UNDERGROUND CRUDE OIL PIPELINE LEAKAGE DETECTION USING DEXINED DEEP LEARNING TECHNIQUES AND LAB COLOR SPACE

Muhammad H. Obaid[1]

[1]*Informatics Institute for Postgraduate Studies, Iraqi Commission for Computers and Informatics,*
ms202110670@iips.icci.edu.iq

Ali H. Hamad[2]

[2]*Department of Information and Communication Engineering, University of Baghdad*
ahamad@kecbu.uobaghdad.edu.iq

Abstract

Computer vision plays a big role in pipeline leakage detection systems and is one of the latest techniques. Still, it requires a powerful image-processing algorithm to detect objects. The purpose of this work is to develop and implement spill detection in oil pipes caused by leakage using images taken by a drone equipped with a Raspberry Pi 4. The acquired images are sent to the base station along with the global positioning system (GPS) location of the captured images via the message queuing telemetry transport Internet of Things (MQTT IoT) protocol. At the base station, images are processed to identify contours by dense extreme inception networks for edge detection(DexiNed) deep learning techniques based on holistically-nested edge detection(HED) and extreme inception (Xception) networks. This algorithm is capable of finding many contours in images. To find a contour with black color, the CIELAB color space (LAB) has been used. The proposed algorithm removes small contours and computes the area of the remaining contours. If the contour is above the threshold value, it is considered a spill; otherwise, it will be saved in a database for further inspection. For testing purposes, three different spill areas were implemented with spill sizes of ($1\ m^2$, $2\ m^2$, and $3\ m^2$). Images have been captured at three different heights (5 m, 10 m, and 15 m) by the drone used to capture the images. The result shows that effective detection has been obtained at 10 meters high. To monitor the entire system, a web application has been integrated into the base station.

Index items: Drone, oil spill, MQTT IoT Protocol, LAB color space, Dexi for Edge Detection, Xception networks, HED.

I. INTRODUCTION

Pipelines are the standard for transporting oil and gas in the modern world. It travels across water or land. As a result, it faces hazards, including vandalism, unintentional damage, structural damage from corrosion, theft, hot spots, punctures, etc. [1]. Prompt oil spill detection is crucial to protecting oil riches and this is a major generator of economic prosperity for the nations in question. The effectiveness of the leak detection monitoring system is compromised by several factors, including the use of conventional pipe monitoring techniques (such as visual monitoring and foot patrols along the pipe) and variations in pipe characteristics. It is also difficult to monitor pipelines passing through agricultural regions since such areas are privately owned. Not only that but there are several vulnerable points along pipeline routes [2].

Several pipeline leak detection technologies have been proposed over the last several decades, each with a unique working principle and approach [3]. External, visible or biological, and internal or computational are the three main categories into which these methods have been sorted by several different models [4]. The external technique relies on a wide variety of artificial sensing devices located away from pipes to detect pipes.

The biological method involves using trained dogs or human specialists to identify leakage. The internal strategy for the detection job consists of software-based solutions utilizing clever computational techniques and is backed up by sensors monitoring the internal pipeline environment. Using smart pigging, a helicopter, autonomous underwater vehicles (AUVs), drones, sensor networks, cameras, and other sensing equipment may be carried to distant monitoring sites [5]. Drones are now more durable and dependable due to advances in sensor technology and navigation systems. drones have been embraced by both the business and government sectors, and their use has spread to almost every industry, including the oil and gas industry.

When it comes to mapping, monitoring, inspecting, and surveilling oil and gas facilities, drones provide significant advantages over traditional methods since they are faster, safer, and more cost-effective [6]. Using artificial intelligence algorithms like machine learning and deep learning to process these images and extract the important features that could help to provide a decision about leakage, computer vision plays a crucial role in drones in a wide variety of fields, from controlling self-driving vehicles to performing tasks to navigating or monitoring oil pipelines [7]. This work aims to use a drone to monitor pipelines; the drone will be outfitted with a Raspberry Pi, Pi camera, and GPS positioning system. The drone utilizes the cloud-based MQTT IoT protocol to transmit photos and location data to the ground station. Using a deep learning method for computer vision is fundamental base station work. Thin edge maps of various spill sizes detected in each picture and even in a very small region are generated by the base station through the DexiNed algorithm. Since black is the spill's inherent color, the LAB method eliminates shadow's

influence on the picture. It determines the contour based on black alone. The perimeter of the black spot can be determined using an algorithm to calculate the area of contours that decides whether or not this is a leak. If there is a leak, the online app will immediately notify the proper authorities with the original photograph, the spot's size, and its geographical position. The rest of the paper is organized as follows. In the second part, we cover the most recent developments in this field. The final part details the theoretical underpinnings of the deep learning approach utilized in this study. Finally, the fourth portion details the suggested system's architecture and the results collected, while the fifth section presents the working conclusions.

II. RELATED WORK

Leak detection has been the major subject of many studies. Alharam et al. [8] suggested using a drone with a thermal camera to detect oil and gas pipeline leaks and cracks in remote and dangerous areas, on-board AI detects leaks in this system. GPS for gas to determine the position of spills. Sharafutdinov et al. [9] proposed installing the following devices on the UAV for the detection of oil spills: infrared, radar, laser radar, microwave radiometer sensor, and ultraviolet spectrometers. They allowed the spill to be determined by performing remote spectrum analysis, which helps determine the location and thickness of the oil slick and calculate the area for it.

Liu et al. [10] proposed to extract oil spill information using an image convolutional neural network) CNN (models from synthetic aperture radar (SAR) images by taking advantage of its features of local connection, weight sharing, and learning for image representation. Myssar et al. [11] proposed a framework for computer software that could be used to locate oil spills and river pollution. An unmanned aerial vehicle was used by the artificial intelligence-based framework to locate and identify oil pollution by analyzing the images it had taken. Ali et al. [12] corrosion is one of the faults that have a major influence on the safety of the surface of oil and gas tanks, thus a localizing visual inspection technology combining drones and AI has been presented. This technology employs an image processing algorithm, a fuzzy logic algorithm, and a threshold algorithm.

Ravishankar et al. [13] proposed the integration of drone technology with deep learning methodology, dubbed DARTS (Drone and Artificial Intelligence Reconsolidated Technological Solution). issue areas by regularly gathering and evaluating picture data. Results from DARTS's validation experiments demonstrate the tool's potential for improving pipeline systems' safety and resilience via proactive maintenance planning.

TABLE 1

SUMMARY OF RELATED WORK.

