CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

Computer Vision Based Pipeline Tracking for Autonomous Unmanned Aerial Vehicle in Unburied Pipeline Onshore Inspection

Yago Melo Rodrigues da Silva

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Orientadores:
Milena Faria Pinto, D.Sc.
João Terêncio Dias, D.Sc.

Rio de Janeiro,
Abril, 2023

Computer Vision Based Pipeline Tracking for Autonomous Unmanned Aerial Vehicle in Unburied Pipeline Onshore Inspection

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em nome do programa, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Yago Melo Rodrigues da Silva

Aprovada por:

gov.br Documento assinado digitalmente
MILENA FARIA PINTO
Data: 13/04/2023 19:55:27-0300
Verifique em https://validar.iti.gov.br

Presidente, Milena Faria Pinto, D.Sc (orientador)
CEFET-RJ

gov.br Documento assinado digitalmente
JOAO TERENCIO DIAS
Data: 13/04/2023 17:55:44-0300
Verifique em https://validar.iti.gov.br

João Terêncio Dias, D.Sc. (orientador)
CEFET-RJ

gov.br Documento assinado digitalmente
LUCIANA FALETTI ALMEIDA
Data: 13/04/2023 16:30:04-0300
Verifique em https://validar.iti.gov.br

Luciana Faletti Almeida, D.Sc.
CEFET-RJ

gov.br Documento assinado digitalmente
MARLEY MARIA BERNARDES REBUZZI VELLA
Data: 13/04/2023 09:09:04-0300
Verifique em https://validar.iti.gov.br

Marley Maria Bernardes Rebuzzi Vellasco, D.Sc.
Pontífica Universidade Católica do Rio de Janeiro - PUC - RJ

António Paulo Gomes Mendes Moreira,
D.Sc. Universidade do Porto - FEUP

Assinado por : **ANTÓNIO PAULO GOMES MENDES MOREIRA**
Num. de Identificação: BI05928212

Rio de Janeiro,
Abril, 2023

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

V614 Silva, Yago M.R.

Computer Vision Based Pipeline Tracking for Autonomous Unnamed Aerial Vehicle in Unburied Pipeline Onshore Inspection / Yago Melo Rodrigues da Silva – 2022.
xv, 97f. : il.color. , grafs. , tabs. ; enc.

Dissertação/Tese (Mestrado/Doutorado), Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2022.
Bibliografia: f, 53–59
Orientadores : Milena Faria Pinto
João Terêncio Dias

1. UAV. 2. Aerial Imagery. 3. Engenharia elétrica. I. Pinto, Milena F. II. Dias, João T. III. Pipe Inspection Using Computer Vision Guided Autonomous UAV.
CDD 006.3

AGRADECIMENTOS

Primeiramente agradeço a Deus e a minha família especialmente minha esposa, Caroline Nogueira ,e ao meu pai, André Rodrigues, que me apoiaram para eu chegar onde estou.

À orientadora Milena Faria Pinto agradeço pelo incentivo para ingressar no programa de pós-graduação do CEFET/RJ e perseverar até o final. Obrigado por toda disponibilidade, conhecimento, experiência, paciência, cobranças e incentivos, sem isso seria impossível conquistar esses resultados. Muito obrigado!

Ao orientador João Terêncio agradeço, primeiramente por ter aceitado me orientar, por toda experiência compartilhada, disponibilidade e boa vontade!

Aos meus parceiros de artigo Lucas Sousa e Gabriel Castro, obrigado por todo apoio, paciência e comprometimento, conseguimos resultados incríveis nos artigos.

Ao Fábio Andrade obrigado por ter aceitado participar do artigo que publicamos na revista Drones da MPDI, por ter disponibilizado a verba de pesquisa para publicação e todo comprometimento para submetermos o artigo no prazo.

Parabéns aos professores e servidores do CEFET/RJ pela competência, dedicação e pela educação de excelência proporcionada a todos os alunos!

Por último agradeço à banca examinadora, pelo convite aceito para avaliação deste trabalho.

Yago Silva

"If you want to go fast, go alone. If you want to go far, go together."

African Proverb

RESUMO

Computer Vision Based Pipeline Tracking for Autonomous Unnamed Aerial Vehicle in Unburied Pipeline Onshore Inspection

Yago Melo Rodrigues da Silva

Orientadores:

Prof. Milena Faria Pinto, DSc. e Prof. João Terêncio Dias, Dsc.

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em nome do programa, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Os Sistemas Aéreos Não Tripulados (SANT) estão se tornando cada vez mais atrativos devido à eficiência no desempenho de diversas tarefas. No cenários de inspeção de tubulações de óleo e gás, os SANTs se destacam pela capacidade de cobrir grandes áreas, reduzir a exposição humana ao risco, execução de tempo, e custo. Os Veículos Aéreos Não Tripulados (VANT) para atividades de inspeção podem ser muito atrativos pela facilidade de cobrir grande extensão das instalações industriais e mitigar risco humano, tais como trabalho em altura e áreas classificadas. Alguns autores propuseram soluções para detectar vazamentos de gás em tubulações onshore, mas apenas alguns deles abordaram o problema de trajetória e rastreamento para a navegação autônoma com esses tipos de veículos. A maioria desses autores propõe soluções com visão computacional tradicional para rastrear as tubulações. Portanto, este trabalho de pesquisa descreve o potencial de um VANT e uma estratégia deep learning para navegar adequadamente em condições complexas de dutos não enterrados, contribuindo para o procedimento de monitoramento e mapeamento das estruturas da Indústria de Petróleo e Gás. Os resultados sugeriram que a estratégia proposta tem um excelente desempenho. Além disso, apresenta viabilidade técnica para implantação prática. Os resultados mostraram que o sistema proposto é capaz de identificar corretamente os dutos e navegar sobre eles com segurança e precisão.

Palavras Chaves: Industria Offshore, Missão Autônoma, Sistemas Inteligentes, Veículo Aéreo não Tripulado, Sistema de Navegação.

ABSTRACT

Computer Vision Based Pipeline Tracking for Autonomous Unnamed Aerial Vehicle in Unburied Pipeline Onshore Inspection

Yago Melo Rodrigues da Silva

Advisors:

Milena Faria Pinto, D.Sc. and João Terêncio Dias, D.Sc.

Abstract of Master Dissertation submitted to Electrical Engineering Graduate Program, Federal Center of Tecnologic Education Celso Suckow da Fonseca, CEFET/RJ, as partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

Unmanned Aerial System (UAS) are becoming more attractive in diverse applications due to their efficiency in task performance by covering large areas, reducing human hazard exposure, timely execution, and/or cost. In the Oil & Gas scope, use Unmaned Aerial Vehicle (UAV)s for inspection activities is an even more attractive scenario because of the industrial facilities' large extension and human risks, such as work at height and classified areas. Different authors proposed solutions to detect gas leaking in unburied onshore pipelines. Still, just a few of them addressed the path and tracking problem for autonomous navigation with these vehicle types. Most of these authors proposed solutions with traditional computer vision strategies to track the tubes. Therefore, this research work describes the potential of an autonomous UAV with a deep learning strategy to navigate appropriately into the complex scenarios of unburied pipeline networks, contributing to the monitoring and mapping procedure of the Oil & Gas Industry structures. The results suggested that the proposed strategy has an excellent performance and it presents a technical viability for practical deployment. In relation to tracking and detection strategies, the outcomes showed that the proposed system is capable of correctly identifying pipelines and navigating above them safely.

Key Words: Autonomous Mission, Intelligent System, Unmanned Aerial Vehicle, Navigation System, Automatic Inspection, Pipe Inspection; Computer Vision

Contents

I Introduction **14**

I.1 Initial Considerations and Work Motivation 14

I.2 Main Contributions 15

I.3 Related Publication 15

I.4 Organization 16

II Background and Related Works **17**

II.1 Related Works 17

II.1.1 Autonomous Inspection in the Oil & Gas Industry 17

II.1.2 Vision-Based Techniques Applied in Inspection Process 18

II.2 Robotic Operating Systems 19

II.3 Computer Vision 20

II.4 Quadrotor Model 22

II.5 Convolutional Neural Network 24

III Proposed Methodology **27**

III.1 Computing Vision and Dataset Creation 30

III.2 YOLO Training 32

III.3 Pipeline Tracking Controller 32

IV Results and Discussion **39**

IV.1 Simulation Environment 39

IV.2 YOLO Training and Results 41

IV.3 Pipeline Follower Simulation Results 44

IV.4 Pipeline Tracking Simulation Results 45

IV.5 Real Flight Results 48

V Conclusions and Future Work **51**

V.1 Conclusions 51

V.2 Future Works 52

Bibliographic References 53

List of Tables

II.1 Related works summary. 19

III.1 Hyperparameters used in YOLO's training step. Table from (1). 32

IV.1 YOLO training results 42

IV.2 Model confusion matrix (1). 43

IV.3 Comparison result between simulation and real environments. Table from (1). 48

