A Liquidity Analysis System for Large-scale Video Streams in the Oilfield

QIANG MA, Tsinghua University, Beijing, China
HAO YUAN, Tsinghua University, Beijing, China
ZHE HU, Tsinghua University, Beijing, China
XU WANG, Tsinghua University, Beijing, China
ZHENG YANG, Tsinghua University, Beijing, China

This article introduces LinkStream, a liquidity analysis 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 information 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.

CCS Concepts: • **Computer systems organization** → **Sensor networks**; • **Networks** → *Network performance analysis*

Additional Key Words and Phrases: Liquidity analysis, edge computing, object counting, cross-camera tracking

ACM Reference Format:
Qiang Ma, Hao Yuan, Zhe Hu, Xu Wang, and Zheng Yang. 2024. A Liquidity Analysis System for Large-scale Video Streams in the Oilfield. *ACM Trans. Sensor Netw.* 20, 3, Article 65 (April 2024), 22 pages. https://doi.org/10.1145/3649222

1 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 work is supported in part by the National Key Research Plan under grant No. 2021YFB2900100, the NSFC under grant No. 62072272, No. 62202263, No. 62372265, No. 62202262, No. 62302254, and No. 62272462.
Authors' address: Q. Ma (Corresponding author), H. Yuan, Z. Hu, X. Wang, and Z. Yang, Software School, Tsinghua University, Haidian District, Beijing, 100084, China; e-mails: thumaq@mail.tsinghua.edu.cn, yuanh19@mails.tsinghua.edu.cn, huz19@mails.tsinghua.edu.cn, wangxu2020@tsinghua.edu.cn, yangzheng@tsinghua.edu.cn.

ACM 1550-4859/2024/04-ART65
https://doi.org/10.1145/3649222

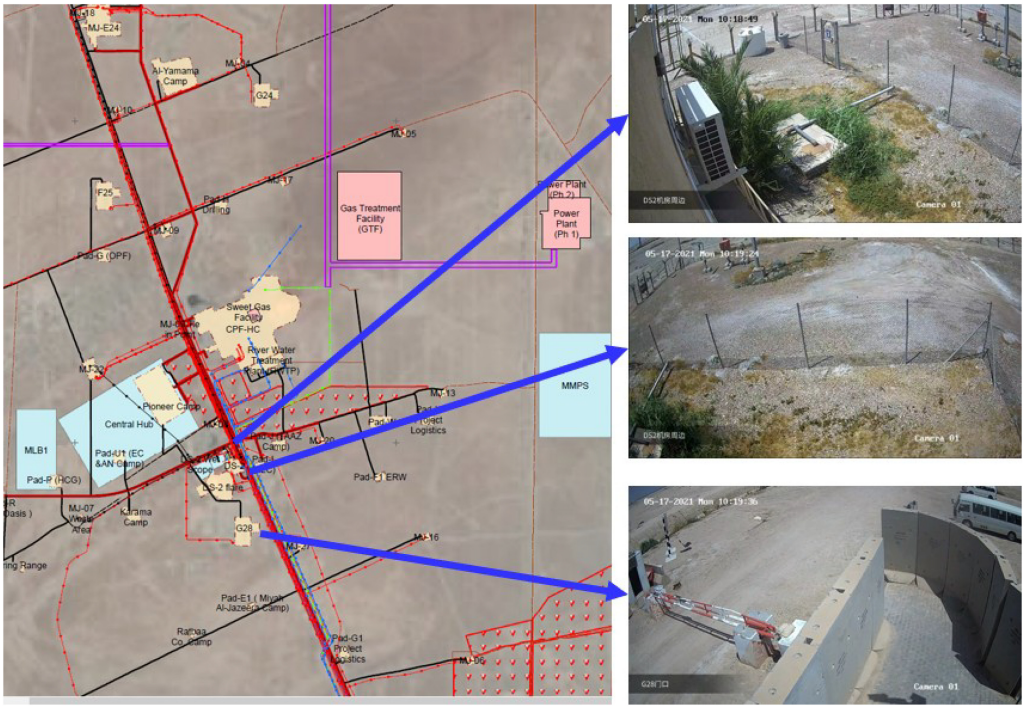


Fig. 1. Part of camera deployment in majnoon oilfield.

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. The use of video information provided by multiple devices to 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 many 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. Considering the volume of video data, network communication overhead and delay, it is unrealistic to use cloud servers for model inference, while the oilfield network environment is poor. At the same time, because of trade secrets, some sensitive information is not allowed to be uploaded to the cloud. So it is difficult to use external servers to assist analysis, and it is necessary to make a reasonable optimization strategy for the system to reduce the computing load of the central server.
— For the cross-camera identity matching task, the computational complexity is quadratic with the size of the video stream and historical time due to the need to discover the correlation between objects across cameras and time (see Section 2.3 for details). When dealing with large-scale video streams, this computational overhead becomes unacceptable. How to speed up this task is another problem that needs to be solved.
— Object counting task is an important part of analyzing liquidity information. For general objects such as person and vehicles, traditional cross-camera object tracking has achieved good results [11, 31, 32]. However, how to count the oil tubing that are densely stacked, have high feature similarity, but with a special internal placement structure is still a problem to be solved.

In this work, we propose LinkStream, a fluidity analysis system based on multiple video streams. This system integrates multiple technologies to solve the above-mentioned 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 edge caching and frame filtering edge caching algorithm of the fixed camera and 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 oil tubing object, we propose a counting algorithm based on Detection and Tracking. The algorithm uses the YOLOv5 model and DeepSORT algorithm to obtain preliminary results and corrects the results based on geometric relations.

Liquidity Information. 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 relationship between the regions.