Ref.	Detection method	category	Technology	Tools	Algorithm
[8]	Visual monitoring of pipes.	Visual	AI-based on-board processing in real-time for leakage detection.	The drone is equipped with a thermal camera, Raspberry Pi3, GPS, and a gas detector.	ML algorithms.
[9]	Detection and prediction of oil pollution by drone.	Visual	Remote spectral analysis to determine an oil spill's extent and depth.	Infrared, microwave, ultraviolet spectrometers, radar, radiometer sensors, and laser radar.	Image analysis by a used software package, built in C++.
[10]	Sensing image remote to the oil spill.	Visual	SAR monitoring of marine oil spills.	CNN	AlexNet model.
[11]	Analyze captured images of rivers.	Visual	CNN model	Drones, camera, and MacBook Pro laptop.	ML.NET
[12]	Research is to detect rust in oil tanks.	Visual	Using AI to localize visual inspection technology through UAVs.	UAVs	Threshold algorithm, image processing algorithm, and fuzzy logic algorithm.
[13]	collecting and analyzing image data	Visual	DARTS	drone	deep learning algorithms
Our proposed work	Pipeline monitoring using computer vision.	Visual	DL algorithm.	Drone, Raspberry Pi, Pi camera, and GPS.	DexiNed, LAB algorithm, Spot area calculation algorithm, and MQTT IoT Protocol.

III. EDGE DETECTION BASED DEEP LEARNING

DexiNed is one of the CNN algorithms used for edge detection. It merges a reduction network up-sampling block (UB) with a dense extreme inception network (Dexi) to create DexiNed. Dexi creates feature maps from the input RGB (color system (red, green, and blue)) image and sends them to UB. The feature maps produce thin edge maps without losing edges in deeper layers, without prior training or fine-tuning [14].

1.DexiNed Architecture

DexiNed is a deep neural network that detects edges in color system red, green, and blue images (RGB) that are network inputs for W rows and H columns. DexiNed produces six side outputs with different scales corresponding to edge maps that represent an H x W map whose value is between zero and one. Mostly it is settled (0, 255), which always indicates the presence of an edge at every location in the image. Each of the six blocks has an output through which the two main outputs of the network are calculated: the average output and the fused output. The fused output is produced by a convolutional layer with a kernel of 1 x 1, its inputs are the six side outputs, and the average output is calculated by the average between all the side outputs and the fused output, as shows in fig.1 [15]. The

DexiNed network consists of six blocks, where each block consists of repeated sub-blocks. The first block input is a BGR image. Its sub-blocks consist of a conv layer with a 3 x 3 kernel, rectified linear unit (ReLU) activation functions, and batch normalization. There are some differences in the composition of the six blocks, especially the first and second blocks. These blocks are completely different from the rest of the other four blocks in terms of the presence or absence of ReLU layers before the up-sampling blocks. There are lateral connections, and the 1 x 1 kernel convolutional layers produce all the lateral connections. The six blocks operate at different scales. The first and second blocks of the input image operate at half-full resolution, while the third block operates at 1/4 accuracy, the fourth block operates at 1/8 accuracy, and the fifth block operates at 1/16 accuracy [16].

It is possible to express DexiNed in terms of a regression function ð, that is, Yˆ = ð(X, Y), where X is an input image, Y is the corresponding ground truth, and Yˆ It is a collection of predicted edge mappings. Yˆ = [yˆ_1, yˆ_2, ..., yˆ_N], where yˆ_i has the same size as Y and is the number of outputs from each upsampling block; yˆ_N is the result from the final fusion layer f (yˆ_N = yˆ_f). Therefore, as this is a model, the same loss as (weighted cross-entropy) is used; hence, this problem is thoroughly supervised and addressed similarly [17].

$$\ell^{n}(\mathbf{W}, \mathbf{w}^{n}) = -\beta \sum_{j \in Y^{+}} \log \sigma\left(y_j = 1 | X; \mathbf{W}, \mathbf{w}^{n}\right) - (1-\beta) \sum_{j \in Y^{-}} \log \sigma\left(y_j = 0 | X; \mathbf{W}, \mathbf{w}^{n}\right), \quad (1)$$

Then,

$$\mathcal{L}(\mathbf{W}, \mathbf{w}) = \sum_{n=1}^{N} \delta^{n} * \ell^{n}(\mathbf{W}, \mathbf{w}^{n}), \quad (2)$$

Each scale has its weight, denoted as W is the set of all network parameters, w is the parameter with index, and δ is the weight for each scale level. β = |Y^-|/|Y^+ +Y^-| and (1−β)=| Y^+|/|Y^++ Y^-|.Where (|Y^-|, |Y^+|) represent the edge and non-edge in the underlying truth, respectively. By comparison with other edge detection techniques using basic metrics, the results were superior to those techniques. the accuracy of (per-image best threshold (OIS)=0.867, fixed contour threshold (ODS)=0.859, and average precision (AP)=.905) on Barcelona images for perceptual edge detection (BIPED) test dataset [18].

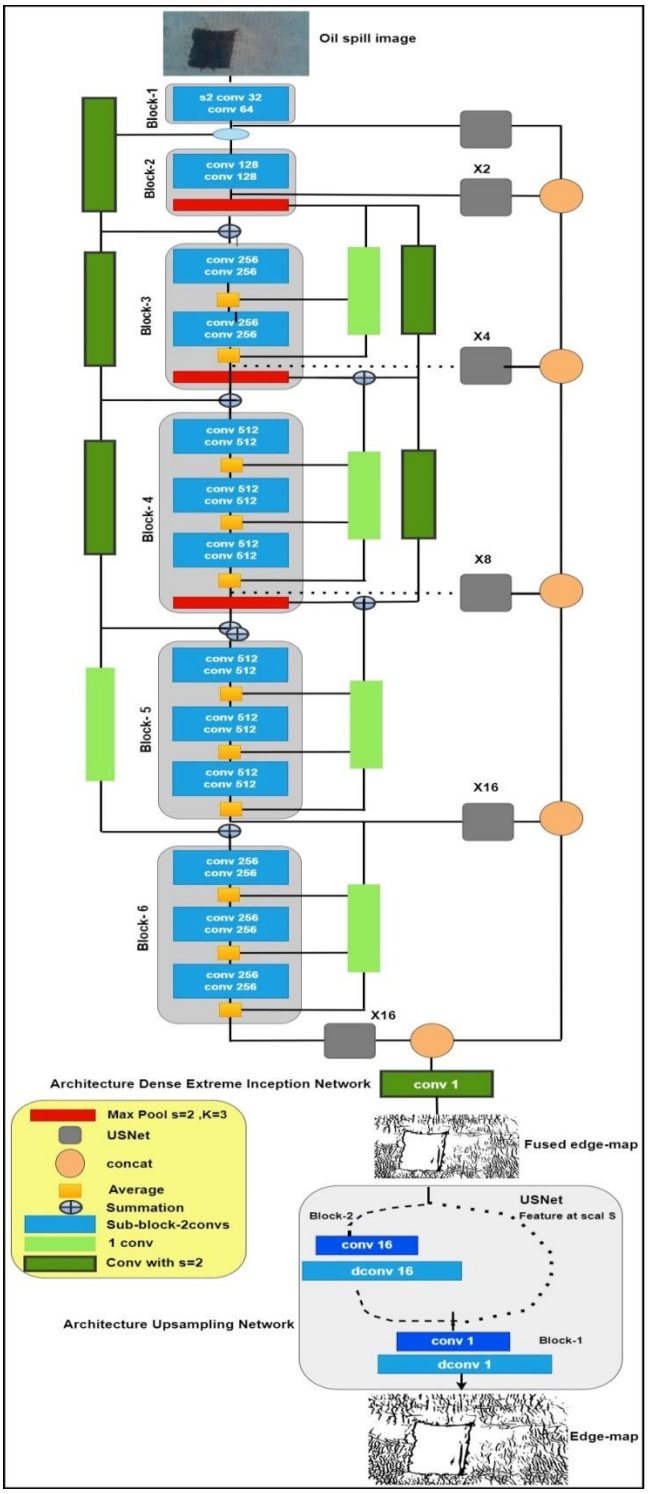


Fig. 1 DexiNed architecture [15].

