is responsible for scheduling hardware and executing algorithms. Because different components play different roles in the system, we will run frame filtering and lite object tracking algorithms on the camera, and run object tracking algorithms and cross-camera analysis algorithms on the central server. Cross-camera analysis of large-scale video stream will bring huge computational resource overhead, especially when running deep learning algorithms that take up a lot of GPU and memory. Therefore, we use the spatio-temporal correlation of the object liquidity between cameras to optimize system performance. To obtain liquidity information, it is also necessary to convert the results of object tracking into quantitative information. So we have designed a set of algorithms for counting different objects in oilfield to count the numbers. Finally, the system combines the above information with the actual map to get a series of descriptions about the oilfield's object liquidity information, which can be used by managers to monitor the status of the oilfield area and also can be used by other advanced applications through API.

B. Edge Camera Process

For the edge-central architecture, using the computing power of the edge nodes is important to solve the problem of poor cellular network in oilfield area and limited computing power of the central server. The edge node is responsible for tasks such as video capture, lightweight analysis, and dynamic offloading. The task of executing heavyweight analysis algorithms and integrating the analysis results will be deployed on the central server.

The cameras in this system includes two parts: fixed-position cameras and mobile cameras. They have various differences in computing power, network, and system roles. Fixed cameras are generally placed in important areas of the oilfield (such as entrances and exits, crossroads, etc.). Mobile cameras are mainly used to somewhere hard to cover by fixed cameras, including roads and open areas. Because of the hardware limitations of the fixed camera (generally normal webcam), the algorithms it can run are limited, but can transmit data to the central server via network cable. The mobile camera can use a device with computing power such as a smart phone, and it is only able to transmit data to the central server through cellular or mesh network.

Based on the above analysis, we only run the intelligent frame filtering task on the fixed cameras, and all filtered video frames are handed over to the central server for subsequent tasks, such as single-camera object tracking and cross-camera analysis. Besides, we deploy a small single-camera object tracking algorithm on the mobile camera, and only send the final processing result to the central server.

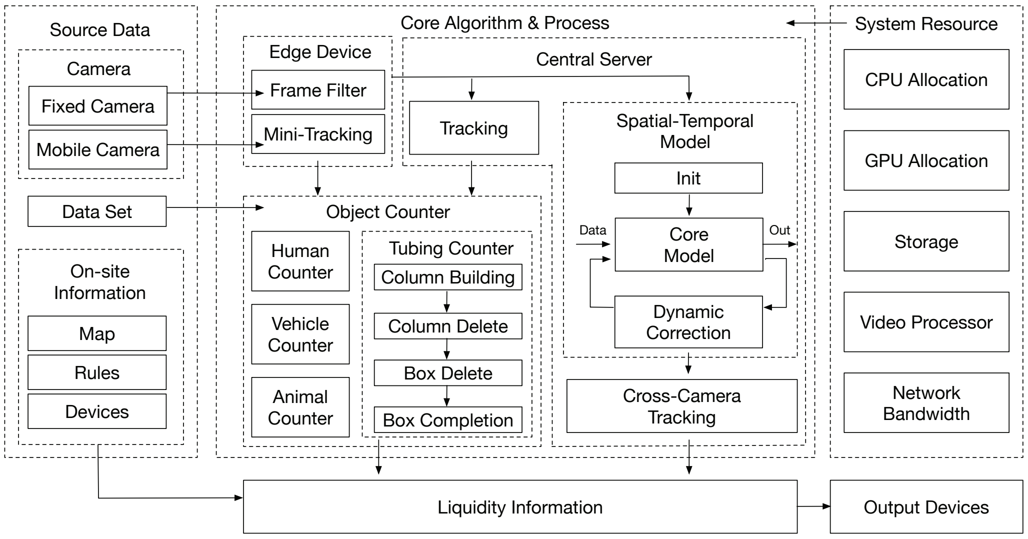

Fig. 2. System Architecture

The frame filtering algorithm is mainly based on difference detection [4]. We will hold a buffered frame and calculate the point of difference with newly frame of the video stream. The newly frame with the point greater than the threshold is marked ”new” and be used to update buffered frame, otherwise it is marked ”same”. The mark information will be sent to the central server along with the video. The judgment of the point of difference mainly relies on the motion vector and the comparison of inter-frame macroblocks.