We deploy the system in a oilfield and use a 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 2 s. Besides, the tubing counting algorithm can count oil tubing at a speed of 15 fps, 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 2.1, and describe the technical details of the system components mentioned in overview in detail in the remainder of Section 2, including the division of labor design between camera and server (Section 2.2), the cross camera analysis algorithms using spatio-temporal correlation model

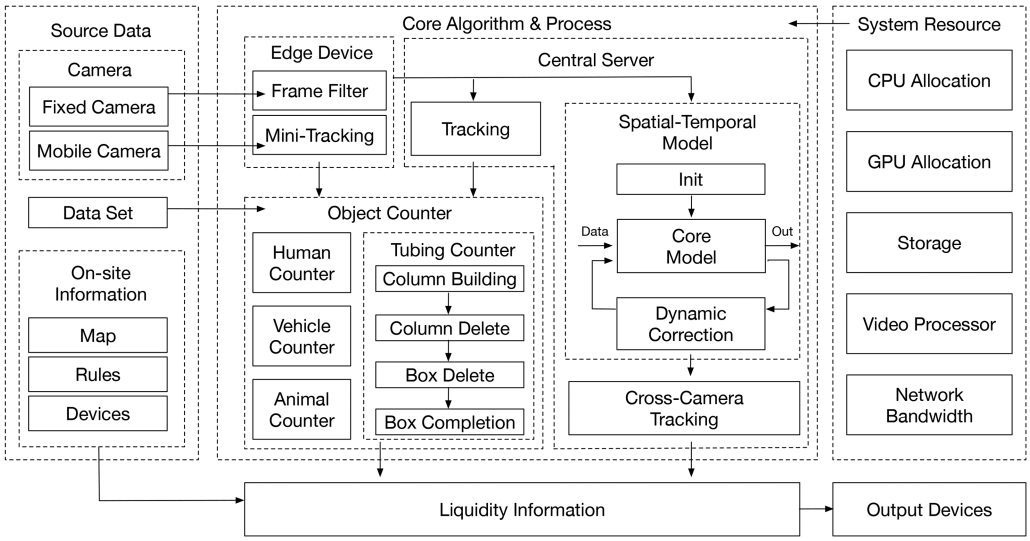


Fig. 2. System architecture.

(Section 2.3), the object counting algorithms (Section 2.4), and the liquidity model (Section 2.5). Then, we will show our detailed deployment configuration and some experimental results on the system in Section 3. Finally, we will introduce some existing related work in Section 4.

2 SYSTEM DESIGN

2.1 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 detection 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.

2.2 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 heavy-weight analysis algorithms and integrating the analysis results will be deployed on the central server.

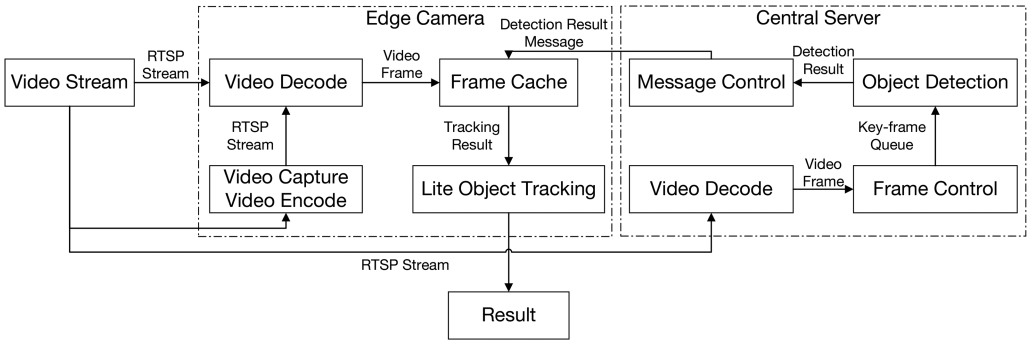


Fig. 3. Edge-central architecture.

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 deploy a small single-camera object tracking algorithm on the mobile camera, and only send the final processing result to the central server. Besides, we adopts a real-time video analysis method based on edge caching and KCF object tracking algorithm on fixed cameras and central server to strike a balance between computational load and computational accuracy. Its basic structure is shown in Figure 3.

The processing module on the central server is mainly divided into two parts, one is responsible for filtering the key-frames that need to be processed, and another is responsible for running the object detection algorithm on the selected key-frames. The edge nodes located near the fixed camera receive the original video stream information on the one hand, and the key-frames object detection result message from the central server on the other hand. Edge nodes cache key-frames analysis results and run lightweight object tracking algorithms (we use multi-threaded KCF algorithm) on all frames. The pseudo-code of the multi-threaded KCF algorithm is shown in Algorithm 1.

For key-frames selection algorithms running on a central server, in general, an increase in the object detection frame rate will improve the accuracy of the analysis, but it will also increase the computational load and processing time. A naive way is to directly use a fixed frame skipping frequency. Note that for different video streams or even different time periods of the same video stream, the impact of the object detection frame rate on the analysis effect is different. Therefore, we propose a dynamic scheduling method for the detection frame rate to automatically adjust the object detection frame rate of each video stream in the system.