2. Xception Networks

The Inception module is a depth-wise separable convolution operation bridge to the more traditional convolution [18]. Xception's key feature is its extreme inception architecture. The

core of Xception is depthwise separable convolution, which is how the algorithm works. The Xception model modified the conventional inception block to be more all-encompassing, switching to a single dimension (3 × 3) and then doing a 1×1 convolution to lessen the burden on the computer's processing power [19].

Xception's architecture implements the depthwise separable convolution operation. There are two types of convolution: depthwise and pointwise and they can be considered separable convolution layers. The three parts of the Xception architecture are entry flows. First, a stem of two convolutional layers of increasing sizes, followed by the first layer of the model, and then three downsampling blocks. Each of these blocks has two separable convolutional layers with a kernel size of 3, combined with a Max Pooling layer. Each block has a skip connection with 1x1 convolution with stride 2 middle flow. The central unit contains eight Xception blocks. Each block has three separable convolutional layers with a kernel size of 3 and stride 1. The method applied does not reshape the input size. For this reason, the feature map size remains 19×19×728 in this part of the network. In addition, there is a residual identity connection around every block in the Exit flow. The closure section starts with one downsampling block, like the ones in the entry flow, followed by two separable convolutions. Lastly, there is a classification head with global average pooling and fully connected layer(s) [20].

3. Holistically-Nested Edge Detection

The HED strategy is neural network-based, with visual geometry group(VGG16) as the central network node. It is a deep neural network-based edge detector for images. The algorithm takes a color picture as input. It outputs a map of edges with confidence scores ranging from 0 to 1 (or, equivalently, 0 to 255) for the existence of boundaries in each pixel. The HED strategy bases itself on the VGG16 network's convolutional layer (encircled by a dotted box in Fig. 2). This section comprises 13 convolutional layers with 3×3 kernels and ReLU activation functions after each layer. The network consists of five subnetworks, each of which processes images at a different scale: the first subnet processes images at the full resolution of the input picture, while the others process images at half, quarter, eighth, and sixteenth that size. A max-pool operation with a 2×2 kernel and a 2×2 stride decreases the resolution while moving from one set to the next. The final result of HED combines the outputs of five VGG16 groups trained to represent edge maps at five distinct scales [21] [22].

During training, the input training data set is represented as $S = \{(X_n, Y_n), n = 1, \dots, N\}$ where sample $X_n = \{x_j^{(n)}, j = 1, \dots, |X_n|\}$ represents the raw input image and $Y_n = \{y_j^{(n)}, j, \dots, |X_n|\}$ $y_j^{(n)} \in \{0,1\}$ represents the matching ground truth binary edge map for the image X_n. Weights are represented as $w = w^{(1)}, \dots, w^{(M)}$, and the network comprises M side-output layers. The loss function is computed at the picture level for auxiliary outputs in the field of image-to-image training [23]:

$$\ell_{side}^{(m)}(W, w^{(m)}) = -\beta \sum_{j \in Y_+} log\, Pr(y_j = 1 | X; W, w^{(m)}) - (1-\beta) \sum_{j \in Y_-} log\, Pr(y_j = 0 | X; W, w^{(m)}) \quad (3)$$

Can determine the fusion layer loss function $\mathcal{L}_{\text{fuse}}$ by:

$$\mathcal{L}_{fuse}(W, w, h) = Dist(Y, \hat{Y}_{fuse}) \quad (4)$$

To optimize for the minimum value of the following objective function using (backpropagation) accidental fall down a gradient by:

$$(W, w, h)^* = argmin\, \mathcal{L}_{side}(W, w) + \mathcal{L}_{fuse}(W, w, h) \quad (5)$$

Both the side output layers and the weighted-fusion layer's predictions on the edge map may be used in the testing step with image X by:

$$\hat{Y}_{fuse}, \hat{Y}_{side}^{(1)}, \dots, \hat{Y}_{side}^{(M)} = CNN(X, (W, w, h)^*), \quad (6)$$

These created edge maps may aggregate further to provide a single result by:

$$\hat{Y}_{HED} = \text{Average}(\hat{Y}_{\text{fuse}}, \hat{Y}_{\text{side}}^{(1)}, \dots, \hat{Y}_{\text{side}}^{(M)}) \quad (7)$$

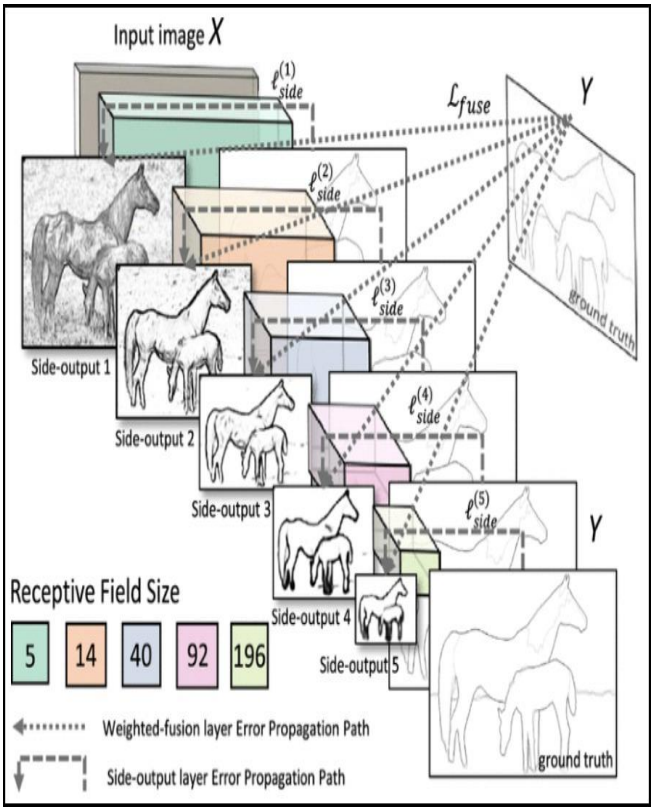


Fig. 2 HED architecture [21]

4. LAB Color Space

A luminance (lightness) channel, and two additional channels, A and B, represent different chromaticity layers in this color space. Where a color lies on the red-green axis can be determined from the A* layer, and where it lies on the blue-yellow axis may be selected from the B* layer. The fact that this color space may convey color information across multiple platforms and devices is its most notable characteristic. Coordinates in L*A*B* color space, the range of possible values for L* is from 0 (complete darkness) to 100 (total lightness) brightness (L*) is displayed along the middle vertical axis. It follows from the coordinate axes that A* color can't be blue or yellow since these are opposing colors [24].