C. Cross-Camera Analysis

The main goal of the cross-camera analysis task is to obtain the movement of the same object between different fixed cameras which has been executed frame filtering.

The distribution of fixed cameras is discrete: compared with the physical area of the oilfield, the coverage area of fixed cameras only occupies a small part, and it takes longer time to move between cameras. So we will solve this problem through cross-camera object re-identification (ReID) based on the single-camera object tracking results. It should be noted that it is expensive to re-identify large-scale objects in large-scale video stream one by one, so we need design an algorithm to tailor the search space. Some empirical analysis [5] shows that the movement of a object between cameras in a specific area shows spatial and temporal correlation. We design a search space pruning algorithm based on this spatio-temporal correlation to improve system performance. We define $n(c_i, c_j)$ as the historical frequency of appearance in camera c_j after leaving camera c_i.

The information about the appearance, movement and departure of objects can be obtained by single-camera object tracking algorithm. Due to the sparsity of camera distribution, when a object leaves a certain camera, it may appear in the camera network next time in tens of minutes. It is unreasonable to find the destination of the object in all of the subsequent frames. Therefore, we use reverse matching to track the trajectory. The system saves the characteristic information of the object leaving one monitoring area in the database. When a new tracking object is detected by a camera, we do ReID task on this object with all of unknown location object in the database. If the ReID match is successful, system will report an inter-camera liquidity event and remove related data from the database. Otherwise, system will report that a new object has appeared.

Doing ReID task in historical objects one by one has a large time overhead, so we use spatio-temporal information to prune the search space. Compared with ”whereabouts”, reverse matching in our system pay more attention to ”source”. Therefore, we define $S(c_i, c_j)$ as:

$$S(c_i, c_j) = \frac{n(c_i, c_j)}{\sum_s n(c_s, c_j)} \tag{1}$$

$S(c_i, c_j)$ represents the probability that the object appearing in the c_j camera comes from the camera c_i. We will give priority to the ReID matching of the object in the camera with a larger robability. For each source camera c_i, we will only consider objects whose interval time is $\lambda_{ij} \pm 2\sigma_{ij}$, where λ_{ij} and σ_{ij} are the average and standard deviation of the interval time of all objects from c_i to c_j.

If there is still no matching result after checking the cameras with 95% of the cumulative probability density, we will execute a complete historical object matching. If this match is still not successful, mark the object as a new object and do subsequent operations.

How to get the spatio-temporal correlation information used in this algorithm is another important issue. A one-time solution is to perform offline analysis on the dataset of the real scene to get frequency information. This is an expensive one-time operation. The modeled spatio-temporal correlation data can be directly used in system deployment.

Fig. 3. Different Types of Oil Tubing Placed in Open Areas.

In the actual system implementation, we also added a series of small tricks to improve system performance. First of all, considering that the oilfield has fences and entrances, we have marked some cameras as ”entrance” and the object leaving from the specific direction of the monitoring screen of ”entrance” cameras is regarded as leaving the oilfield area These objects will not be added to the database that needs to be tracked. Secondly, we believe that when a object leaves the monitoring area for an hour, it has already resided in certain areas of the oilfield. Further tracking in these object does not have the meaning. Therefore, cross-camera tracking is only for the object within an hour. In addition, we believe that the spatio-temporal correlation model is class-special. This idea is natural due to the difference in the speed of people walking and vehicles traveling. Therefore, we will establish their own spatio-temporal correlation models for different tracking classes.

D. Crowd Counting

In order to improve the robustness of the algorithm, we adopt object tracking framework in single-camera object counting task. For objects with obvious feature, such as people and vehicles, direct use of object tracking algorithms has achieved good performance. However, there are few researches on special object like tubing, which have similar shapes and be small. We adopt the idea of ”Detection + Tracking” based on multi-object tracking to counting to solve the problems of incomplete detection and repeated detection that counting by one pictures. Figure 3 shows that a large number of oil tubing of various shapes are placed on the open area.

In the object detection stage, we add 2 feature maps of larger size from the original 3 size feature maps of YOLOv3, and correspondingly add 6 types of a prior bounding boxes with smaller scales. This change reduces the receptive field of the object detection model, making the network more sensitive to small objects. Subsequently, we retrained the modified model on our oil tubing data set.