This is an optimization problem, the following constraints need to be satisfied:

(1) There is an upper limit on the number of video frames processed by the object detection task on the server per second, which is determined by the computing power of the GPU.
(2) The total delay of the analysis tasks performed by each video stream cannot exceed the total length of the time segment. Otherwise, the video stream will be congested.
(3) Analysis accuracy are affected both by the video stream detection frame rate and the object motion speed. We make this system perform regular evaluation to obtain the relationship

ALGORITHM 1: Multi-Threaded KCF Algorithm

```
input  : frame, interval, detection_bboxes, trackers
output : detection_bboxes, trackers
def track_frame (frame, interval, detection_bboxes, trackers):
    if frame.index % interval = 0 then
        trackers ← [];
        foreach bbox ∈ detection_bboxes do
            tracker ← cv2.TrackerKCF_create();
            tracker.init(frame, bbox);
            trackers.append(tracker);
    else
        detection_bboxes ← [];
        threads ← [];
        foreach tracker ∈ trackers do
            thread ← Thread (execute_track, (tracker, frame));
            threads.append(thread);
            thread.start();
        foreach thread ∈ threads do
            thread.join();
            bbox ← thread.get_result();
            if bbox! = NULL then
                detection_bboxes.append(bbox);
    return detection_bboxes, trackers;
def execute_track(tracker, frame):
    ok, bbox ← tracker.update(frame);
    if ok then
        return bbox;
    else
        return NULL;
```

function between the accuracy and detection frame rate of each video stream at different object motion speeds.

Let $S = s_1, s_2, \ldots, s_n$ be the set of video streams. F is the number of frames of the original video stream, and C is the computing resources. Consider each time slice with a length of Γ, f_i^t is the frame rate of the tth time slice to the video stream s_i. τ_{det} and τ_{track} are represent the processing delay of object detection and object tracking tasks. $\phi_i^t = \Psi(s_i, v_i^{t-1})$ is the function of the relationship between the accuracy with video frame rate and the speed of the object. Then the problem can be formally defined as follow:

$$\max_{f_i^t} \frac{1}{n} \sum_{i=1}^{n} \phi_i^t(f_i^t)$$

$$\begin{aligned} s.t. \quad & f_i^t \geq f_{min}, && i = 1, 2, \ldots, n \\ & f_i^t \leq \tfrac{1 - F \times \tau_{track}}{\tau_{det} - \tau_{track}} && i = 1, 2, \ldots, n \\ & \textstyle\sum_{i=1}^{n} f_i^t \leq C. \end{aligned}$$

The first constraint gives a lower bound on the number of detection frame rate to ensure that the tracking results will not be too biased. This problem can be solved by the interior point method.

In addition, we adopt an intelligent frame filtering task on the fixed cameras to avoid executing the object detection algorithm in a still frame to improve system. The frame filtering algorithm is mainly based on difference detection [23]. 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 macro blocks.

2.3 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 single camera tracking.

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.

Let's consider a simplified model. Assuming that object appear and disappear evenly in a single camera, in the past T time interval, the expected number of objects leaving a camera is $O(T)$, and the expected number of objects appearing in a camera is $I(T)$. Due to the uniformity, we can simply assume that $O(T)$ and $I(T)$ are both linear functions, that is, $I(T) = k_i T$, $O(T) = k_o T$. In a scene with n video streams, for an object appearing at a camera at time t, the number of possible matches to be considered is $nO(t) = nk_o t$, because all possible objects in the past need to be matched. Then for all objects in the past time period T the total number of possible matches is approximately:

$$n \sum_{t_1}^{t_{I(T)}} nk_o t = n^2 k_o \sum_{t_1}^{t_{I(T)}} t = n^2 k_o \times \frac{k_i T^2}{2}.$$

So we need design an algorithm to tailor the search space. Some empirical analysis [17] shows that the movement of an 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 an 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 robustness. 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.

The pseudocode of the object matching module based on spatio-temporal correlation model is as algorithm 2:

ALGORITHM 2: Cross-Camera Analysis Algorithm

```
input : Object Tracking List object_list
input : Last Object Tracking List last_object_list
appear_object_list, leave_object_list ← compare (object_list, last_object_list);
foreach object ∈ leave_object_list do
    saveDataByTrackID (object.id, object);
foreach object ∈ appear_object_list do
    step ← 0;
    while step < 2 do
        if step == 0 then
            cur_object_list ← getFilterData (object);
        else
            cur_object_list ← getData ();
        foreach object_c ∈ cur_object_list do
            result ← ReID (object, object_c);
            if result == True then
                updateDataByTrackID (object_c.id, object);
                ReportMove (object);
            else
                ReportNewObject (object);
        step = step + 1;
```

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. Furthermore, during the running of the system, the matching results are also recorded, and the recorded results are regularly added to the model to dynamically optimize the spatio-temporal correlation model.

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

Fig. 4. Different types of oil tubing placed in open areas.

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 an 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.

2.4 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 oil 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 4 shows that a large number of oil tubing of various shapes are placed on the open area.

Different from the general multi-target tracking task, the task scene is the camera in motion rather than the object in motion, but considering that the tubing is stationary, in essence, or a static and moving situation, and the camera fixed multi-target tracking task situation is not intrinsically different, so it can still use the multi-target tracking processing method for processing. There are still two problems that need to be dealt with. One is the inevitable jitter of the camera in the shooting process, which makes some objects move out of the image for several frames and then move in again, so that they have different ids in the tracking, resulting in repeated counts. Another problem is that some of the tubing will be missed because of factors such as occlusion and lighting, and the dense target is easy to produce detection box overlap, both of which will cause the detection results to be inaccurate and then shadow tracking and counting.