All axes display values from positive to negative. The A channel, whose values range from -128 to +127, reveals the color balance between red and green. The B channel, whose values range from -128 to +127, also specifies the image's relative amount of yellow and blue. Colors with a high value in the A or B channels tend to be red or yellow. In contrast, colors with a low value tend to be green or blue. The zero point shows grayscale neutrality on both axes.

An image with RGB is converted to a LAB image by converting it to grayscale images or binary images. This is done to detect edge information in the image. The space RGB is converted to space XYZ and then converted to space LAB. The value range is R, G, and B = (0 ~255), L=(0~100), and A, B=(-128~+128). The color values of R, G, and B are converted to X, Y, and Z by Equation (8) [25].

$$\begin{cases} X = 0.49 \times R + 0.31 \times G + 0.2 \times B; \\ Y = 0.177 \times R + 0.812 \times G + 0.011B; \\ Z = 0.01 \times G + 0.99 \times B. \end{cases} \quad (8)$$

The values X, Y, and Z are converted into LAB using Equation (9):

$$\begin{cases} L = 116 f_Y - 16; \\ A = 500 \times \left(\frac{f_X}{0.982 - f_Y}\right); \\ B = 200 \times \left(f_Y - \frac{f_Z}{1.183}\right). \end{cases} \quad (9)$$

IV. MESSAGE QUEUING TELEMETRY TRANSPORT

The MQTT protocol is a free and open implementation of the transmission control protocol (TCP) transport layers publish-subscribe model. Using a broker, customers in a publish/subscribe model may either subscribe to or publish on relevant subjects. Any client nodes that have subscribed to the topic will get the message when a client "publishes" to it. A publisher and a subscriber are the two possible roles for a node in the MQTT protocol. Fig.3 shows a simplified representation of a publisher/subscriber network. The protocol maximizes dependability and provides delivery assurance while minimizing the burden on networks and devices. Furthermore, the protocol is well-suited for machine-to-machine (M2M) communications among IoT gadgets due to its respect for limited resources like bandwidth and battery life [26] [27].

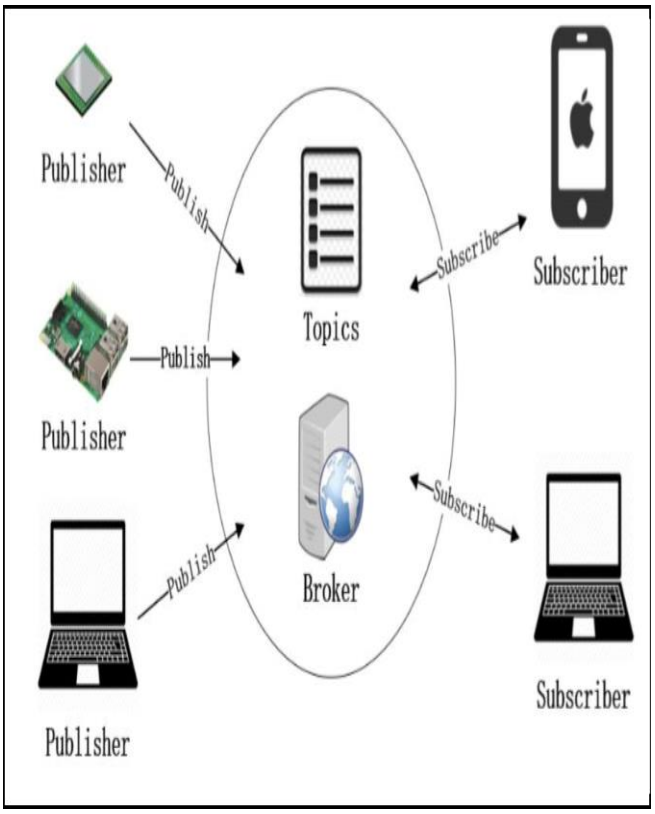


Fig.3 Schematic diagram of the proposed publisher /subscriber exchange.

V. PROPOSED SYSTEM

The most important design criterion when designing leakage detection systems for underground crude oil pipelines is that the proposed system should detect spills of different sizes. There are various causes for spills, such as the pipe's age or damage due to the surrounding environment, as well as sabotage or theft. The spill size could increase over time due to cautious leakage. The second criterion is spill color, which in most cases is black due to the nature of the oil. The spill color does not change over time.

The last design criterion is the location of the leakage. Most detection methods, such as differential pressure, can detect a leak in the pipe segment (up to 30 km) but cannot provide the exact location within the segment, so it requires another mechanism to find the spill location. Fig. 4 shows part of the practical side of implementing the system, the work steps can be summarized:

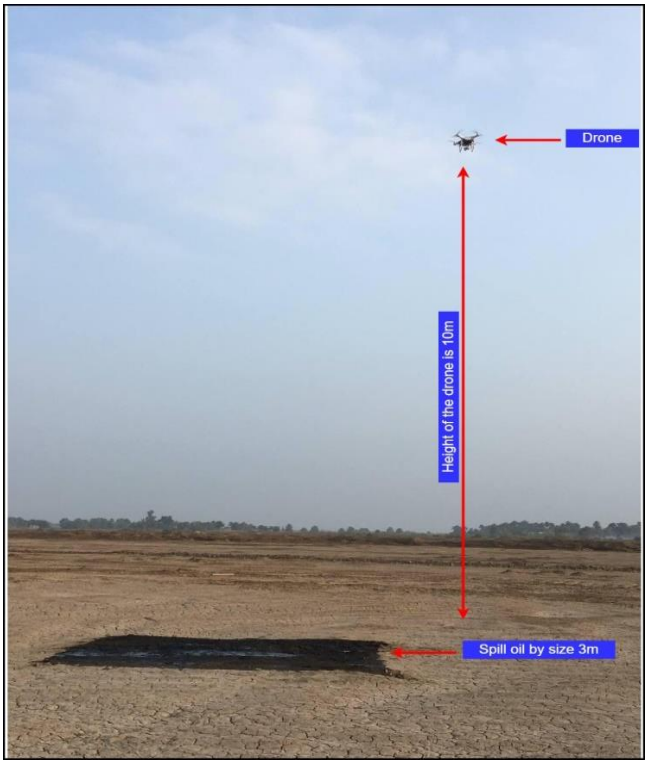


Fig. 4 Drone taking pictures of the oil leakage

1. Experimental Work

In this work, an embedded system represented by a Raspberry Pi 4 with a Pi camera and a GPS module was used to detect the spill size and location within any part of the pipes. The embedded system is flown by air by a drone which is sent on a tour after it is given the coordinates of the routes taken by the pipelines. Fig. 5 shows the design and implementation of the proposed system.