Because the oil tubing are densely packed and irregular, the object detection results often cause box overlap and loss. Therefore, we use geometric laws to correct the object detection. We assume that the tubing is arranged in Columns (this assumption is true for the stored tubing), so the center point of the detection frame must also be distributed in the vicinity of several straight lines. Based on this assumption, the algorithm can be given as follows:

Algorithm 1: Initialization of the Tubing Column

```
input  : Detection boxes sequence $box\_list$
output: List of ”Column” $result\_list[]$
1  Sort box_list by abscissa;
2  $result\_list[] \leftarrow$ initResultList ();
3  foreach element item of box_list do
4      $flag \leftarrow 0$;
5      foreach element list of $result\_list[]$ do
6          $a \leftarrow list.end()$;
7          if $a.x == item.x$ then
8              CONTINUE;
9          $k \leftarrow \frac{item.y-a.y}{item.x-a.x}$;
10         if $|k| < \lambda_1$ and $|item.y - a.y| < item.y \times \lambda_2$ then
11             $list.append(item)$;
12             $flag \leftarrow 1$;
13             BREAK;
14     if $flag == 0$ then
15         $nlist \leftarrow$ newList();
16         $nlist.append(item)$;
17         $result\_list.append(nlist)$;
```

1) **Column Building**: We arrange the detection boxes according to the abscissa, and use several lists to represent several columns arranged in parallel, and then assign the detection boxes to the middle area of the corresponding column. See the pseudo code of Algorithm 1 for specific allocation and construction rules.
2) **Column Delete**: For each column, calculate the degree of overlap of the abscissa with other columns by the coordinates of the first element and the last element. If the degree of overlap with any other column is less than the threshold (We set the threshold to 0.2), this column will be deleted.
3) **Box Delete**: Traverse the elements in each column in order. If the overlapping area of a detection box with the front and back boxes is greater than 70%, it will be regarded as a duplicate box and will be deleted.
4) **Box Completion**: We calculate the average width of each line of detection boxes and traverse. For a check box in a column if its s_i of any box is greater than 0.5 times the average Width and the horizontal distance between this box and the previous box is larger than 0.7 times of average width, a detection frame with an average width is added in the middle of the two. The calculation method of s_i is:

$$s_i = x_i - x_1 - \sum_{k=1}^{i-1} w_k \tag{2}$$

After correcting the detection frame, the detection result will be entered into DeepSORT for object tracking, and the unique track-id will be used as the counting object. Taking into account the repeated occurrence of objects caused by camera shake, only the tracking objects appearing in the central area of the frame are considered in the counting phase.

E. Liquidity Model

In Section II-B - II-D, we showed several parts that make up the system. These algorithms finally get some information by analyzing the data of multiple video streams, and we will design our liquidity model based on these information.

First, we divide the oilfield area into multiple sub-areas according to the physical logic, such as the oilfield gate, warehouse entrance, warehouse interior, mine inspection points, etc. These sub-areas do not overlap with each other and constitute a coverage of the effective area of the oilfield.

Then, we associate the camera with these sub-areas. The physical installation location of a fixed camera determines which sub-area it will be associated. Our model allows multiple cameras to be associated with the same area. For mobile cameras, due to its flexibility, managers can freely designate patrol routes, which means that it will be associated with multiple sub-areas, depending on the specific location corresponding to the current video frame. We make the mobile camera cover the sub-area not covered by the fixed camera as much as possible.

The single object tracking algorithm on the mobile camera reports the number of objects detected by the camera at the current position to the central camera. Although this detection task can also be performed by the object detection framework, the counting based on object tracking can reduce the influence of the outlier detection results of individual frames caused by short-term occlusion and other factors, and improve the robustness of counting. The detection result is combined with the area corresponding to the current coordinates of the mobile camera to give information about the number of objects in the area.

Although the fixed camera video is processed by the central server, similar information about the number of objects in the area can also be obtained. In addition, the cross-camera analysis of the central camera on the multi-channel fixed camera video stream can obtain the liquidity relationship of the object between the regions. Considering that there are multiple cameras associated with the same sub-area, in order to convert the liquidity information between cameras into inter-area liquidity information, it is also necessary to filter the object liquidity in the same sub-area and merge the liquidity between sub-areas with the same starting point or end point. Besides, since we have marked the ”entrance” camera, we can also get information about the entry and exit of person and vehicles in the oilfield area.