In the object detection stage, we use yolov5l with a larger model scale as the underlying framework for object detection to obtain a higher receptive field. 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 oil tubing is arranged in Columns (this assumption is true for the stored oil 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:

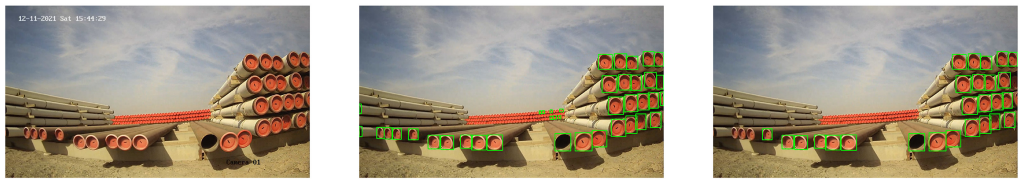
ALGORITHM 3: Initialization of the Oil Tubing Column

input : Detection boxes sequence *box_list*
output : List of "Column" *result_list*[]

```
Sort box_list by abscissa;
result_list[] ← initResultList ();
foreach item ∈ box_list do
    flag ← 0;
    foreach list ∈ result_list[] do
        a ← list.end();
        if a.x == item.x then
            CONTINUE;
        k ← (item.y−a.y)/(item.x−a.x);
        if |k| < λ1 and |item.y − a.y| < item.y × λ2 then
            list.append(item);
            flag ← 1;
            BREAK;
    if flag == 0 then
        nlist ← newList();
        nlist.append(item);
        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 3 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

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



(a) Original (b) Without Filter (c) With Filter

Fig. 5. Remote tubing stack.

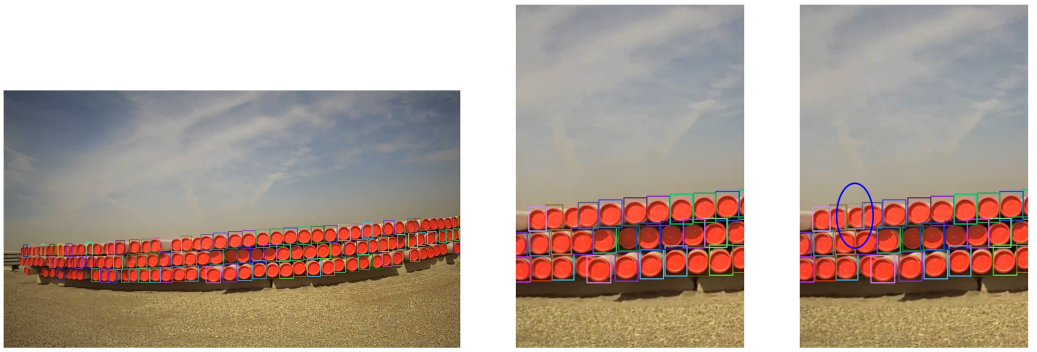
In addition to geometric correction algorithm, we adds a series of box filter tricks to improve counting accuracy. First of all, in order to ensure the validity of the above correction algorithm, we only counts the oil tubing detected at the position of the center with a left-right ratio of 0.35 to 0.65 and an upper-lower ratio of 0.1 to 0.9. Second, as shown in Figure 5, sometimes distant tubing that are not belong to the current stack will be detected. And this will cause the current tubing stack to count too much. So we correct the result according to the currently detected size of the tubing stack. When there is a detection box with an area smaller than one third of the maximum area detection result, the algorithm will filter it out. The pseudocode is shown in Algorithm 4.

ALGORITHM 4: Filter of object detection results

input : object detection box *bbox_list*
output: filtered object detection box *filtered_bbox_list*