The drone is sent repeatedly along the pipe buried at a depth of three meters almost. An oil pipeline extending 30 kilometers in the desert regions of Iraq was used to test the system. Each time, the embedded system takes pictures along the route at a rate of one image every 0.2 seconds, resulting in an image every 1 meter of the path. GPS coordinates are included with the image when it is sent to the base station using the MQTT IoT protocol. the DexiNed algorithm analyzes the picture for distinct contours. There might be several shapes in each image, but one could be a spill. As a result, minor contour details would be ignored. Then, the LAB method is used to locate the edge of a picture with a single black color spot, a possible spill.

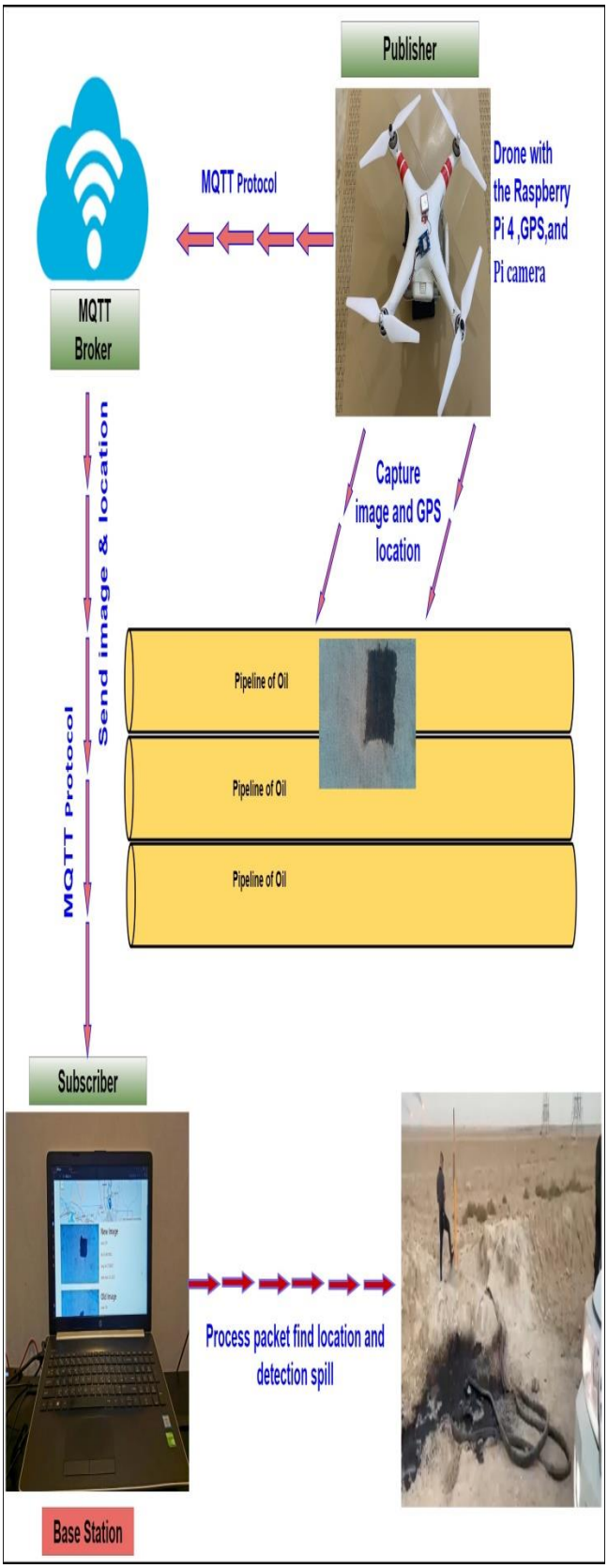


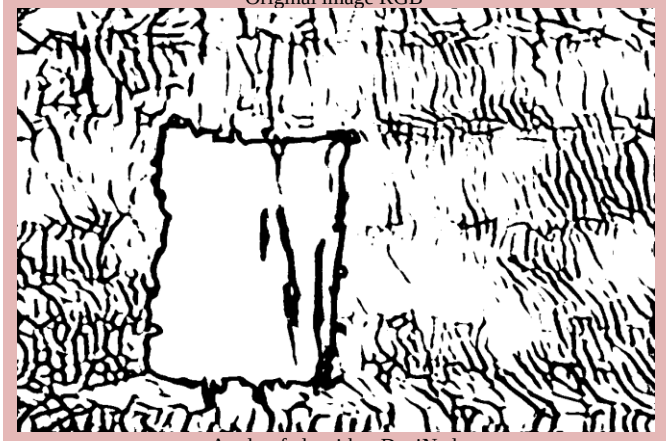
Fig. 5 Proposed system design and implementation.

2. Base Station Design

The proposed web application has been designed as a base station to receive data packets from the drone using the MQTT protocol, which acts as the publisher while the web application acts as a subscriber. First, the application receives images with its GPS location (longitude and latitude). After receiving this information, the web applies a different algorithm to process the data and finally provides a decision about the spill. The application uses Flask library-based Python 3.9 with a database in Microsoft Access. Three algorithms have been implemented in this work:

The first algorithm represents the MQTT IoT Protocol's operation in the publish/subscribe schema, where the drone and Raspberry Pi 4 are the publishers and the base station is the subscriber to the working mechanism. The second algorithm represents applying the DexiNed deep learning algorithm on the received images, defining the contour for all image parameters, and deleting the minimal contour less than the threshold limit. Finally, the third algorithm applies the LAB algorithm to find contours with only black features, which is the spill's natural color, and sends a warning via the web application of the spill's location if the spot area is greater than the threshold limit, or it is saved in the database at the current time and checked in the next round of the drone. Fig. 6 shows the results of operations on the captured image.

Original image RGB



Apply of algorithm DexiNed

Area in pixel for all detected contours

Applied the LAB algorithm and find the real area of the spill

Fig. 6 (3 m^2)spill with (10 m) height.