In general, our description of the liquidity within the oilfield mainly includes: a) Information about the number of objects counted within the scope of a fixed or moving camera at a certain time; b) Information about the movement of the object across the camera; c) People, vehicles, etc. entering and leaving the oilfield region information; d) The estimated total number of each object in the oilfield by summarizing above data. These information can be used by managers to monitor the status of the oilfield in real time, and can be used by other advanced analysis programs.

III. Evaluation

In this section, we will evaluate the performance of each component of the system in terms of resource occupation and algorithm effect.

A. Configuration

We deployed the LinkStream system in the oilfield area. Approximately two hundred cameras are deployed in various areas of the oilfield, covering almost all key areas. We use Hikvision cameras as fixed cameras, and connect a small codec device to perform frame filtering operations. We invited 10 patrol officers to hold mobile phones as mobile cameras. The patrol personnel are trained to take the initiative to take pictures when passing through the oil tubing object in the open area to facilitate counting. In addition, we use a linux server cluster equipped with NVIDIA Tesla T4 and GTX 1660Ti GPUs as the edge server for processing video stream data.

In terms of software implementation, we use the DeepStream framework as the main framework for the central server to process video streams, and make several modifications based on the official version to support frame filtering, automated deployment, message control, time stamp synchronization and other functions. We implemented the frame filtering algorithm based on some open source code [6]. For object tracking tasks, we chose the YOLOv3 tiny framework in mobile and use YOLOv3 framework optimized for small objects in central server as the object detection framework, and retrained in our real samples. The object tracking part of both uses the DeepSORT algorithm. In addition, we use lightweight ReID algorithm [7] to reduce the resource consumption of the system. Next, we will evaluate the performance of LinkStream on different performance indicators and make some comparisons, and finally show a schematic diagram of the system.

B. Bandwidth Consumption

We only measured the impact of mini-tracking algorithms and frame filtering algorithms running on the camera on network bandwidth because the traffic interaction only includes the video stream and additional data.

For a 1080p, 30fps video stream shot by a mobile phone, due to the movement of the camera, the video stream is not same images frame by frame, and more than 100Mb of original video data will be generated every minute. The cellular network in the real oilfiled is only 1Mbps, so video data cannot be transmitted in time. A small tracking algorithm can save this part of the bandwidth, and only transmit object statistical data that does not exceed 1Kbps. For fixed cameras, the frame filtering algorithm only appends an additional byte of data per frame, which can be ignored for fixed optical fibers.

TABLE I
RESULTS OF SPEED

Object number	Average FPS	Proportion
0	24.62	19.0%
1-10	22.76	13.8%
11-20	20.87	16.7%
21-30	18.73	22.3%
31-40	15.32	15.2%
41-50	11.46	8.7%
>50	6.89	4.2%

After the optimization of this design, the optical fiber data received by the central server every minute is almost unchanged, still at 1Gb (200 channels, fixed picture saves bit rate), and the cellular network data has been reduced from more than 1G (10 patrol personnel) to less than 1M.

C. System Delay

The delays of different parts of the system have different effects on the overall delay of the system. We will discuss the different components separately.

1) Mini-Tracking and Crowd Counting Delay: The retrained YOLOv3 tiny + DeepSORT model can perform object tracking at an average speed of 15 FPS on the mobile phone. We noticed that the speed would change greatly with the number of objects in the image. If there is no object in the field of view, the detection speed can exceed 20 FPS; when the number of objects is large (such as a large number of oil tubing in the field of view), the detection speed will be as low as 10 FPS or even lower.

We divided the interval according to the different number of objects in the image to test the corresponding detection speed. We use each frame in the video as an image for the experiment, and give the proportion of each interval in the video. The results are shown in Table I. Figure 4 shows the change curve of the algorithm processing speed when passing a group of object dense areas.

Statistical analysis of the patrol video shows that most of the areas where the patrol personnel pass are few objects. Therefore, for the 30-frame original video stream, we adopt the strategy of sampling every other frame and only input the 15fps video stream into the processing pipeline. Experiments show that this will not cause video congestion in a time window, and the maximum processing delay of processing results in densely-objected areas is about 53s.