List of Figures

II.1 ROS node and topic overall. . . . 20
II.2 Hough points. . . . 21
II.3 (a) Input image. Hough transform comparison. (b) Standard. (c) Probabilistic. Accessed in 03/2023, image from https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html 22
II.4 Variables of the UAV movement. Image from (1). . . . 23
II.5 CNN/YOLO architecture. Input image(a).Convolutional layers (b). Pooling (c). Flattening (d). Output (e). . . . 25
II.6 YOLO object detection process applied in the proposed scenario. (a) Pipeline. (b) Generated boxes. (c) Merge boxes that the pipe was detected. (d) Generate bounding box. Image adapted from (1) . . . 26

III.1 General idea of the proposed solution. . . . 27
III.2 General systems interfaces. . . . 28
III.3 Project proposal. (a) Part 1. (b) Part 2. . . . 29
III.4 Object detection to make the turn. . . . 30
III.5 General image interface. . . . 31
III.6 Pipe follower flow data. (a) pipe CNN detection and approach. (b) Canny edge detection. (c) Hough transform. . . . 33
III.7 Angle Correction Illustration. (a) no correction. (b) needs correction . . . 34
III.8 Line follower outputs. (a) Raw image. (b) Canny filter without Region of Interesting (ROI) on the left and with ROI on the right. (c) Hough image with the reference of the life follower. . . . 35
III.9 Navigation Representation. (a) Pipe detection with the reference line. (b) Binarized image. (c) Canny kernel. (d) Hough transform. Part (e) Simulated environment. Image adapted from (1) . . . 36
III.10 ROI for You Only Look Once (YOLO) usage. (a) Raw image. (b) YOLO running with ROI set up. . . . 37
III.11 Navigation logic diagram. . . . 38

IV.1 Line Follower World. Image made by Lucas Silva Sousa and Yago Melo. . . . 40

IV.2 Gazebo Worlds. (a) Heights Valley with big tubes, image from (2). (b) A simple world with a green tube. 40
IV.3 UAV models. (a) Typhoon 480. (b) Iris with the camera. 41
IV.4 Box and object loss through the 280 epochs of training. (a) Bounding box regression loss (Mean Squared Error) X Epochs. (b) Confidence of object X Epochs. Image from (1) 43
IV.5 Pipe class Precision x Recall. Image from (1). 44
IV.6 Unburied pipeline example used for our dataset. 45
IV.7 UAV performing the line detection. 45
IV.8 UAV performing the pipeline track. Image from (1). 46
IV.9 UAV performing the pipeline tracking in the simulation environment. 47
IV.10 UAV performing the pipe tracking in real test. Image adapted from (1). 49
IV.11 UAV positioning during real tests. Image from (1). 50

Abbreviation

ANN Artificial Neural Network 24, 25, 52
AP Average Precision 25
AUV Autonomous Underwater Vehicle 17
CED Canny Edge Detector 18
CNN Convolutional Neural Network 24, 25, 27, 28, 31, 34, 39, 42, 45, 51, 52
CPU Computer Processing Unit 25, 26, 39
FCU Flight Control Unit 27, 31, 34, 39, 52
GPS Global Positioning System 14
GPU Graphical Processing Unit 25, 26, 51
IOU Intersect Over Union 26
MAP Mean Average Precision 26, 41, 42
PHT Probabilistic Hough Transformation 18
PID Proportional-Integral-Derivative 34, 52
PIG Pipeline Inspection Gauge 17, 51
ROI Region Of Interesting . . . . . xi, 35, 36, 37
ROS Robot Operating System 15, 19, 27, 30, 31, 39, 51
ROVS Remotely Operated Vehicles 14
SITL Software In The Loop 15, 27
TPU Tensor Processing Unit 42
UAS Unmanned Aerial System . . . . . vii, 36, 37
UAV Unmaned Aerial Vehicle . . vii, 14, 15, 16, 17, 18, 22, 23, 27, 28, 30, 31, 32, 34, 36, 37, 39, 40, 41, 44, 46, 47, 48, 51, 52
UDP User Datagram Protocol 27, 31, 51
VOC Visual Object Classes 26
YOLO You Only Look Oncexi, 16, 25, 26, 27, 28, 30, 31, 32, 36, 37, 39, 41, 42, 43, 45, 46, 48, 51, 52

Chapter I Introduction

I.1 Initial Considerations and Work Motivation

UAV are becoming more popular in diverse applications, being applied in security areas (3, 4), small deliveries (5, 6), military combat, photogrammetric applications (7), remote inspections (8, 9), agriculture, among others. Basically, the application of this resource intends to increase the efficiency of the task by reducing human hazard exposure, timely execution, and/or cost (10).

The UAV controlling process can be performed remotely (e.g., radio-controlled, via satellite) or autonomously. Typically, UAVs use Global Positioning System (GPS)-based navigation. However, due to the constant changing in the environments, as well as the unmapped obstacles, by using only the GPS, the trajectory performed by the UAV can be inaccurate and inefficient (11). Thus, an intelligent strategy for autonomous movement uses image processing along with other sensors (12). Computer vision plays an important role in increasing the autonomous functionalities of robotic systems (13). By using cameras and image processing, it is possible to interpret situations/scenes and detect objects (14). Therefore, these are one of the main tools for controlling autonomous vehicles, whether performing a task or navigating environments.

In the literature, several researchers applied computer vision techniques to increase the automaticity level of robotic systems (15, 16, 17). Khaloo et al. (18) presented a study with the application of UAVs for inspecting dams looking for breakpoints in these structures. Another application of autonomous inspection with UAVs is presented in Vam et al. (19). The authors proposed an autonomous inspection of an electrical power station, in which the UAV must be able to navigate the installation autonomously, carry out video inspections, and indicate the location that contains signs of degradation, serving as a basis for infrastructure inspection work as a whole.

In large industrial parks, refineries, and Oil & Gas processing units, the scenario is even more attractive for applying UAVs for inspection activities (20). As the dimension and extensions of these facilities are very large and with different risks among the areas, the operator could remain stationary in a safe area while monitoring the flight parameters during the inspection process with the UAV (21). Note that the Oil & Gas industry has used Remotely Operated Vehicles (ROVs) since the 70's decade to inspect and maintain underwater structures (22). The work of Kus and Srinivasan (20) applied UAV for visual inspection of external corrosion in the refinery unit. In (17) and (23), the authors presented

the use and technical feasibility of the autonomous vision-based UAV application for messenger cable transfer in mooring tasks. Another UAV application in Oil & Gas environment is presented (24). In their work, they applied a multirotor UAV for small cargo transport between naval structures.

As shown, different authors proposed the usage of UAV in the Oil & Gas industry, but just a few of them are applied to detect damage in unburied onshore pipelines. Another small part of the research addressed the path and tracking problem for autonomous navigation with these vehicle types. Most of these authors proposed solutions with traditional computer vision strategies to track the tubes. Thus, this work is relevant because it describes the potential of an autonomous UAV with a deep learning strategy to navigate appropriately into complex scenarios of unburied pipeline networks, contributing to the monitoring and mapping procedure of the Oil & Gas Industry structures.

I.2 Main Contributions

As can be seen, novel robotic systems have been applied to the O&G industry sector to move the traditional processes towards more autonomous operations (25). As a result of this process, operational costs can be reduced and human safety can be improved (26). Therefore, this study presents a tracking strategy for an autonomous UAV guided through computer vision control development. In the context of unburied pipelines, gas inspection in the Oil&Gas industry has difficult areas to access and with significant human risk, classified areas, and work at heights, among others. Considering this solution, the work aims to test the feasibility of using UAVs to navigate the unburied pipelines in the O&G facilities. The main contributions of this research can be summarized as follows:

- Development of a simple tracking strategy, based on image processing and neural networks, to navigate the UAV over the extensive structures of unburied pipelines in the O&G industry;
- Testing the proposed path based on computer vision in the Iris/Typhoon 480 UAV on PX4 Software in the Loop (SITL) and Gazebo software with Robot Operating System (ROS) to prove its functionality in a real application.

I.3 Related Publication

Until now, this research work has published a dataset in the repository IEEE DataPort, a paper at the seventh International Conference on Robotics and Automation Engineering (ICRAE), and an article at the Drones Journal (MDPI). A summary is presented below:

- **Created Dataset:**

  Yago Silva, Lucas C. Sousa, Caio Souza, Gabriel Castro, June 30, 2022, "Pipeline Recognition for Drone Navigation", IEEE DataPort, doi: https://dx.doi.org/10.21227/pncz-sv78.

- **Conference paper:**

  Lucas C. Sousa, Yago Silva, Gabriel Castro, João Dias, Guido Berger, José Lima, Milena F. Pinto.*Autonomous Path Follow UAV to Assist Onshore Pipe Inspection Tasks*. In: 7th International Conference on Robotics and Automation Engineering, 2022, Cingapura. 7th International Conference on Robotics and Automation Engineering, 2022.

- **Journal article:**

  Yago Silva, Fabio Andrade, Lucas C. Sousa, Gabriel Castro, João Dias, Guido Berger, G.; José Lima, Milena F. Pinto. *Computer Vision Based Path Following for Autonomous Unmanned Aerial Systems in Unburied Pipeline Onshore Inspection*. Drones, v. 6, p. 410, 2022.