```
filtered_box_list ← initBboxList ();
frame_w, frame_h ← imgInfo ();
now_size ← None;
foreach bbox ∈ box_list do
    if bbox.x_br > frame_w × 0.65 and bbox.x_tl < frame_w × 0.35 and bbox.y_br > frame_h × 0.9
      and bbox.y_tl < frame_w × 0.1 then
        CONTINUE;
    bbox_w ← bbox.x_br − bbox.x_tl;
    bbox_h ← bbox.y_br − bbox.y_tl;
    if now_size == None then
        now_size ← bbox_w × bbox_h;
    else if now_size × 3 < bbox_w × bbox_h then
        filtered_box_list.clear();
        filtered_box_list.append(bbox);
    else if now_size > 3 × bbox_w × bbox_h then
        CONTINUE;
    else
        filtered_box_list.append(bbox);
```

Figure 6 qualitatively shows the detailed steps of our tubing counting process. The original video gets the detection boxes through the object detection algorithm, and the processed image is shown in Figure 6(a). Subsequently, our filter algorithm will only consider object detection boxes within the region shown in Figure 6(b). Finally, our geometric correction algorithm will correct the filtered detection boxes and add an additional detection box in the above example, the result



(a) Step-1 (b) Step-2 (c) Step-3

Fig. 6. The Oil tubing counting step.

is shown in Figure 6(c). The gray detection box we circled with a blue ellipse in Figure 6(c) is the effect of our algorithm.

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.

2.5 Liquidity Information

In Sections 2.2–2.4, 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 can get liquidity information in this region.

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, and so on. 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. 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, and so on. 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.

3 EVALUATION

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

3.1 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. From the perspective of deployment and application, the system adopts layered architecture, which consists of three parts: terminal device layer, edge server layer, and user layer. The terminal equipment layer includes the network camera and other terminal equipment; The edge server layer is composed of the analysis server with GPU and the NAS storage server. The user layer refers to the front-end display devices such as desktops and laptops used by users to access intelligent video analysis systems. We use Hikvision cameras as fixed cameras, and connect to the Nvidia Jetson AGX Xavier development board as the edge device side. We invited 10 patrol officers to hold mobile phones as mobile cameras. We use a linux server cluster equipped with NVIDIA Tesla T4 as the edge server for processing video stream data.

In addition, the staff drive the video vehicle to the oil tubing stack yard in the open area to shoot the object of the oil tubing for material inventory. It should be noted that the video capture vehicle needs to keep a certain distance from the tubing stack, and try not to turn around near the tubing stack, otherwise the running accuracy may be reduced. It will be discussed in detail later. Besides, the speed of the vehicle needs to be controlled within a certain range to avoid the phenomenon of video stream.

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 [19]. For object tracking tasks, we chose the YOLObile framework in mobile and use YOLOv5l framework optimized for small objects in central server as the object detection framework, and retrained in our real samples. The object tracking part on mobile camera uses the DeepSORT algorithm. In addition, we use lightweight ReID algorithm [9] 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.

3.2 Modular Architecture

The system adopts a modular architecture and uses Docker containerization to construct, deploy and manage each module, which can isolate the environment of each module and facilitate system deployment and migration of each module. Video playback module saves RTSP stream slices into TS fragments based on GStreamer to provide video playback services. The edge video analysis

module is based on the DeepStream framework and the edge video analysis method mentioned above for intelligent analysis of video streams.

The above three modules will send the generated video footage information and video analysis results to the Kafka message communication module in real time during operation. The large-scale streaming data processing module based on Flink framework will receive Kafka messages in real time and save the message data to MySQL database and Redis cache database in real time after processing. In the process of processing stream data, the second analysis algorithm of video analysis results such as perimeter protection and violation event monitoring is also run, and the final analysis results are also stored in the MySQL database and Redis cache database. The Django back-end service module provides the back-end foundation of the system website, interacts with the MySQL database, and provides RESTful APIs for other modules to call. The data communication module uses the WebSocket protocol to transmit analysis data in real time to the front-end service module through a long connection. The front-end service module is implemented by Vue.js framework, and data interaction is carried out between the front and back ends through RESTful APIs. In addition, Nginx provides reverse proxy services to manage all network ports within the system in a unified manner, and exposes only one unified port to the outside.

3.3 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 1080 p, 30 fps 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 100 Mb of original video data will be generated every minute. The cellular network in the real oilfield is only 1 Mbps, 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 1 Kbps. For fixed cameras, the frame filtering algorithm only appends an additional byte of data per frame, which can be ignored for fixed optical fibers.

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

3.4 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.

3.4.1 Mini-tracking and Crowd Counting Delay. The retrained YOLObite + 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 1. Figure 7 shows the change curve of the algorithm processing speed when passing a group of object dense areas.

Table 1. 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%

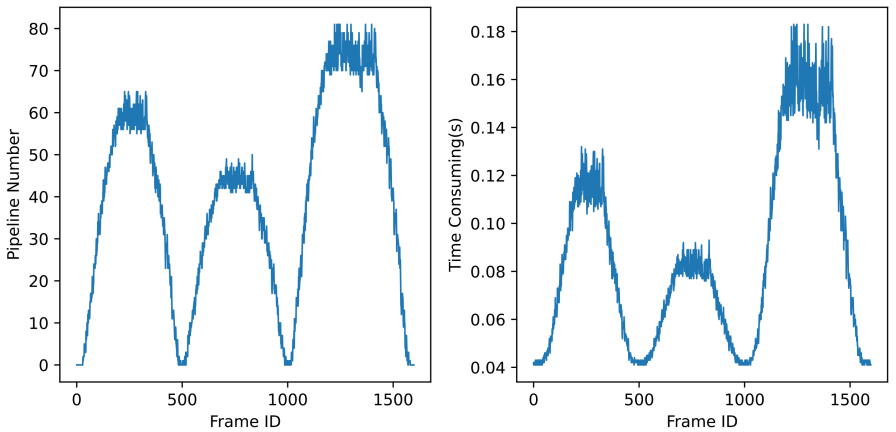


Fig. 7. It can be seen that the processing time of the video frame is positively correlated with the number of objects in the current frame.