2) Cross-Camera Analysis Delay: Since the fixed camera has a fixed field of view, the number of detected objects does not change much, and the object detection and tracking algorithm running on it runs at a relatively stable speed. The experiment results show that under the acceleration of DeepStream and TensorRT, the YOLOv3 + DeepSORT model can be used to process 24 channels of 30 FPS 1080P video stream data in real time on a single Tesla T4 GPU. The server computing power of our cluster is no less than 10 T4 GPUs, which can support the real-time processing of more than 200 video streams in the entire oilfield area.

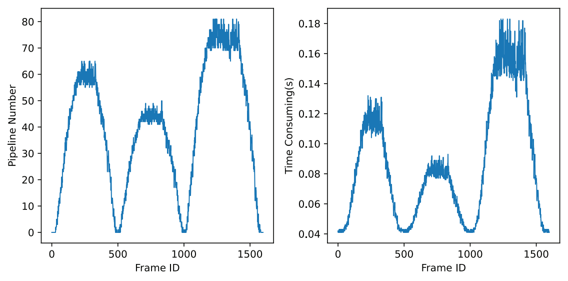

Fig. 4. It can be seen that the processing time of the video frame is positively correlated with the number of objects in the current frame.

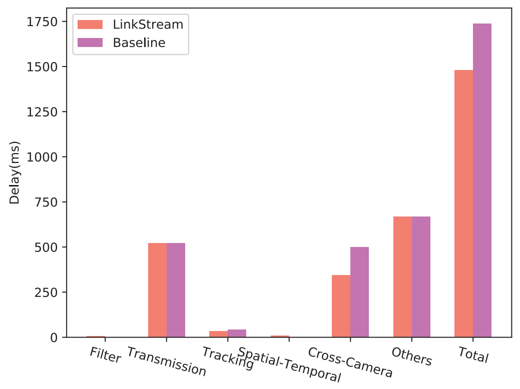

Fig. 5. Overall Delay

The hit rate of spatio-temporal correlation model is one of the important components that affects the delay of cross-camera analysis. We first manually annotate the camera data for a week, which is used for system initialization. Subsequently, we tracked the system log to check the time-space-related hit rate. The results show that the hit rate of the system is finally stable at about 37%, which can about 82% of time on average in each hit on average compared to naked search without spatial tailoring.

3) Overall delay and Comparison: In addition to the main components above, the overall delay of the system also includes the processing delay of the frame filtering algorithm, the transmission delay of the network (wired link and cellular network), video stream encoding and decoding delay, stream redirection delay (system internal resource scheduling), database cost of query. Figure 5 shows the delay of the different components of the system and the overall analysis delay from the generation of the video stream to the analysis result. In addition, Figure 5 also compares the system analysis latency without our optimization strategy.

As can be seen, due to the high transmission delay and internal system delay, and the cluster has sufficient computing power, our design has limited optimization of the overall system delay. But it can better save system resource consumption and ensure operational robustness.

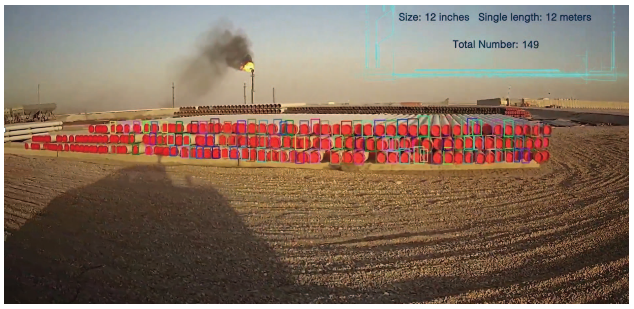

Fig. 6. Real-Time Detection and Counting Result

D. Accuracy

1) Tubing Counting Accuracy: We choose several independent case in the real environment to show the result of counting results, and compared the program processing results with the real value, which is the accurate result after multiple manual counting. The results are shown in Table II.

We found that the error of the 8 experiments was less than 4%, and the average error was 2.72%. We noticed that the overall counting results were higher than the true value, which should be due to the ID switch phenomenon caused by the change of characteristics in the process of moving in the object tracking. As a result of the phenomenon, some tubing was counted for many times, making the result higher.

The real-time detection and counting effect is shown in Figure 6. You can view the demo of the effect of the tubing counting algorithm through [8].