I.4 Organization

To achieve the objective previously stated, this dissertation is divided as follows. Chapter II resumes the necessary background for onshore pipeline inspection in the Oil & Gas industry.

Chapter III shows an overview about image processing, neural networks (YOLO v5), path planning based on vision strategy, and a basic overview of the quadcopter UAV dynamic and the control response based on vision.

Chapter IV presents and evaluates most of the results obtained during the methodology, such as neural network (YOLO version 5) evaluation, simple line follower performance, and pipeline tracking for inspection.

Chapter V reveals the final results and achievements that involve the main objective and also provides possibilities for continuing this work.

Chapter II Background and Related Works

This chapter presents the background foundations needed to support the understanding of the proposed thesis.

II.1 Related Works

II.1.1 Autonomous Inspection in the Oil & Gas Industry

UAVs are classified into different roles, such as flying height and range, size and weight. The main three types are *(i)* those for military usage, which are generally shaped like airplanes; *(ii)* filming drones, which are multirotor they have two or more propellers to fly, the quadcopters (four propeller helicopters) is the most commonly used; *(iii)* the toy ones that are similar to the filming drones, but have smaller reach and low speed. Thus, a multirotor is perfect for activities where flight stability is the focus, with the possibility of stopping in the air, precisely controlling the speed of movement and performing sharp turns, among other performances (27).

UAVs are being used in the most different missions that can be risky for humans or in tedious tasks that demand an outstanding level of attention (28). When the UAV is deployed for inspection in large structures, the inspection process can be speeded up, and cost-efficient, due to the reduction of engineering labor fees and risk minimization due to human errors (26). For instance, the pipeline corridors inspection is still monitored by foot patrol, crewed vehicles, and air patrols with inspection personnel, which opens space for researching autonomous inspection themes.

Several works proposed solutions to inspect pipeline corridors. For instance, Wang et al. (29) used an Autonomous Underwater Vehicle (AUV) equipped with image-processing technology to detect the pipeline's corners. Concerning onshore pipelines, there are different types of robots, and the locomotion style reflects directly in the robot's performance for the inner pipeline inspection. A simple device called Pipeline Inspection Gauge (PIG) was used to collect the pipeline data proposed by Mazreah et al. (30). Their robot was driven by oil or gas flow through the pipeline to perform the inspection. A wheel-type robot was proposed by Kakogawa et al. (31). The wheel of their robot was used to touch the pipe wall. In Kwon et al. (32), the authors used a track type as an alternative solution to the wheeled robot.

Regarding the use of UAVs for pipeline corridor inspection, many authors proposed solutions to detect a gas leak on onshore pipelines (33, 34). For instance, in Bretschneider et al. (35), the authors

described a payload solution to detect gas leaks in unburied onshore pipelines. Their solution presented a combination of lasers and an electro-optical sensor. However, a few works described solutions to the autonomous navigation solution to support pipeline corridors inspection and monitoring in the Oil & Gas industry.

Shukla et al. (36) developed an autonomous tracking and navigation controller for UAVs to inspect straight oil and gas pipelines. Their work can not handle curves in the pipeline corridors as a drawback. In (37), the authors showed an optimization strategy for UAVs inspection of an oil and gas pipeline network. Their process used a mixed-integer nonlinear programming model to consider the constraints of the mission scenario and the safety performance.

As can be seen, there is still space for improvements in inspection and monitoring solutions of unburied onshore pipeline corridors, especially regarding the autonomous pipeline tracking solution with UAVs.

II.1.2 Vision-Based Techniques Applied in Inspection Process

Computer vision is one of the fastest-growing areas in the computing field. The interpretation and transformation of data into useful information are complex tasks. In the pipeline inspection area, image processing is often used to identify and detect parts inside the pipe (38). There are plenty of methods for corrosion and cracks region segmenting, such as morphological operations (39, 40, 41), geometrical operations (42, 43), segmentation methods (44, 45), Thresholding (46) and active contour (47).

For instance, morphological operations, thresholding and labeling were used for corrosion inspection in (39). Similarly, in the work of (41), the authors applied morphological-based edge detection in order to detect cracks in the pipeline. Recently, to overcome the effort of traditional image processing, deep learning became a prominent solution for the field of computer vision (48, 49).

However, to the best of the authors' knowledge, no work in the literature applied deep learning strategies as a tool to the navigation control problem in the unburied pipeline network. In the Xiaqian et al. (50) work, the authors proposed an autonomous navigation control strategy for tracking and inspecting linear horizontal structures of pipeline networks with UAVs. They used traditional methods, such as Canny Edge Detector (CED) and the Probabilistic Hough Transformation (PHT) to identify the structures of the control input. They did not consider more complex scenarios with the pipeline networks and only considered the control problem.

Gomez et al. (34) performed an extended review of the requirements for small unmanned airborne systems applied to pipeline inspections and monitoring. Their paper concluded that UAVs are reliable and can substitute traditional monitoring procedures such as foot patrolling and aerial surveys. Therefore, this research describes the potential of an autonomous UAV along with a deep learning strategy to navigate appropriately to complex unburied pipeline networks, contributing to the monitoring and map-

ping procedure of these structures of the Oil & Gas Industry. The related works summary is presented in Table II.1.

Table II.1: Related works summary.

Work	Robot	Path Planning			O & G
		Segment.	IR sensor	Identif. of curv. and pipe	
Wang et al.(29)	AUV	Yes	No	No	Yes
Mazreah et al.(30)	Pipeline Inner Robot	No	No	No	Yes
Kakogawa et al.(31)	Wheeled Robot	No	No	No	Yes
Basso et al.(51)	UAV	Yes	No	No	No
Okoli et al.(52)	Wheeled Robot	No	Yes	No	Yes
Santa et al. (53)	UAV	Yes	No	No	No
Proposed System	UAV	Yes	No	Yes	Yes

II.2 Robotic Operating Systems

ROS is a set of libraries and tools contained in an Open-Source Software that allows communication of various controller hardware and software in a simplified way, achieving compatibility without the use of a low-level programming language. The Operating System, developed by Stanford University, consists in a collection of several software frameworks that enables the creation of modular applications and services, called nodes, which act in a varied and flexible way in dynamic systems. In addition to managing packages, ROS acts mainly as an operating system capable of performing hardware system abstraction. In short, as stated in (54), the system adapts tools and software to simplify the task of creating complex and robust robotic behavior on various robotic platforms. Traditionally ROS Workflow can be broken down into processes that describe nodes, topics, and services:

- **Nodes:** are a simple process that performs computation. They can be combined together into a graph and communicate with other nodes using topics, services, etc. Example: there is a node to control a laser range-finder, one Node controls the robot's wheel motors and another node performs localization, among others (55);
- **Topics:** are named buses over which nodes exchange messages. According to the definition of (55), nodes do not know who they communicate with. Therefore, nodes that are interested in data subscribe to the relevant topic; nodes that generate data publish to the relevant topic.
- **Services:** Both request and reply are done by using a Service. Service is defined by a pair of messages, i.e., one for the request and another one for the reply (55).

Figure II.1 identifies the communication processes of the ROS structure.

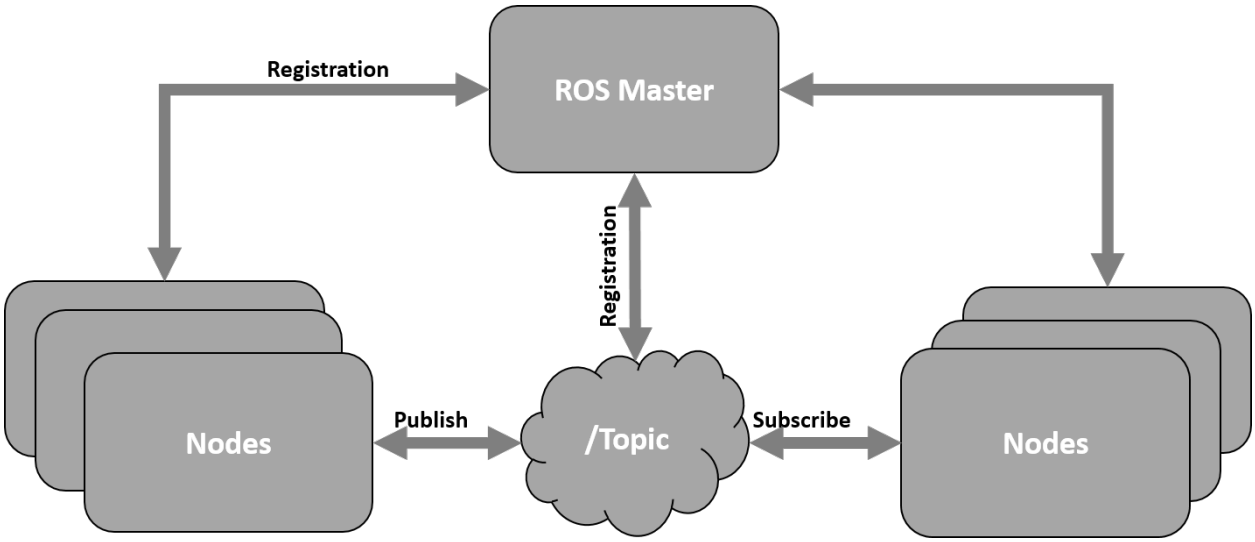


Figure II.1: ROS node and topic overall.