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 15 fps 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 53 s.

3.4.2 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, our method can be used to process 24 channels of 30 FPS 1080 P 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.

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.

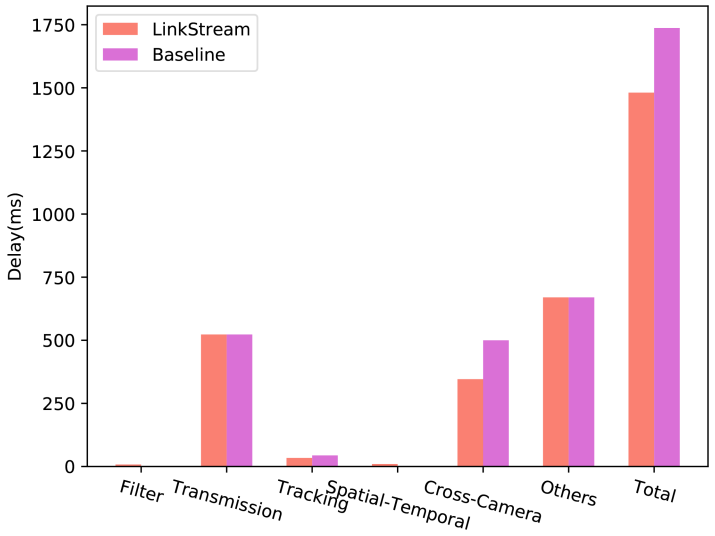


Fig. 8. Overall delay.

3.4.3 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 8 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 8 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.

3.5 Accuracy

3.5.1 Oil 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 2.

The results show that most of the experimental oil tubing stacks have error rates below 4%, except for 2.2 and 4.3. But 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.

Considering the two special samples with high error rates. The error of the tubing stack 2.2 is mainly due to the video collector turning near the tube stack, then the large amount of deformation caused by the turning causes the track-id of many tube to change. The error of tubing stack 4.3 is mainly due to the fact that the video collector is close to the tube stack, resulting in many incomplete tubing objects that cannot be detected in the video. These unrecognized oil tubes cause the algorithm to generate missing records. For normally collected video, this algorithm can better complete the task of tubing counting.

Table 2. Results of Oil Tubing Counting Accuracy

Case ID	Real value	Processing results	Error
1.1	191	196	−2.55%
1.2	91	92	−1.09%
1.3	241	239	0.08%
1.4	344	338	1.78%
2.1	341	338	0.88%
2.2	264	239	**10.46**%
2.3	262	255	2.75%
3.1	371	360	3.06%
3.2	103	101	1.98%
3.3	60	58	3.45%
4.1	52	51	1.96%
4.2	184	177	3.95%
4.3	310	343	**−9.62**%

Table 3. Ablation Experiment of Oil Tubing Counting

Case ID	Detection+Filter		Our Algorithm		Real Value
	Counting Result	Error	Counting Result	Error	
1.1	185	−5.61%	**191**	−2.55%	196
1.2	88	−4.34%	**91**	−1.09%	92
1.3	230	−3.77%	**241**	0.08%	239
1.4	**339**	0.30%	344	1.78%	338
2.1	328	−2.96%	**341**	0.88%	338
2.3	**262**	2.75%	**262**	2.75%	255
3.1	343	−4.72%	**371**	3.06%	360
3.2	**101**	0%	103	1.98%	101
3.3	61	5.17%	**60**	3.45%	58
4.1	**50**	−1.96%	**52**	1.96%	51
4.2	**180**	1.69%	184	3.95%	177

Further, we compare our algorithm with an algorithm that does not include geometry correction and only uses filter part. The results are shown in Table 3.

It can be seen that the tubing count designed in this article has better results in most cases. Some examples with worse effects are mainly caused by the neat placement of the original video stream tubing, the inability of the correction algorithm described in this article to take advantage, and the cancellation of positive and negative errors. Overall, the method described in this article has good performance in most cases.

3.5.2 Human and Vehicle Counting 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 4.

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.

Table 4. Results of Human and Vehicle Counting Accuracy

Case ID	Real value		Counting		Mini Counting	
	Vehicle	Human	Vehicle	Human	Vehicle	Human
1	0	20	0	20	0	19
2	0	45	0	44	0	47
3	7	13	7	13	6	13
4	10	20	10	22	9	16
5	5	45	5	43	3	42

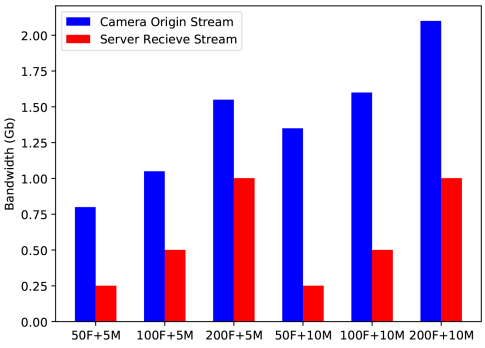


Fig. 9. Bandwidth.

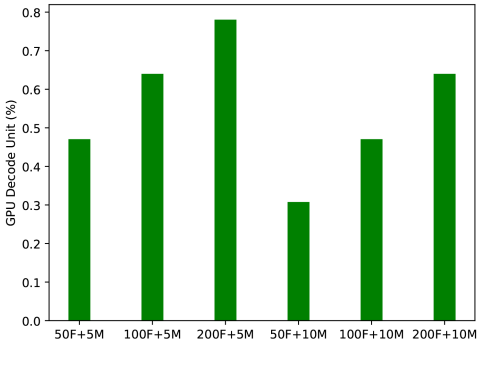


Fig. 10. GPU decode consumption.

3.6 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 9 shows the bandwidth overhead, which has been explained in Section 3.3. Figure 10 shows the relative occupancy rate of the GPU decode unit. 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 200 F+10 M, 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.

4 RELATED WORKS

Video Analysis System. In terms of hardware, some camera manufacturers, such as Hikvision, Omnicast, ProVigil, and so on, 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 [15, 24, 33], or configuring and customizing the content of different cameras to balance cost and efficiency [18] to optimize overall system performance. On the one hand, we draws on and extends some classic methods in existing work, such as frame filtering technology and parameter configuration for analysis of different video streams. On the other hand, we also optimizes the overall video stream processing in large-scale video stream scenarios.

Edge Computing. Input filtering is a simple and important solution to accelerate DNN model inference [23]. At the same time, the lightweight research of deep learning [1, 3, 4, 13, 14, 34] 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 [8, 16, 26] to save network overhead and central server resources. These works make it possible to run part of the algorithm on edge node near the fixed camera and mobile camera.

Spatio-temporal Correlation. A series of work has begun to consider using spatio-temporal information to optimize cross-camera analysis [6, 17, 21]. 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. We also performs search space tailoring based on spatio-temporal correlation information, and makes some optimizations for our system, including one-time initialization and self-learning, optimization using category and geolocation information, and so on.

Object re-identification. The task of object ReID is the most direct way to solve the same object matching of different video sources in cross-camera tracking. Pedestrian ReID is a traditional research direction in this field, including representation learning methods [7], metric learning methods [27], local feature-based methods [28], and GAN-based methods [35]. Besides, there has been some good progress in recent years in the field of vehicle ReID [20, 22]. On the whole, the research in the field of person re-identification and vehicle re-identification has made good progress in recent years. Currently, the BoT [25] series algorithms are widely used in the industry.

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 [5, 12] and deep learning [29–31]. For the task of ReID, good progress has been made in personnel [7, 35] and vehicles [20, 22], 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 [10], gaussian mixture model [21] or spatio-temporal information [17] to assist in the establishment of a complete motion trajectory. The existing cross-camera object tracking algorithm mainly considers how to use a better way to establish the motion trajectory information of the object on the basis of the single-camera object tracking results. This type of trajectory information can be naturally applied to the acquisition of the object cross-regional liquidity information studied in this article.