TABLE II
RESULTS OF ACCURACY

Case ID	**Real value**	**Processing results**	**Error**
1	144	149	3.47%
2	190	188	1.06%
3	318	306	3.77%
4	184	190	3.26%
5	207	199	3.86%
6	156	159	1.92%
7	411	423	2.92%
8	276	272	1.50%

2) Mini-Tracking Accuracy: In order to ensure the accuracy of the mobile camera counting, we compared the counting effects of the small tracking algorithm and the large tracking algorithm. The results are shown in Table III.

It can be seen that because the robustness of object detection is slightly inferior to that of large tracking algorithms, small tracking algorithms running on mobile devices will produce significant results, but this loss of accuracy is acceptable.

E. Resource Consumption

We compared the cost of system resources under different camera numbers, where $xF + yM$ means to connect x fixed camera and y mobile camera. Figure 7 shows the bandwidth overhead, which has been explained in Section III-B. Figure 8 shows the relative occupancy rate of the GPU decode unit.

TABLE III
RESULTS OF MINI-TRACKING ACCURACY

Case ID	Real value		YOLOv3		YOLOv3-Tiny	
	Vehicle	Human	Vehicle	Human	Vehicle	Human
1	0	20	0	20	0	19
2	0	45	0	43	0	46
3	7	13	7	13	6	13

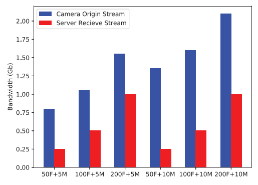

Fig. 7. Bandwidth

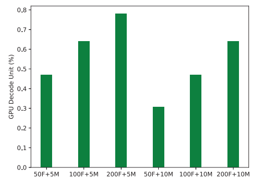

Fig. 8. GPU Decode Consumption

The comparison object is a system that has not adopted an optimization strategy. The save of decode unit is mainly because the calculation on the mobile camera reduces the number of video streams that the central server needs to process. In addition, under the condition of 200F+10M, the cost of GPU computing resources is reduced to 31% of the original. The save of computing unit is mainly due to the frame filtering algorithm and search space tailoring based on spatio-temporal correlation, which greatly reduces the amount of calculation performed by the central server.

IV. RELATED WORKS

Video Analysis System. In terms of hardware, some camera manufacturers, such as Hikvision, Omnicast, ProVigil, etc., provide closed solutions based on smart cameras, but this can only be used in a limited environment, and it is difficult to handle more complex and customized requirements. In terms of software system, some video analysis systems mainly consider optimizing query tasks on a single video stream [9], [10], [11], or configuring and customizing the content of different cameras to balance cost and efficiency [12] to optimize overall system performance.

Edge Computing. Input filtering is a simple and important solution to accelerate DNN model inference [4]. At the same time, the lightweight research of deep learning [13], [14], [15] makes it possible to run neural networks on edge devices. Therefore, there are a series of studies that mainly consider executing part of models or lightweight models on edge devices with limited performance [16], [17], [18] to save network overhead and central server resources.

Spatio-Temporal Correlation. A series of work has begun to consider using spatio-temporal information to optimize cross-camera analysis [19], [5]. But there are two problems in these studies: 1) The acquisition of the spatio-temporal information largely relies on manual marking, but complete manual marking and modeling is often costly or impossibly. 2) The currently used spatio-temporal information considers fewer factors, and more information can be explored and utilized.

Object Tracking. Cross-camera tracking tasks are mainly divided into three sub-tasks: single-camera object detection and tracking, cross-camera ReID, and cross-camera tracking. The current mainstream single-camera object tracking is mainly divided into two categories: correlation filtering [20], [21] and deep learning [1], [22], [23]. For the task of ReID, good progress has been made in personnel [24], [25] and vehicles [26], [27], but there is little research on other objects. Cross-camera object tracking is based on the results of the first two tasks, and uses data association [28], gaussian mixture model [19] or spatio-temporal information [5] to assist in the establishment of a complete motion trajectory.

V. Conclusion

The oilfield industrial area is a special realistic environment. In this environment, the object tracking, counting, and liquidity monitoring of video streams have their own unique significance and challenges. We propose LinkStream, a system for real-time analysis on large-scale video stream to obtain liquidity. In order to achieve this goal, we use frame filtering and edge computing to save bandwidth and GPU occupancy, and use the spatio-temporal correlation model to reduce system delay. In addition, we also proposed an algorithm for real-time accurate counting of tubing objects. The results show that the system can perform real-time inference at a low error rate and greatly reduce system resource overhead. In the future, we will explore more complete spatio-temporal correlation models to better realize cross-camera analysis.