II.3 Computer Vision

For computer vision usage one of the biggest and most important libraries available is Open Computer Vision, OpenCV. This open-source library written in C, with many applications using Python code language, offers different functionalities and filters for image processing which can be used for simple applications to real-time image recognition. Some of the features embedded with this library are threshold, convert color scale (gray, RGB, BGR, etc.) and blur filter, among others (56) (57).

In 1986, John Canny published an article that propose some filters for image processing capabilities to find contour patterns in an image, called Canny Edge detection. It is an optimal detector with low error rate and every edge is detected only once (58).

The method uses basically 4 steps to perform edge detection with high accuracy.

- **Gaussian Blur:** once noise can be easily confused as an edge, the blur filter reduces noise using kernel size 5 - 5x5 matrix;
- **Pixel Gradient:** finds the gradient intensity (the directional change in the intensity of/or the color) of the image, applying Sobel kernel, this is important because an edge has a specific pixel gradient pattern;
- **Non-Maximum Suppression:** removes pixels that are not considered part of an edge, only the edges with higher probability will remains;
- **Hysteresis - Upper and Lower Threshold:** this kernel has important points:
  - it is an edge if pixel gradients are higher than the upper threshold;
  - it is not an edge if pixel gradient values below the lower threshold;

– for pixel gradient is between the upper and lower threshold only it is an edge if the gradient is connected to a pixel above the upper threshold.

Another technique is the Hough Transform. This is a technique created to optimize line detection for high-scale implementation.

Hough transform is a popular feature extraction technique in which the main objective is to convert the image points into a line, a simple equation of a straight line (II.1) and a polar equation (II.2). With these two equations, it is possible to transform the curve or straight in points, Figure II.2 illustrates mathematically, and build an accumulator matrix $A(N)$. So, this feature is based on mapping the digital image pixel into the space parameter. An N-dimensional accumulator to pixel organization will be quantized and added into an accumulator matrix A(N). Inside this matrix, the highest peaks detected are the higher potential equation of a straight-line detection, working like a voting system.

$$Y = aX + b \tag{II.1}$$

$$\rho = X \cdot cos(\theta) + Y \cdot sin(\theta) \tag{II.2}$$

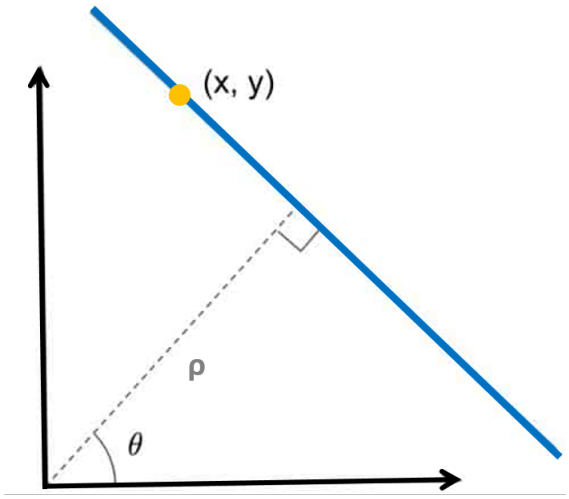
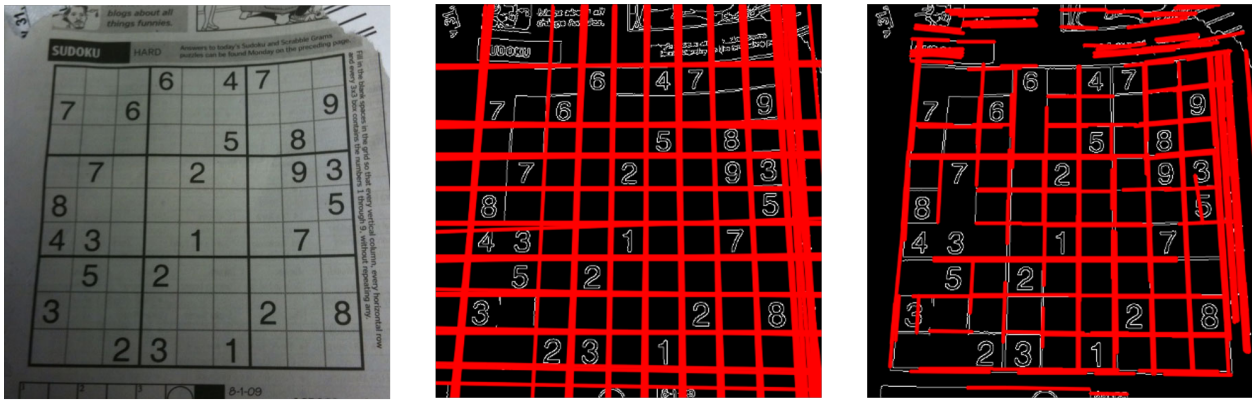


Figure II.2: Hough points.

The standard Hough Transform trace a straight line through two points, not between theses points, but it is not necessary true that the line traced is that big. With this mind the probabilistic Hough Transform version looks more attractive to real life application, Figure II.3 illustrates an example.

The probabilistic Hough transform analyzes a points subset N-dimensional accumulator and estimates the probability that all these points belong to the same line. Note that a subset of N is smaller than the whole N. Consequently, the accumulator matrix A(N) is smaller and will process faster and more accurately. The output of the probabilistic Hough transform contains just the points with a high chance

of composing the line. Thus, the result is one line connecting two points (59). Figure II.3 shows the standard Hough transform (b) and the probabilistic Hough transform output (c).



(a) (b) (c)

Figure II.3: (a) Input image. Hough transform comparison. (b) Standard. (c) Probabilistic. Accessed in 03/2023, image from https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html

Basically, with Canny Edge Detection and probabilistic Hough transform, it will be possible to find a clear line to be followed, as shown in some works to identify street car lanes applied to autonomous car path system (60) (61) (62).

II.4 Quadrotor Model

It is important to identify the mechanics required to perform the flight. The quadrotor UAV model is based on (63). The motor speed variation results in the UAV rotation on its axis. Due to the generated inclination, it results in vehicle displacement in any direction of space. To understand how the variation of each rotor affects the vehicle's speed control, it is necessary to define the coordinates specified by the equations ${}^{e}\mathbf{q} = [{}^{e}\xi \;\; {}^{e}\eta]^T$, where ${}^{e}\xi = [x \;\; y \;\; z]^T \in \Re^3$ corresponds to the longitudinal, lateral and normal displacement in the structure $\langle e \rangle$. The variable ${}^{e}\eta = [\phi \;\; \theta \;\; \psi]^T \in \Re^3$ corresponds to the angles of *roll*, *pitch* and *yaw* referenced in the space $\langle s \rangle$.

The motors are coupled to form a 45-degree angle between the x-y coordinates for the quadrotor UAV used in this project. Thus, to perform any longitudinal movement, all the motors must act. The vector of forces referring to the UAV that causes the acceleration generated to different axes is represented in Equation (II.3).

$$F = \begin{bmatrix} fx \\ fy \\ fz \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sum_{n=1}^{4} f_n \end{bmatrix} \tag{II.3}$$

The longitudinal forces cancel each other out, causing the vehicle to reach equilibrium in the x-y plane. The force on the z-axis results from the sum of the forces exerted by the four engines, calculated

from Equation (II.4).

$$f_n = C_f \omega f^2 \tag{II.4}$$

where the variable C_f is a constant that depends on parametric constants associated with the propellers and motor coupling region, and the variable ω represents the angular velocity generated by the rotation. The speed is directly controlled by the torques generated by the engines and applied to the vehicle body. Equation (II.5) represents the torque vector.

$$\tau = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} k1 & k1 & -k1 & -k1 \\ -k1 & k1 & k1 & -k1 \\ k2 & -k2 & k2 & -k2 \end{bmatrix} \begin{bmatrix} f1 \\ f2 \\ f3 \\ f4 \end{bmatrix} \tag{II.5}$$

where $k1$ is the distance between the reference axes and the point where each force is applied (i.e., motor positioning center). $k2$ is the determinant of the relationship between the torque generated by the rotor and its corresponding impulse. Figure II.4 shows the variables that compose the UAV movement.