5 DISCUSSION

5.1 Index Selection

Delay and precision are the most important indexes of target detection. For target detection tasks, it is hoped that the delay can be reduced as much as possible, and the precision can be improved as much as possible. In order to measure the detection effect more objectively, this study chooses precision minus delay as utility function. In this way, the accuracy is improved or the delay is reduced, the detection effect is optimized, and the utility function value is also increased. The construction of utility function value makes the overall analysis effect of the task can be quantified, so that the increase of utility function value and analysis performance optimization can be established.

In order to improve the overall analysis performance of the system, the sum of utility function values of different RTSP video streams should be maximized. Although the utility function is defined as precision minus delay, it needs to multiply a coefficient before delay due to the difference in importance degree, order of magnitude or difference and other factors. However, this coefficient

is not invariable, different tasks have different tradeoffs of delay and accuracy, and the coefficient will be different to reflect the differential requirements of different tasks. In addition, although the overall analysis effect is measured by the summation of the utility function values of different RTSP video stream analysis, the importance of different tasks is also different, so the final sum should be weighted summation to reflect the priority of different tasks. The analysis effect of the more important task can better represent the overall analysis effect. Therefore, there should also be more weight in the weighted sum.

5.2 Wireless Networking

We address the network delay with **wireless Mesh network** (**WMN**), which is a network composed of a group of wireless nodes distributed in a Mesh topology in space, and the nodes are connected by wireless multi-hop mode to form a de-centralization communication architecture [2]. It is a special form of **wireless Ad-Hoc network** (**WANET**), so it inherits the characteristics of no center, no infrastructure, multi-hop, self-organization, and so on. Different from traditional wireless networks, WMN relies on a fixed access point AP, and each client communicates with AP in single-hop mode, which greatly limits the coverage range. The former uses a multi-hop communication mode, and the nodes are forwarded through multiple intermediate nodes, so as to achieve a more flexible networking structure, higher communication bandwidth and network congestion avoidance.

In response to the lack of visibility in self-organizing IoT management, this project develops an intelligent network management platform that integrates network fault diagnosis, network automatic configuration, network visualization and other network supervision requirements to support the long-term safe and stable operation of large-scale self-organizing IoT. The platform provides route security protection for Mesh networks, enhances the security of HWMP routing algorithm in Mesh networks, and realizes tamper-proof and authenticable routing information.

5.3 Resource Allocation

The resource monitoring module will monitor the GPU resource usage on the edge server, estimate the total detection frame rate that the system can carry, and pass it to the dynamic scheduling module. The effect evaluation module receives the final analysis results returned by the terminal device, and performs the following two tasks: one is to calculate the target motion speed in each time slice; the other is to evaluate the analysis effect regularly to update the relationship function between the accuracy rate and the detection frame rate of each video stream under different target motion speeds, and pass the evaluation results to the dynamic scheduling module. The dynamic scheduling module is responsible for implementing the dynamic scheduling algorithm, obtaining the optimal detection frame rate corresponding to each video stream in each time slice, and passing it to the video frame control module. The video frame control module will filter the decoded frames of each video stream according to the detection frame rate given by the dynamic scheduling module, and only retain the frames that need to perform detection tasks. Finally, these frames will be mixed according to the set batch size and then transmitted to the target detection module.

5.4 Scene Expansion

When deploying a large-scale multi-camera video analysis system to a new environment, the engineering team needs to conduct a comprehensive site survey and planning, understanding camera layout, lighting conditions, and scene characteristics. Subsequently, they will design a network architecture adapted to the large-scale transmission of video data and plan sufficient bandwidth, while adjusting the data storage system to handle the substantial data in the new scenario. Video analysis algorithms require optimization to adapt to the specific features of the new environment,

and real-time processing and response systems, as well as performance testing, are deployed to ensure the system operates efficiently in the new setting. The overall goal of the process is to optimize the system configuration in the new environment, enabling it to reliably handle video data and improve algorithm accuracy and system performance.

6 CONCLUSION

The advancement of the Internet of Things and deep learning technology has made it possible to establish systems for automated and intelligent analysis of video stream data, and the application of such systems in real life is becoming increasingly common. However, when faced with large-scale camera video data, existing methods face significant challenges in system load and latency. In this environment, the object tracking, counting, and liquidity monitoring of video streams have their own unique significance and problems. 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.

REFERENCES

[1] Xinjun Cai, Zheng Yang, Liang Dong, Qiang Ma, Xin Miao, and Zhuo Liu. 2023. DiTing: Edge assisted real-time ID-aware visual interaction for multi-user augmented reality. In *Proceedings of the 28th International Conference on Parallel and Distributed Systems*. 737–744.

[2] Zhichao Cao, Xiaolong Zheng, Qiang Ma, and Xin Miao. 2021. COFlood: Concurrent opportunistic flooding in asynchronous duty cycle networks. In *Proceedings of the Annual IEEE International Conference on Sensing, Communication, and Networking*. 1–9.