Algorithm.1: MQTT IoT Protocol

```
1   Base station sends the message to the drone to start
    the operation.
2     {
3       While (True): {
4           Image=capture (image_ path)
5           [latitude, longitude ]=read GPS location
6           Payload message =image+ latitude+
            longitude
7           Publish ( Payload) to Base Station
8            }
9       }
```

Algorithm.2: Apply DexiNed

```
1   Read image
2   Apply DexiNed
3   Number of contours =n;
4   {
5     While (n!=0) {
6       If contour _ area _ in pixel(small)
7             {
8               Eliminate contour;
9               n--;
10            }
11      Else {
12              Contour _ counts++;
13              n--;
14            }
15        }
16  }
```

Algorithm.3: LAB algorithm

```
1   Read image i from DexiNed.
2   While (True):
3   {
4    Apply the LAB algorithm.
5    If (black _ contour) exists
       If (area _ black_contour> threshold)
6        {
     Send an alarm message to find leakage by a web
     application.
7          Set a new round.
8       }
       Else
9          {
             Save the image in a database.
10         }
11   Else
12      {
13       Check the database for the stored image.
14         If the spill exit in a database
15           {
16            Get the GPS location of the spill.
17            Send the drone to a location.
18           }
19         Else
20           {
21             Set a new round.
22           }
23      }
24  }
```

3. Area Calibration Procedure

To compute the actual area of the spill, extensive tests were applied in this work, where three sizes of the spill were done ($1\ m^2, 2\ m^2$, and $3\ m^2$), as shown in Fig. 6 The drone is set directly above the spill for each area size at three different heights (5m, 10m, and 15m). For each high, a set of images was captured (up to 10). Table .2 shows a statistical analysis for those images that were applied to compute the area of the spill in pixels and also the spill diameter.

The spill was detected using the DexiNed deep learning algorithm, where it could result in a lot of contours, a large spill contour, or even small contours. Therefore, an algorithm is required to eliminate these contours and isolate the oil slick by means of color from the rest of the slicks. Fig. 7 and Fig.8 show the result of the comparison between the pixel area and the perimeter of the three spills at the three heights. By doing this, the actual area can be estimated from the pixel area, and thus the small or large perimeter can be determined easily.

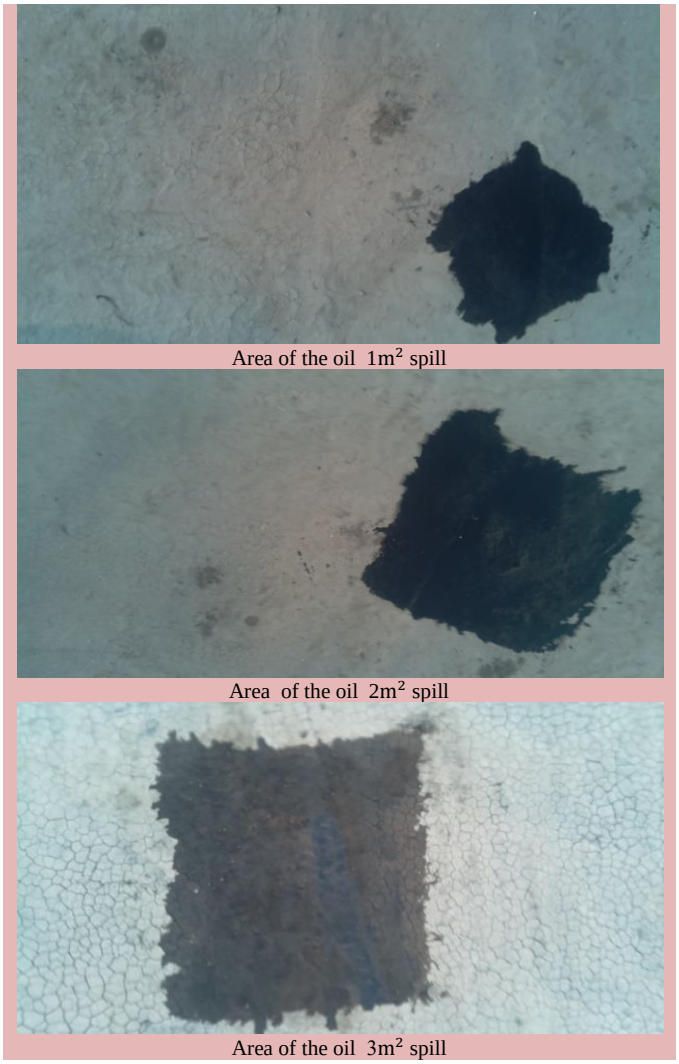


Fig. 6 Image of spilled oil of different sizes taken by a drone.

TABLE 2
STATISTICAL ANALYSIS OF A GROUP OF IMAGES CAPTURED IN PIXELS

Height=5 and The spill is(1 m^2)		
image	pixel area	perimeter area
1	101204.5	2478.83
2	97713.5	1816.19
3	96422.0	1568.59
4	87357.0	1384.76
5	91652.0	1373.94
Height=5 and the spill is 1 m^2		
1	23219.0	725.41
2	23019.5	797.25
3	23103.5	814.28
4	23099.0	762.38
5	23122.0	792.52
Height=15 and the spill is 1 m^2		
1	8078.0	457.1
2	7610.5	363.12
3	8760.0	499.16
4	8342.0	431.91
5	8041.0	393.42

Height=5 and the spill is(2 m^2)		
image	pixel area	perimeter area
1	158437.0	2032.64
2	153535.0	2030.09
3	158332.5	2015.99
4	184341.0	2309.71
5	162207.5	2083.99
Height=10 and the spill is(2 m^2)		
1	61926.0	1170.63
2	63696.0	1396.94
3	64244.5	1311.79
4	60996.5	1245.02
5	59781.5	1230.73
Height=15 and the spill is(2 m^2)		
1	27382.5	767.71
2	27901.0	753.47
3	27481.0	711.04
4	27127.0	703.39
5	25840.0	864.1

Height=5 and the spill is (3 m^2)		
image	pixel area	perimeter area
1	259915.0	2758.26
2	254728.0	2932.97
3	254765.0	3213.02
4	251951.0	3627.29
5	261107.5	3273.14
Height=10 and the spill is (3 m^2)		
1	143821.0	2523.32
2	138306.5	1932.6
3	135590.5	2400.83
4	138213.0	2475.24
5	138131.5	2364.26
Height=15 and the spill is(3 m^2)		
1	68020.5	1916.86
2	62248.0	1232.75
3	61547.5	1240.65
4	62929.5	1296.65
5	63246.5	1229.6

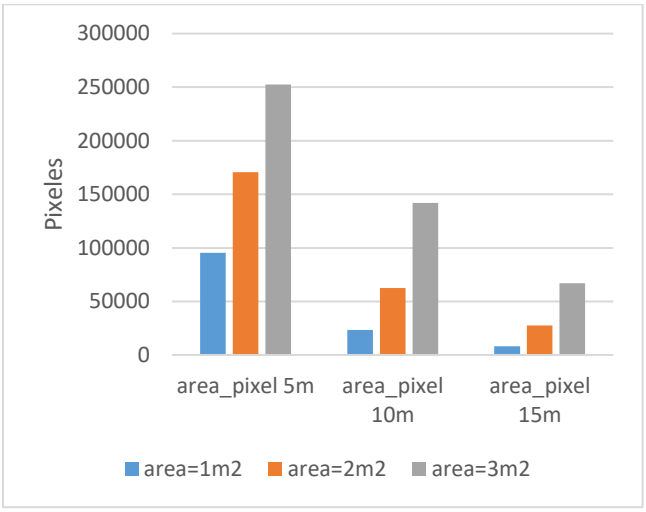


Fig.7 Average pixel area of a set of leakage images.