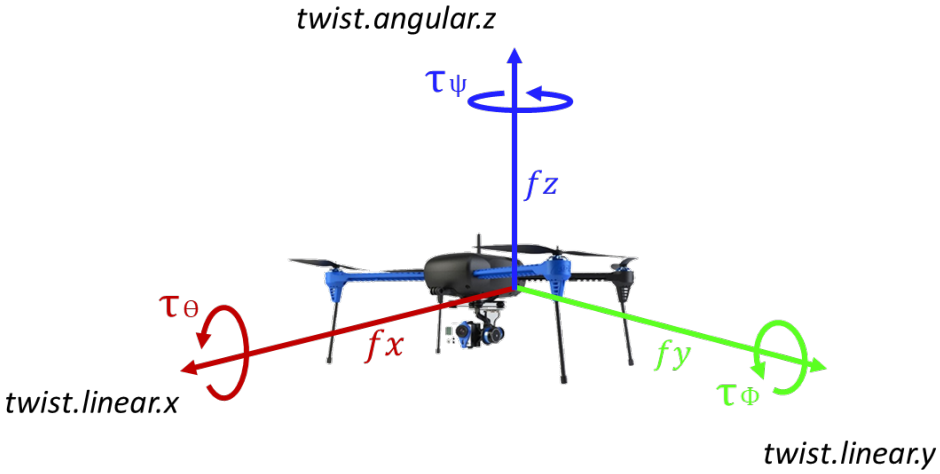


Figure II.4: Variables of the UAV movement. Image from (1).

The quadrotor is used in most parts of this research. Besides that, during the middle-final research, it has been used the hexacopter model in the simulation. The hexacopter has a mathematical model quite similar to the quadrotor, shown in (64). Basically, the hexacopter has six motors, 120 degrees frame-mounted, and it can carry more weight. This robot has a redundancy motor. Thus, to move the hexacopter, the idea is to unbalance the drone in an angular axis just like the quadrotor. To move forward (x-axis), it is used the pitch movement (twist.linear.y), to displace the UAV to left and right, it is used the roll axis (twist.linear.x), and to make the hexacopter move in its z-axis, it is used the yaw axis (twist.angular.z).

II.5 Convolutional Neural Network

It is well known that Convolutional neural network (CNN) is the state of art for object detection task (65). CNN is one of Artificial Neural Network (ANN) classes commonly used to produce an accurate visual image processing widely applied to object detection (66). Starting with LeNet in 1998 (67) since then recognition capability has been highly improved with the emergence of deep learning architectures, such as AlexNet (68), GoogleLeNet (69) and others.

CNNs are based on biological processes, similar to ANNs, where the connectivity pattern between neurons imitates the animal visual cortex organization (70, 71). The basic idea of CNN, after the image input, is resumed on the Convolution Layer, Pooling Layer, and Flattening and then apply to a traditional ANN.

1. **Input:** it is the data input that could be anything. In this thesis, the image is used as input, Figure II.5 (a).

2. **Convolution:** it is the CNN main building block, which contains a set of kernels (previous filters such as edge detection, sharpen, blur, among others) whose parameters need to be learned. Each filter is convoluted with the input image to generate an activation function. The height and weight of the filters must be smaller than the input volume, so each neuron in the activation map (composed of any features matrix) is connected to a small location/region of the input image (72). In resume, this phase reduces the image extracting only the most important attributes from the input and creating a feature set, Figure II.5 (b).

3. **Pooling:** it is used to reduce the feature maps dimensions combining each output of the previous layer group into a single neuron. There are two common pooling operations, average pooling (takes the mean value matrix feature part) and max pooling (takes the biggest value matrix feature part)(73). This resource reduces the overfitting generated during the convolution, making the CNN faster and easier to be executed, Figure II.5 (c).

4. **Flattening:** it is used to transform the matrix into a vector form and generate the neural network, following the traditional ANN (73), , Figure II.5 (d).

5. **Output:** it is the network weights (this file is usually based on iterations numbers) used with the configuration file, Figure II.5 (e).

Figure II.5 illustrates these five processes. As stated in (74), the CNNs can transform an original input layer by layer using convolutional and downsampling techniques to produce class scores for classification and regression purposes.

You Only Look Once - YOLO

It's clear that CNN is one of ANN classes commonly used to produce an accurate image processing, widely applied to object detection. YOLO algorithm makes the process easier and faster according (75), because deals with small problems dividing the image in region (76).

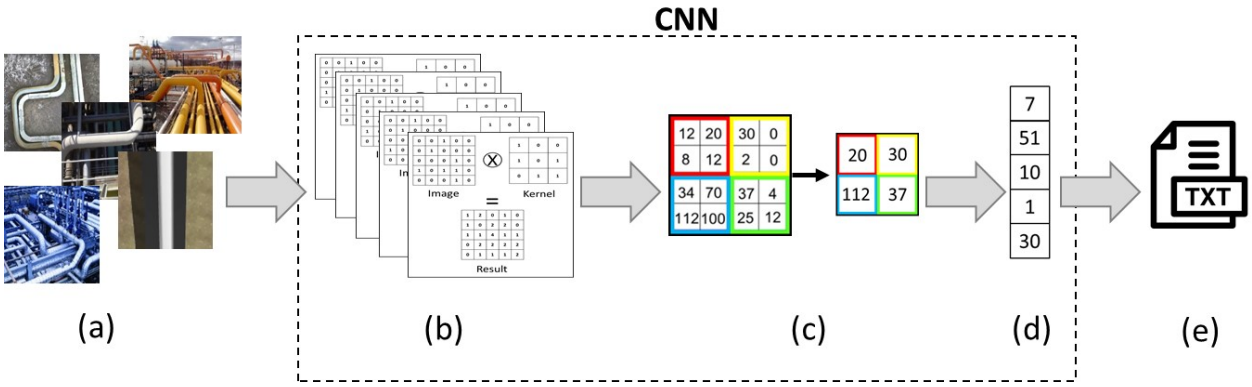


Figure II.5: CNN/YOLO architecture. Input image(a).Convolutional layers (b). Pooling (c). Flattening (d). Output (e).

According to (76), YOLO is a state-of-the-art, real-time object detection, it is extremely fast and accurate. Figure II.5 illustrates the basic architecture of the YOLO, which comprises 24 convolutional layers, followed by two fully connected layers.

YOLO is one of the best neural networks for object detection in cost-benefit terms regarding accuracy and speed, with 55.5% Average Precision (AP) for the MS COCO dataset. Following this idea, it is not hard to find YOLO to perform object detection, for instance, to improve autonomous driving by avoiding pedestrians (77) and to help the automated driving to keep on track (77). A similar idea can be used in this work, finding a tube to ensure that the drone it is on track.

This network is a single-stage detector, which divides the image into regions, predictions bounding boxes (if the image contains the object of this detection, one or some of them are the ground truth), and probabilities. This process is illustrated in Figure II.6. This procedure runs into a pre-trained CNN called Darknet, an open-source neural network written in C and CUDA, which supports Computer Processing Unit (CPU) and Graphical Processing Unit (GPU) computation (76) (78).

YOLO is in continuous development. Until the writing of this dissertation, the latest YOLO is in version 8, developed by Ultralytics. This work used YOLOv4 in the beginning, which uses the Darknet framework as the first YOLO versions. Trying to improve the ANN performance, it was used YOLOv5 because it is much easier to use. YOLOv5 is not much different from version 4 and there are no expressive performance improvements in object detection. One of the biggest differences is that the framework with version 5 is PyTorch, making this neural network lighter than the old version (79). The version 5 it is also developed by Ultralytics and there is no official article published until this work's

moment (80).

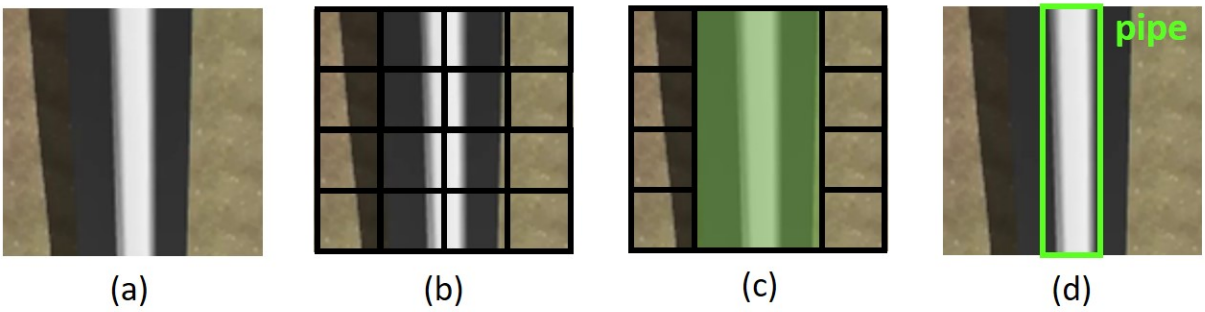


Figure II.6: YOLO object detection process applied in the proposed scenario. (a) Pipeline. (b) Generated boxes. (c) Merge boxes that the pipe was detected. (d) Generate bounding box. Image adapted from (1)

Note that the convolutional layers of the network, Figure II.5, are responsible for extracting the image features. The fully connected layers give the prediction of the output probabilities and coordinates. Interestingly, the YOLO is based on GoogleLeNET(81). However, YOLO does not use the inception modules. It uses $1X1$ reduction layers followed by $3X3$ convolutional layers.

Once the model is trained, it is possible to use the weight and configuration file to identify the desired object using CPU or GPU.