VI. Acknowledgment

This work is supported in part by the National Key R&D Program of China under grant 2018AAA0101200, the NSFC under grant 62072272, Shuimu Tsinghua Scholar Program.

References

[1] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*, pp. 3645–3649, IEEE, 2017.

[2] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," in *European conference on computer vision*, pp. 749–765, Springer, 2016.

[3] M. Xu, M. Zhu, Y. Liu, F. X. Lin, and X. Liu, "Deepcache: Principled cache for mobile deep vision," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pp. 129–144, 2018.

[4] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *The 25th Annual International Conference on Mobile Computing and Networking*, pp. 1–16, 2019.

[5] S. Jain, X. Zhang, Y. Zhou, G. Ananthanarayanan, J. Jiang, Y. Shu, P. Bahl, and J. Gonzalez, "Spatula: Efficient cross-camera video analytics on large camera networks," in *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, pp. 110–124, IEEE, 2020.

[6] V. Kantorov and I. Laptev, "Efficient feature extraction, encoding and classification for action recognition," in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE, 2014*, 2014.

[7] L. He, X. Liao, W. Liu, X. Liu, P. Cheng, and T. Mei, "Fastreid: A pytorch toolbox for general instance re-identification," *arXiv preprint arXiv:2006.02631*, 2020.

[8] "cube counting demo." https://cloud.tsinghua.edu.cn/f/923507a70e1249368e67/, 2021.

[9] Y. Lu, A. Chowdhery, and S. Kandula, "Optasia: A relational platform for efficient large-scale video analytics," in *Proceedings of the Seventh ACM Symposium on Cloud Computing*, pp. 57–70, 2016.

[10] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, pp. 377–392, 2017.

[11] K. Hsieh, G. Ananthanarayanan, P. Bodik, S. Venkataraman, P. Bahl, M. Philipose, P. B. Gibbons, and O. Mutlu, "Focus: Querying large video datasets with low latency and low cost," in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pp. 269–286, 2018.

[12] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: scalable adaptation of video analytics," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pp. 253–266, 2018.

[13] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[14] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[15] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

[16] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy, "Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 123–136, 2016.

[17] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "Deepdecision: A mobile deep learning framework for edge video analytics," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pp. 1421–1429, IEEE, 2018.

[18] P. Jain, J. Manweiler, and R. Roy Choudhury, "Low bandwidth offload for mobile ar," in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, pp. 237–251, 2016.

[19] Y.-G. Lee, Z. Tang, and J.-N. Hwang, "Online-learning-based human tracking across non-overlapping cameras," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 2870–2883, 2017.

[20] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 3, pp. 583–596, 2014.

[21] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg, "Eco: Efficient convolution operators for tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6638–6646, 2017.

[22] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, "Fast online object tracking and segmentation: A unifying approach," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1328–1338, 2019.

[23] G. Wang, C. Luo, X. Sun, Z. Xiong, and W. Zeng, "Tracking by instance detection: A meta-learning approach," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6288–6297, 2020.

[24] M. Geng, Y. Wang, T. Xiang, and Y. Tian, "Deep transfer learning for person re-identification," *arXiv preprint arXiv:1611.05244*, 2016.

[25] Z. Zhong, L. Zheng, Z. Zheng, S. Li, and Y. Yang, "Camera style adaptation for person re-identification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5157–5166, 2018.

[26] W. Lin, Y. Li, X. Yang, P. Peng, and J. Xing, "Multi-view learning for vehicle re-identification," in *2019 IEEE international conference on multimedia and expo (ICME)*, pp. 832–837, IEEE, 2019.

[27] S. Lee, E. Park, H. Yi, and S. H. Lee, "Strdan: Synthetic-to-real domain adaptation network for vehicle re-identification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 608–609, 2020.

[28] Y. He, X. Wei, X. Hong, W. Shi, and Y. Gong, "Multi-target multi-camera tracking by tracklet-to-target assignment," *IEEE Transactions on Image Processing*, vol. 29, pp. 5191–5205, 2020.