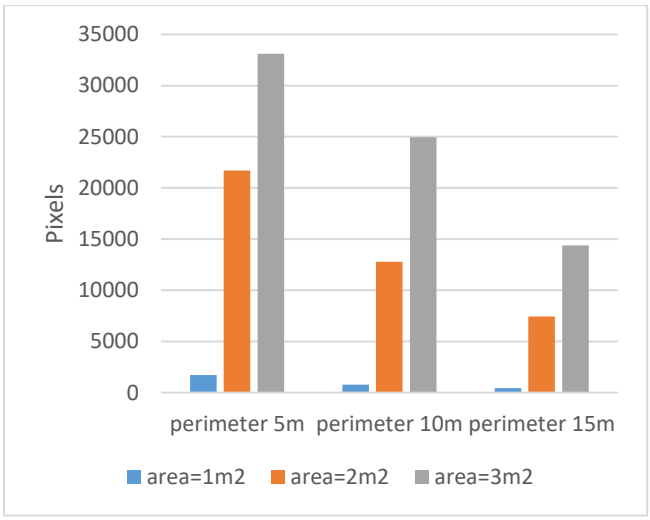


Fig. 8 The average perimeter of a set of leakage images.

4. Web Application

During the first round of the drone, the search results are displayed on a web page for the data received from the drone via the MQTT IoT protocol. After completing the processing process, the search results are displayed as shown in Fig. 8, where if one of the images contains a spot and the contour area is two meters, it is considered a leak, and the details appear, which contain map information that displays the location and spill information (volume of the spill, location of the spill, date when the spill was seen, and area of spill). In Fig. 9, the distance between the main station and the spot location is shown on the map.

But if the spot contour area is less than the minimum ($1m^2$), it is not considered a leak, and the image with its location is kept in the database to monitor it. If the spot perimeter area is greater than the threshold limit in other rounds of the drone compared

to the stored spot size and location, it will be shown as shown in Fig. 10 with the first case before it is considered a leak.
The system makes it possible to display a summary of each round that the drone performs by displaying all images that are detected and suspected oil spills that have been pre-processed.

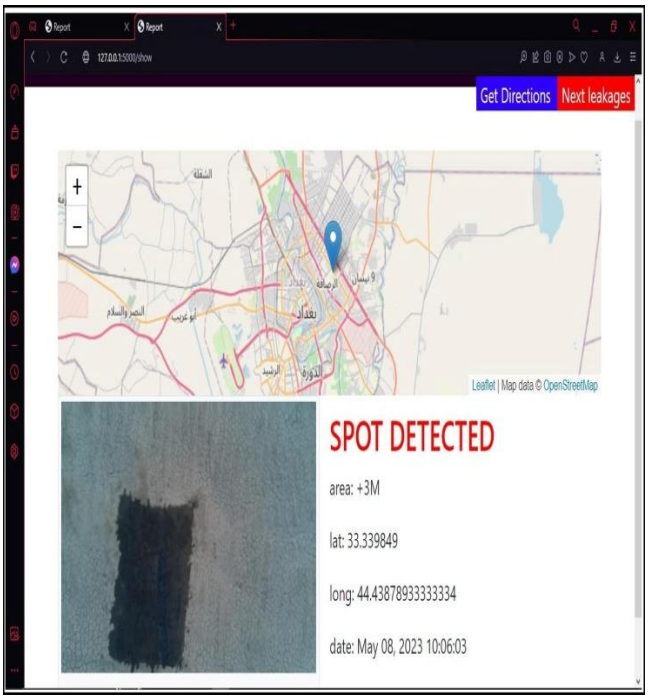


Fig. 8 The main interface of the program for the presence of an oil spill is greater than the threshold limit.

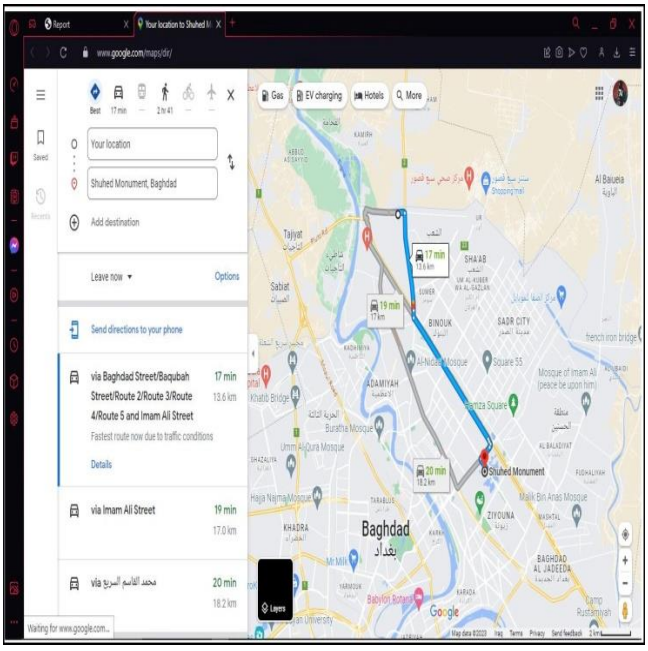


Fig. 9 Distance between the spill and the main station.