Once the YOLO is already trained, it is necessary to evaluate the network efficiency. In the work of Padilha et al. (82), the authors presented a detailed explanation of the methods to evaluate the YOLO performance based on mean Average Precision (mAP), Visual Object Classes (VOC) Pascal Metric, and by plotting a curve Precision x Recall. A summary of these metrics is given below:

- **Precision:** It is the percentage of positive cases that were correctly identified. This can be used in cases where false positives are considered more harmful than false negatives.
- **Recall**: It is the total number of positives that were correctly classified divided by the total number of elements that are actually positive. This can be used in a situation where false negatives are more harmful than false positives.
- **PASCAL VOC:** It is a popular dataset for object detection. A prediction for the PASCAL VOC challenge is positive if Intersect over Union (IoU) ≥ 0.5. Also, if multiple detections of the same object are detected, count the first one as a positive while the rest as negatives.
- **mAP:** It is the average of average precision (AP). AP is a popular metric for measuring object detectors' accuracy, such as Fast R-CNN, YOLO, and Mask R-CNN.

Chapter III Proposed Methodology

This research work intends to provide an autonomous approach to tracking unburied pipelines in the O&G industry facilities. Figure III.1 illustrates the general idea of the proposed solution. Basically, the UAV uses the CNN to approach the unburied pipeline (Figure III.1 (a)). Then, the follower path strategy is used to track the pipeline as a line follower (Figure III.1 (b)) as long as the CNN gives the feedback of a pipeline. In case the system encounters a curve, the CNN is used to guarantee that the UAV guidance system (Figure III.1 (c)) is able to turn. The whole algorithm is developed in Python, using existent libraries and features. The heart of the project is the CNN YOLOv5 trained to identify a pipe, which ensures that the tracking strategy will work with low error. Contributing to the scientific community with a good solution to autonomous vehicles in the inspection field.

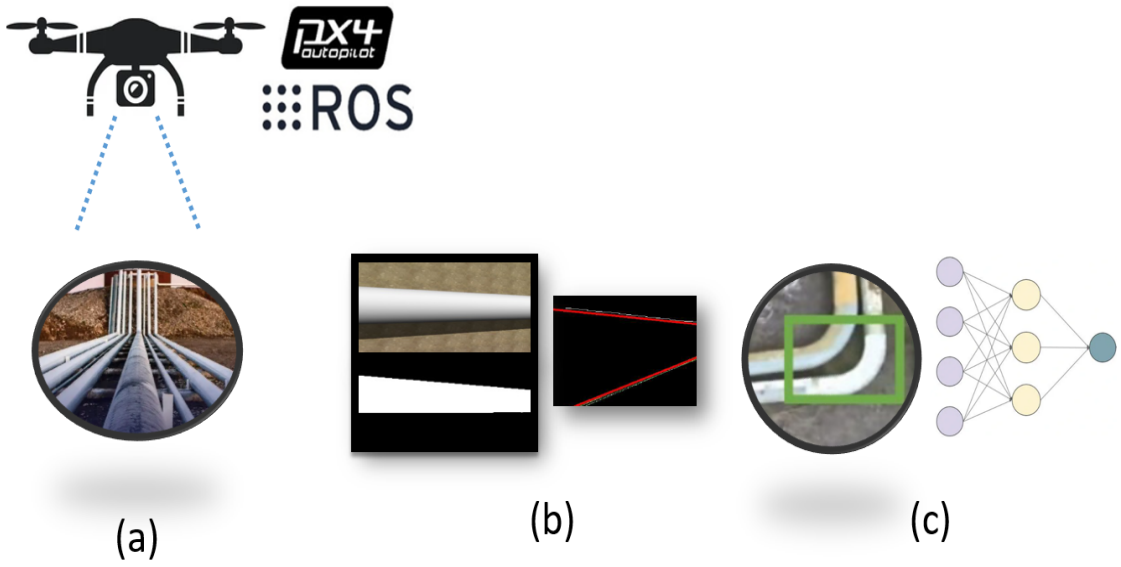


Figure III.1: General idea of the proposed solution.

To achieve the main objective of tracking an unburied pipeline, the hardware and algorithms are initialized along with ROS and the communication with the Flight Control Unit (FCU), PX4. The UAV receives the camera image from an User Datagram Protocol (UDP) video streaming from the PX4 (in this case from PX4 SITL Gazebo through GStreamer data pipeline), processes this information in a Python script, and sends position setpoints to PX4 with MAVros package via MAVlink protocol. Figure III.2 illustrates the interfaces between each system.

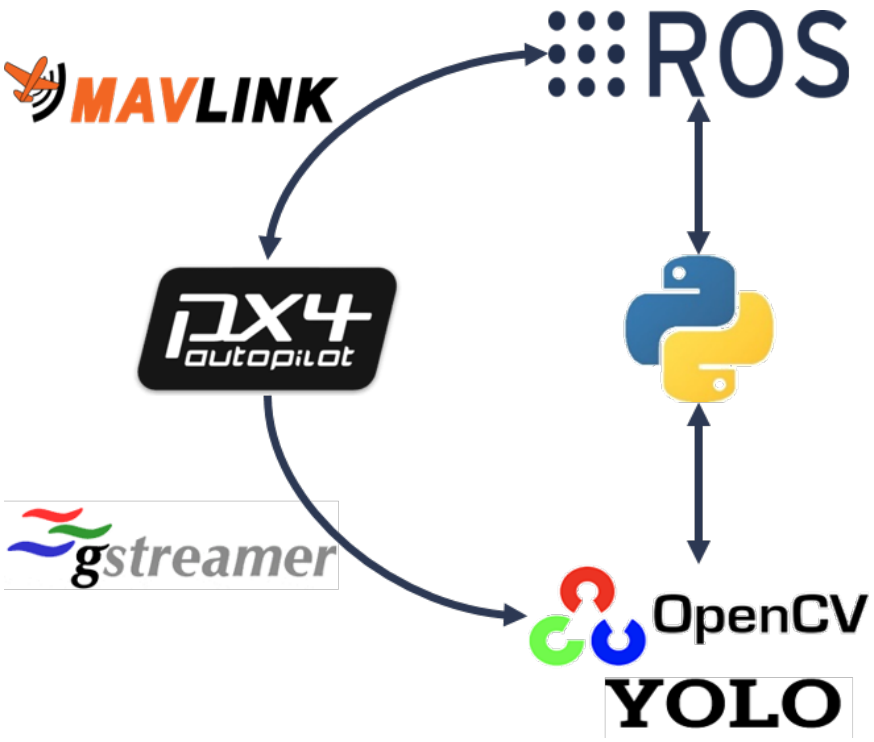


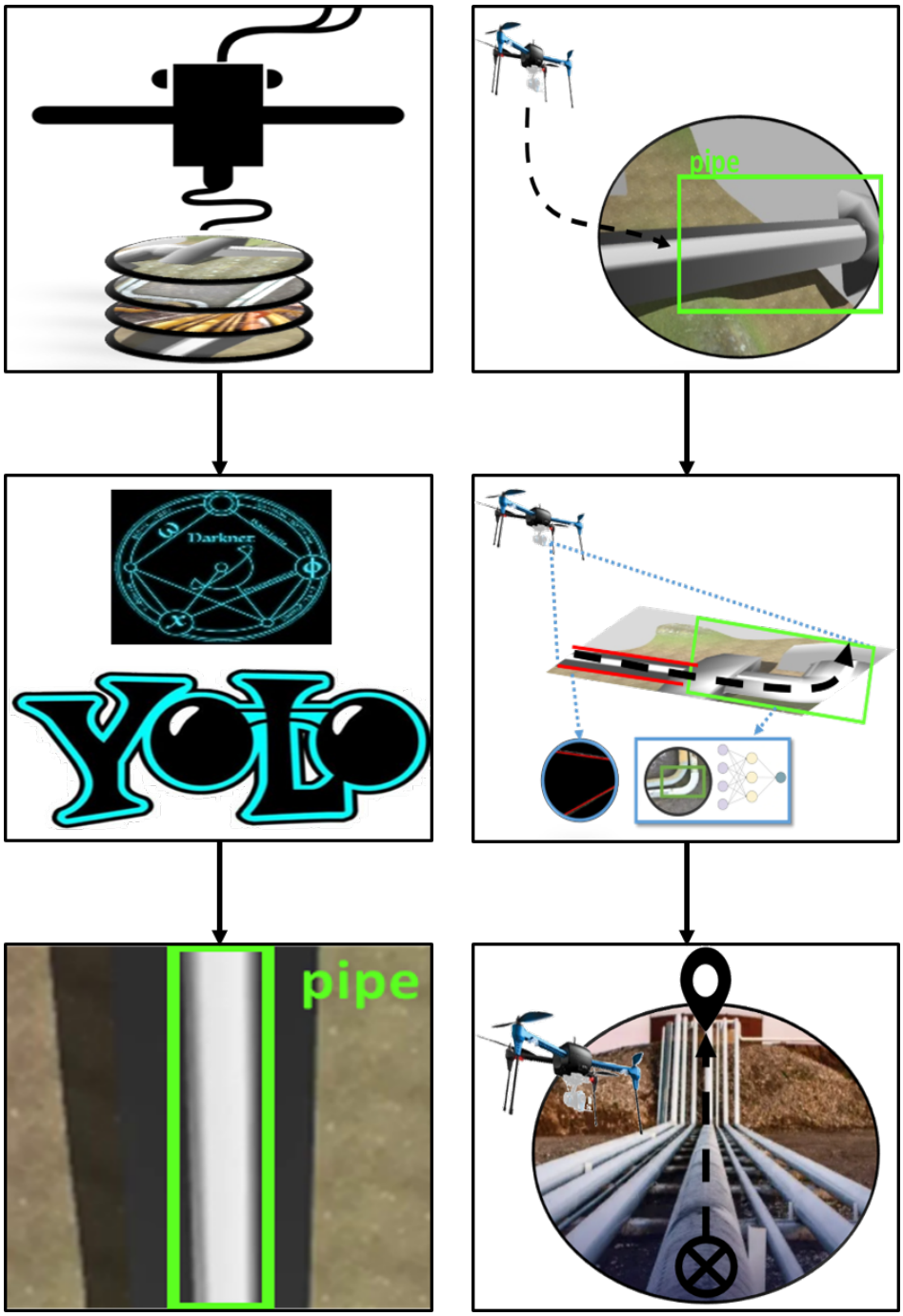
Figure III.2: General systems interfaces.