[3] Guoxuan Chi, Jingao Xu, Jialin Zhang, Qian Zhang, Qiang Ma, and Zheng Yang. 2023. Locate, tell, and guide: Enabling public cameras to navigate the public. *IEEE Transactions on Mobile Computing* 22, 2 (2023), 1010–1024.

[4] François Chollet. 2017. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1251–1258.

[5] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. 2017. Eco: Efficient convolution operators for tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6638–6646.

[6] Liang Dong, Zheng Yang, Xinjun Cai, Yi Zhao, Qiang Ma, and Xin Miao. 2023. WAVE: Edge-device cooperated real-time object detection for open-air applications. *IEEE Transactions on Mobile Computing* 22, 7 (2023), 4347–4357.

[7] Mengyue Geng, Yaowei Wang, Tao Xiang, and Yonghong Tian. 2016. Deep transfer learning for person re-identification. *CoRR* abs/1611.05244 (2016). http://arxiv.org/abs/1611.05244

[8] Seungyeop Han, Haichen Shen, Matthai Philipose, Sharad Agarwal, Alec Wolman, and Arvind Krishnamurthy. 2016. 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*. 123–136.

[9] Lingxiao He, Xingyu Liao, Wu Liu, Xinchen Liu, Peng Cheng, and Tao Mei. 2020. FastReID: A pytorch toolbox for general instance re-identification. *CoRR* abs/2006.02631. https://arxiv.org/abs/2006.02631

[10] Yuhang He, Xing Wei, Xiaopeng Hong, Weiwei Shi, and Yihong Gong. 2020. Multi-target multi-camera tracking by tracklet-to-target assignment. *IEEE Transactions on Image Processing* 29 (2020), 5191–5205.

[11] David Held, Sebastian Thrun, and Silvio Savarese. 2016. Learning to track at 100 fps with deep regression networks. In *Proceedings of the European Conference on Computer Vision*. Springer, 749–765.

[12] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. 2014. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37, 3 (2014), 583–596.

[13] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR* abs/1503.02531. http://arxiv.org/abs/1503.02531

[14] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient convolutional neural networks for mobile vision applications. *CoRR* abs/1704.04861 (2017). http://arxiv.org/abs/1704.04861

[15] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B. Gibbons, and Onur Mutlu. 2018. Focus: Querying large video datasets with low latency and low cost. In *Proceedings of the 13th {USENIX} Symposium on Operating Systems Design and Implementation*. 269–286.
[16] Puneet Jain, Justin Manweiler, and Romit Roy Choudhury. 2016. Low bandwidth offload for mobile AR. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*. 237–251.
[17] Samvit Jain, Xun Zhang, Yuhao Zhou, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, Paramvir Bahl, and Joseph Gonzalez. 2020. Spatula: Efficient cross-camera video analytics on large camera networks. In *Proceedings of the IEEE/ACM Symposium on Edge Computing*. 110–124.
[18] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: Scalable adaptation of video analytics. In *Conference of the ACM Special Interest Group on Data Communication*. 253–266.
[19] V. Kantorov and I. Laptev. 2014. Efficient feature extraction, encoding and classification for action recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
[20] Sangrok Lee, Eunsoo Park, Hongsuk Yi, and Sang Hun Lee. 2020. 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*. 608–609.
[21] Young-Gun Lee, Zheng Tang, and Jenq-Neng Hwang. 2017. Online-learning-based human tracking across non-overlapping cameras. *IEEE Transactions on Circuits and Systems for Video Technology* 28, 10 (2017), 2870–2883.
[22] Weipeng Lin, Yidong Li, Xiaoliang Yang, Peixi Peng, and Junliang Xing. 2019. Multi-view learning for vehicle re-identification. In *Proceedings of the IEEE International Conference on Multimedia and Expo*. 832–837.
[23] Luyang Liu, Hongyu Li, and Marco Gruteser. 2019. Edge assisted real-time object detection for mobile augmented reality. In *Proceedings of the 25th Annual International Conference on Mobile Computing and Networking*. 1–16.
[24] Yao Lu, Aakanksha Chowdhery, and Srikanth Kandula. 2016. Optasia: A relational platform for efficient large-scale video analytics. In *Proceedings of the 7th ACM Symposium on Cloud Computing*. 57–70.
[25] Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, and Wei Jiang. 2019. Bag of tricks and a strong baseline for deep person re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*.
[26] Xukan Ran, Haolianz Chen, Xiaodan Zhu, Zhenming Liu, and Jiasi Chen. 2018. Deepdecision: A mobile deep learning framework for edge video analytics. In *Proceedings of the Conference on Computer Communications*. 1421–1429.
[27] Rahul Rama Varior, Mrinal Haloi, and Gang Wang. 2016. Gated siamese convolutional neural network architecture for human re-identification. In *Proceedings of the European Conference on Computer Vision*. Springer, 791–808.
[28] Rahul Rama Varior, Bing Shuai, Jiwen Lu, Dong Xu, and Gang Wang. 2016. A siamese long short-term memory architecture for human re-identification. In *Proceedings of the European Conference on Computer Vision*. Springer, 135–153.
[29] Guangting Wang, Chong Luo, Xiaoyan Sun, Zhiwei Xiong, and Wenjun Zeng. 2020. Tracking by instance detection: A meta-learning approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6288–6297.
[30] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip H. S. Torr. 2019. Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1328–1338.
[31] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. 2017. Simple online and realtime tracking with a deep association metric. In *Proceedings of the IEEE International Conference on Image Processing*. 3645–3649.
[32] Mengwei Xu, Mengze Zhu, Yunxin Liu, Felix Xiaozhu Lin, and Xuanzhe Liu. 2018. Deepcache: Principled cache for mobile deep vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. 129–144.
[33] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J. Freedman. 2017. Live video analytics at scale with approximation and delay-tolerance. In *Proceedings of the 14th {USENIX} Symposium on Networked Systems Design and Implementation*. 377–392.
[34] Jialin Zhang, Xiang Huang, Jingao Xu, Yue Wu, Qiang Ma, Xin Miao, Li Zhang, Pengpeng Chen, and Zheng Yang. 2022. Edge assisted real-time instance segmentation on mobile devices. In *Proceedings of the IEEE International Conference on Distributed Computing Systems*. 537–547.
[35] Zhun Zhong, Liang Zheng, Zhedong Zheng, Shaozi Li, and Yi Yang. 2018. Camera style adaptation for person re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5157–5166.

Received 27 August 2023; revised 7 January 2024; accepted 11 February 2024