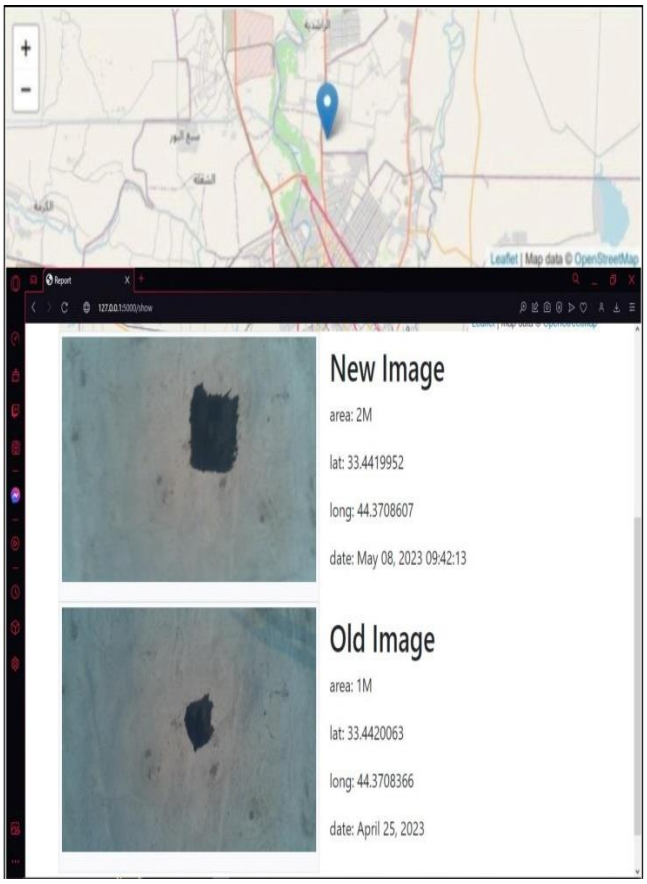


Fig. 10 Main program interface for an oil spill expansion

VI. CONCLUSIONS

This work proposes the design and evaluation of a UAV image processing and computer vision monitoring system based on deep learning algorithms for Iraqi crude oil pipeline systems in desert areas. Thin edge maps, including oil slick identification, were generated by implementing the DexiNed algorithm that proved accuracy by comparison with other edge detection techniques using basic metrics, the results were superior to those techniques (per-image best threshold (OIS)=.867, fixed contour threshold (ODS)=.859, and average precision (AP)=.905) in the base station. The LAB algorithm has proven effective in accurately distinguishing black spills from other spots, calculating spill area, and determining if it exceeds a threshold. The best results were obtained when the aircraft was at a height of 10 meters compared to other heights.
However, some factors negatively affect the operation of the system, including the weather conditions during the inspection rounds, and the accuracy of the image varied according to the time of day, as well as the terrain through which the pipes pass.

ACKNOWLEDGMENT

Iraqi Commission supports this work for Computers and Informatics /Informatics Institute for Postgraduate Studies.

REFERENCES

[1] Wang, Qinying, et al. "Evolution of corrosion prediction models for oil and gas pipelines: From empirical-driven to data-driven." Engineering Failure Analysis (2023) : 107097.

[2] Li, Hao-Jie, et al. "Detecting pipeline leakage using active distributed temperature Sensing: Theoretical modeling and experimental verification." Tunnelling and Underground Space Technology 135 (2023): 105065.

[3] Guo, Dongsheng, et al. "Analysis of the Influencing Factors of the Leak Detection Method Based on the Disturbance-Reflected Signal." Energies 16.2 (2023): 572.

[4] Korlapati, Naga Venkata Saidileep, et al. "Review and analysis of pipeline leak detection methods." Journal of Pipeline Science and Engineering (2022): 100074.

[5] Murugan, J. Senthil, et al. "A Study on Oil Spill Detection in Ocean with Look a likes." Mathematical Statistician and Engineering Applications 72.1 (2023): 90-97.

[6] Korlapati, Naga Venkata Saidileep, et al. "Review and analysis of pipeline leak detection methods." Journal of Pipeline Science and Engineering (2022): 100074.

[7] Korlapati, Naga Venkata Saidileep, et al. "Review and analysis of pipeline leak detection methods." Journal of Pipeline Science and Engineering (2022): 100074.

[8] Alharam, Aysha, et al. "Real time AI-based pipeline inspection using drone for oil and gas industries in Bahrain." 2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT). IEEE, 2020.

[9] Sharafutdinov, Azat A., et al. "Development of a method for calculating fire and oil spills parameters." AIP Conference Proceedings. Vol. 2216. No. 1. AIP Publishing, 2020.

[10] Wang, Xinzhe, et al. "Detection of oil spill using SAR imagery based on AlexNet model." Computational Intelligence and Neuroscience 2021 (2021).

[11] AL-BATTBOOTTI, Myssar Jabbar Hammood, G. O. G. A. Nicolae, and Iuliana MARIN. "Detection and Analysis of Oil Spill using Image Processing." International Journal of Advanced Computer Science and Applications 13.4 (2022).

[12] Ali, Mohammed AH, et al. "An Automatic Visual Inspection of Oil Tanks Exterior Surface Using Unmanned Aerial Vehicle with Image Processing and Cascading Fuzzy Logic Algorithms." Drones 7.2 (2023): 133.

[13] Ravishankar, Premkumar, et al. "Darts—drone and artificial intelligence reconsolidated technological solution for increasing the oil and gas pipeline resilience." International Journal of Disaster Risk Science 13.5 (2022): 810-821.

[14] Bakirman, Tolga, Bahadir Kulavuz, and Bulent Bayram. "Use of Artificial Intelligence Toward Climate-neutral Cultural Heritage." Photogrammetric Engineering & Remote Sensing 89.3 (2023): 163-171.

[15] Soria, Xavier, et al. "Dense extreme inception network for edge detection." Pattern Recognition 139 (2023): 109461.

[16] Grompone von Gioi, Rafael, and Gregory Randall. "A brief analysis of the dense extreme inception network for edge detection." IPOL. Journal Image Processing On Line, no 12, Oct 2022, pp. 389-403. (2022).

[17] Poma, Xavier Soria, Edgar Riba, and Angel Sappa. "Dense extreme inception network: Towards a robust cnn model for edge detection." Proceedings of the IEEE/CVF winter conference on applications of computer vision. 2020.

[18] Soria, Xavier, et al. "Dense extreme inception network for edge detection." Pattern Recognition 139 (2023): 109461.

[19] Alzubaidi, L. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J. Big Data 8 (1), 1–74 (2021)."

[20] Shavit, Hadar, et al. "From Xception to NEXcepTion: New Design Decisions and Neural Architecture Search." arXiv preprint arXiv:2212.08448 (2022).

[21] Grompone von Gioi, Rafael, and Gregory Randall. "A brief analysis of the holistically-nested edge detector." IPOL. Journal Image Processing On Line, no 12, Oct 2022, pp. 369-377 (2022).

[22] Yu, Naigong, et al. "An improved method for cloth pattern cutting based on holistically-nested edge detection." 2021 IEEE 10th Data Driven Control and Learning Systems Conference (DDCLS). IEEE, 2021.

[23] Misra, Indranil, et al. "SPRINT: Spectra Preserving Radiance Image Fusion Technique using holistic deep edge spatial attention and Minnaert guided Bayesian probabilistic model." Signal Processing: Image Communication 113 (2023): 116920.

[24] Dong, Lili, Weidong Zhang, and Wenhai Xu. "Underwater image enhancement via integrated RGB and LAB color models." Signal Processing: Image Communication 104 (2022): 116684.

[25] He, Haojie, et al. "Tomato disease degree recognition based on RGB and Lab color space conversion method." 2023 IEEE 6th International Conference on Industrial Cyber-Physical Systems (ICPS). IEEE, 2023.

[26] Alatram, Alaa, et al. "DoS/DDoS-MQTT-IoT: A dataset for evaluating intrusions in IoT networks using the MQTT protocol." Computer Networks 231 (2023): 109809.

[27] Alshammari, Hamoud H. "The internet of things healthcare monitoring system based on MQTT protocol." Alexandria Engineering Journal 69 (2023): 275-287.