The dissertation development was divided into two main phases. The first one regards the CNN process to collect and classify images to produce the dataset, as illustrated in Figure III.3 (a), which will provide the CNN (i.e., YOLO) weights files for object detection. While the second part, III.3 (b), is about pipe tracking using YOLOv4/v5 and the line follower strategy to build the pipeline path follower. It was used at the beginning of this work, the YOLO version 4, then changed to version 5 because it is much easier to use. Following this idea, the strategy is based on a simple line follower algorithm with Hough Transform, for the straight pipeline parts and the CNN to ensure that the UAV is seeing a tube. Once the UAV finds a turn, the neural network can precisely identify the turn way (left or right), based on the detection position in the image, as represented in middle Figure III.3 (b).

The simple pipe follower algorithm is based on Canny edge detection that performs the image's filtering, locating, and corners (83). Then, this outcome is analyzed using the Hough transform to detect the lines (84). Once the straight pipelines are identified, an angle correction method is used to ensure the tube tracking, as applied in (2) with a similar control idea to (85).

Although the Hough transform is a traditional and efficient method to identify contour, it may suffer difficulties in a few situations. Therefore, this work proposes a CNN to find and orientate the UAV during curves.

The YOLO application is based on keeping the UAV focus on a tube. When the drone finds a turn, the next part still is a straight or curved pipe with a center line as a reference in the image frame. The object detection will be placed in the image frame on a specific side, Figure III.4 illustrates the



(a) (b)

Figure III.3: Project proposal. (a) Part 1. (b) Part 2.

bounding box detection and the reference line. The drone must turn counterclockwise if the detection is positioned on the left of the line reference. The drone must turn clockwise if the detection is on the right side. Completing the curve, the system uses pipe detection to activate the lines detection strategy, as illustrated in Figure III.4.

Based on the previous definitions, Figure III.2 represents the features used to implement autonomous pipeline tracking. Note that Python is the main program language that contains the logic to control

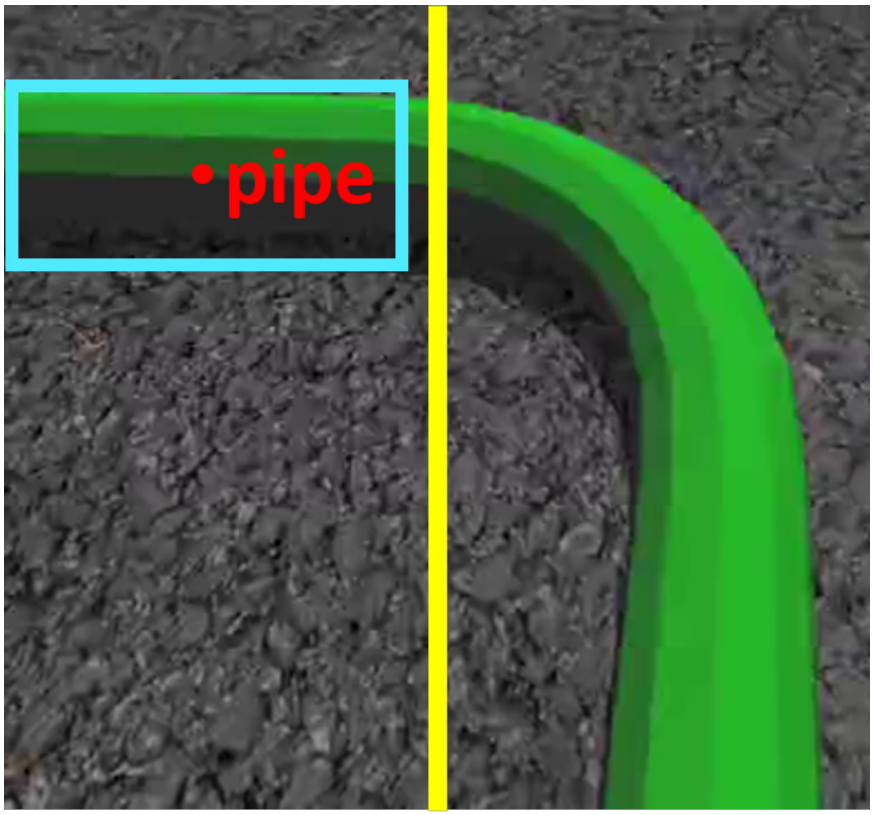


Figure III.4: Object detection to make the turn.

the UAV making interface between ROS, OpenCV (that contains Hough and Canny filters), and YOLO. Conversely, the MAVLink protocol makes the interface from ROS to PX4 (Flight Controller) and the UAV.

During the test phase, the topology represented in Figure III.2 was implemented in the Gazebo simulation environment to ensure human and low material risks and reduce development time and cost. The Gazebo was efficient enough for the application described in this thesis because the main point is the path plan based on vision (86).

III.1 Computing Vision and Dataset Creation

One of this work's basis is the image processing step to guide a UAV to track pipelines. Using just the line follower strategy, there is a huge chance that the UAV will get lost due to the different industrial structures that can easily be misunderstood as a line, such as electrical rail, retaining walls, or industrial area marking. To avoid this problem, this work also uses object detection to guarantee that the line detected (pipe binarized image) is a tube. Thus, the proposed research includes the YOLO training which demands the pipeline dataset creation.

Creating a dataset is tough work and can be resumed as; *(i)* acquires a synthetic image, preferentially from the UAV in a simulation environment; *(ii)* acquires real images, from the internet and other places;

(iii) and build the image database, with the data classification and network training.

The PX4 controls UAV and makes an interface with ROS and the ground station. The FCU has a UDP connection that transmits images from the drone to the ground. With this UDP connection, it is possible via Gstreamer to get this image data with Python and OpenCV. Once the image is in the OpenCV format, it is possible to use the CNN approach to identify the pipes and support the navigation, as Figure III.5 illustrates.



Figure III.5: General image interface.

Once Python and OpenCV receive the image, it can be collected as a frame in a specific time interval. This work collected a big part of the synthetic data with a script to get frames in different time intervals and drive the UAV by using an Xbox controller. This controller is connected to ROS using QGround Control, as described in (87). Thus, the front camera was used as a navigation reference, and the ground-pointing camera was used for frame acquisition.

Another method used to build the pipeline image database was googling image searching by words, such as pipeline, pipe corridors, pipe, refinery, and oil and gas plant. Every image that could be useful was saved one by one and then labeled around 644 images, among real ones from Google and synthetic ones from the Gazebo world. To classify them, it was used the application/website called Roboflow. This tool is free for public datasets and easy to use. Besides, it accepts different format annotations and framework exports, like YOLOv4 Darknet and YOLOv5 PyTorch formats. Roboflow also generates *.txt* bounding box file format, classes *.txt* and *.yaml* files. The classification was performed based on the pipe class defined as a pipe.

III.2 YOLO Training

After the YOLO creation (76) and the improvements made, YOLO is state-of-the-art for object detection with a pre-trained model capable of objects with high accuracy and efficiency (65).

YOLO is a pre-trained model that can easily be trained to identify objects. In order to do that, it needs to prepare the input images, the image labeling, and the annotation that contains the object position on the image for training, testing, and evaluation.

Almost all YOLO configurations were defaulted (88), except for the ones listed in Table III.1. With a transfer learning technique, in the first moment, the dataset was trained with the first 12 convolutional layers frozen and the default large YOLOv5 weight. After this, it was performed other training with the convolutional layers unfrozen, the hyperparameters with the weight of the first training (Table III.1), and the large architecture. The procedure is similar to (79).

The dataset has 644 labeled pipe images, divided into real and artificial ones (Gazebo images). Real images are from online tube pictures. The synthetic images are derived from the Gazebo simulation environment. All the images (synthetic and real) are annotated using pass-through data augmentation such as resize, flip horizontally, vertically, and grayscale conversion techniques of the Roboflow platform. The dataset, after this procedure, is composed of at least 1243 labeled pipe images split into 1000 for training, 129 for validation, and 64 for testing.

Table III.1: Hyperparameters used in YOLO's training step. Table from (1).

Optimizer	SGD (learning rate = 0.01)
Epochs	2000
Batch size	16
Patience	100
Image Size	448x448
Weight Decay	0.0004

III.3 Pipeline Tracking Controller

The pipeline tracking/follower controller concept is based on finding the pipe (Figure III.6 (a)), use Canny edge detection algorithm (Figure III.6 (b)) as pre-treatment and Hough transform (Figure III.6 (c)) lines for track the pipe.

Basically, the UAV image is first processed by YOLO algorithm to find and guarantee that the drone is seeing a pipe, Figure III.6 (a). The second stage is the Canny edge detection kernel (83), which is responsible to perform the image filtering, that is, giving the locating and corners of the image,

Figure III.6 (b). Then, the outcomes go to the Hough transform algorithm to be analyzed during the parameterization, as demonstrated in (84) and Figure III.6 (c).

The Hough transform is a mechanism for extracting features from an image using a voting policy. Note that a single pixel can have an infinite number of lines passing through it. The author refers to the work of (89) for more detailed information about this process. Besides the featuring extraction with Hough transform this part also has the angle correction logic to ensure the line follower correction.

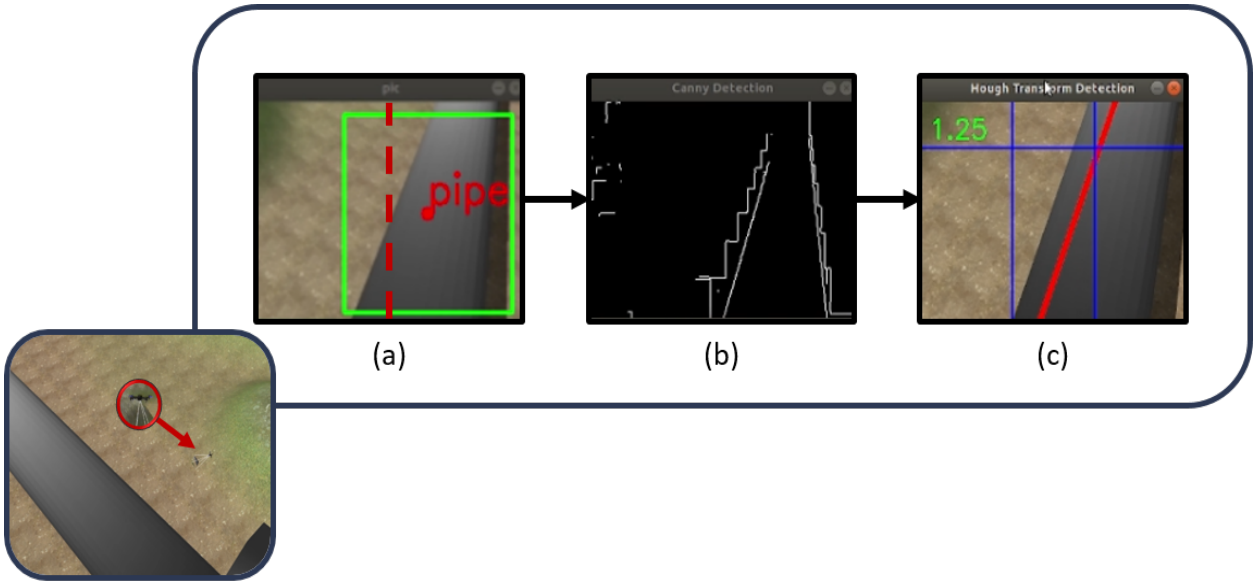


Figure III.6: Pipe follower flow data. (a) pipe CNN detection and approach. (b) Canny edge detection. (c) Hough transform.

This procedure specifies a narrow line, as the red line reference in Figure III.6 (c), represented by the angle θ from the normal and delimited by the distance ρ from the origin, as shown in Equation (III.1). In this equation, the distance corresponds to the geometry when applied in a way to consolidate a space of **n** points (Equation III.2 is the equation II.2 rewritten). Note that the Hough space and the Cartesian coordinate can be transformed into each other. This process is illustrated in Figure II.2. From this figure, it is possible to identify a line by the number of intersections. If the number of intersections is greater than a minimum value, the probability of having a line in the area of the analyzed image is more accurate. For this work, we set up a *threshold*, i.e., a minimum amount of intersections.

$$y = (-\frac{\cos\theta}{\sin\theta})x + (\frac{\rho}{\sin\theta}) \tag{III.1}$$

$$\rho = X_n \cdot \cos\theta + Y_n \cdot \sin\theta \tag{III.2}$$

Note that the polar coordinate (ρ, θ) represents a line passing through a point (X_1, Y_1). From Figure II.2, it is possible to get the Equations (III.3) and (III.4). So, by making the Hough transform into a

single line, the planning path (parameters ρ and θ) can be obtained. Figure III.7 illustrates the yellow line reference used for angle correction guidance, in Figure III.7 (a) the yellow line is aligned to the tube, and in Figure III.7 (b), the UAV needs correction despite the blue lines limits, that the path must follow. Important to note that the yellow line is a reference of the middle point between two Hough lines marked in blue on tube extremities.

$$X_1 = \rho * \cos\theta \tag{III.3}$$

$$Y_1 = \rho * \sin\theta \tag{III.4}$$

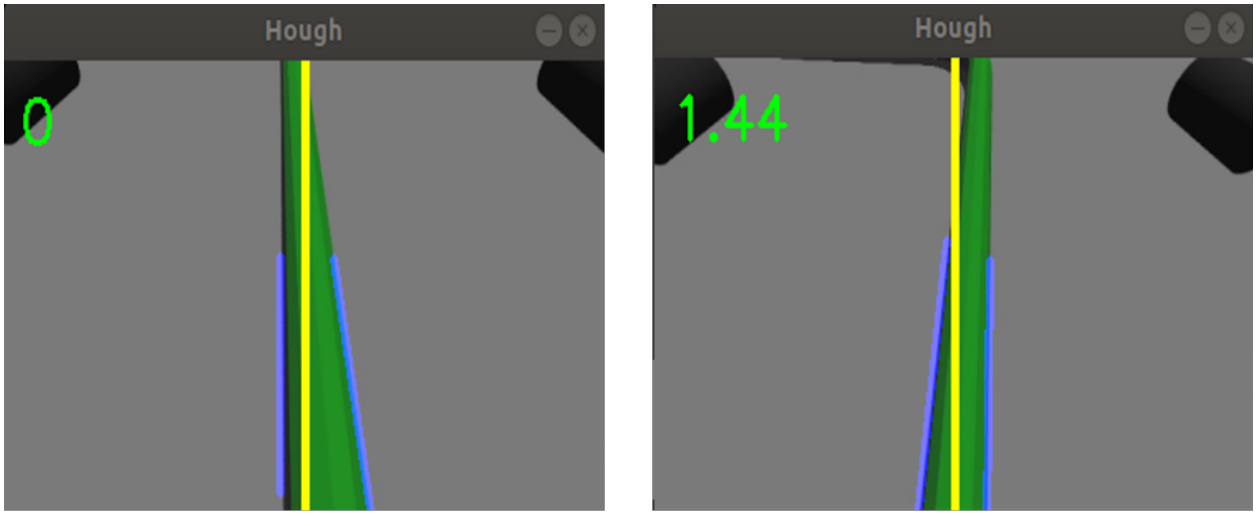


Figure III.7: Angle Correction Illustration. (a) no correction. (b) needs correction

It is important to highlight that the UAV has embedded into the PX4 FCU a Proportional-Integral-Derivative (PID) controller used to regulate speed, direction, and orientation. In this way, the parameters ρ and θ generate the x and y position of the line that will trigger velocity commands to make the UAV move forward, and adjust displacement in the y-axis and yaw-axis. The PX4 PID controls the motor to move the drone. This work is using a simple control strategy, similar to on/off controller, which the ground station publishes speed setpoints to the FCU and the internal control moves the drone softly. By default, it is defined the velocity of 0.5 m/s to be published based on CNN and Hough transform results.

Note that before the work was published in (1), this research was using the standard Hough transform that traces a line that passes through two points, instead of linking the two points as the probabilistic Hough transform, that have been implemented to the final part of this work (59). Figure III.8 (c) shows the probabilistic Hough transformer.

The probabilistic Hough transform is also implemented in the final part of this research for the line follower script along with the ROI method, which is responsible to focus the image processing into the area where it is needed. Avoiding any other possible lines harm the performance and reduce the line size, minimizing error as stated in (61). Figure III.8 (b) shows the ROI implementation with the area marked with a blue polygon (i.e., triangle and rectangle).

Putting together the Probabilistic Hough transform and the ROI, is possible to see that the probabilistic Hough transform delivers a much cleaner line detection and better image output as can be seen in Figure III.8 (a) with the UAV image, (b) the Canny and the Canny with ROI and (c) with blue probabilistic Hough lines.

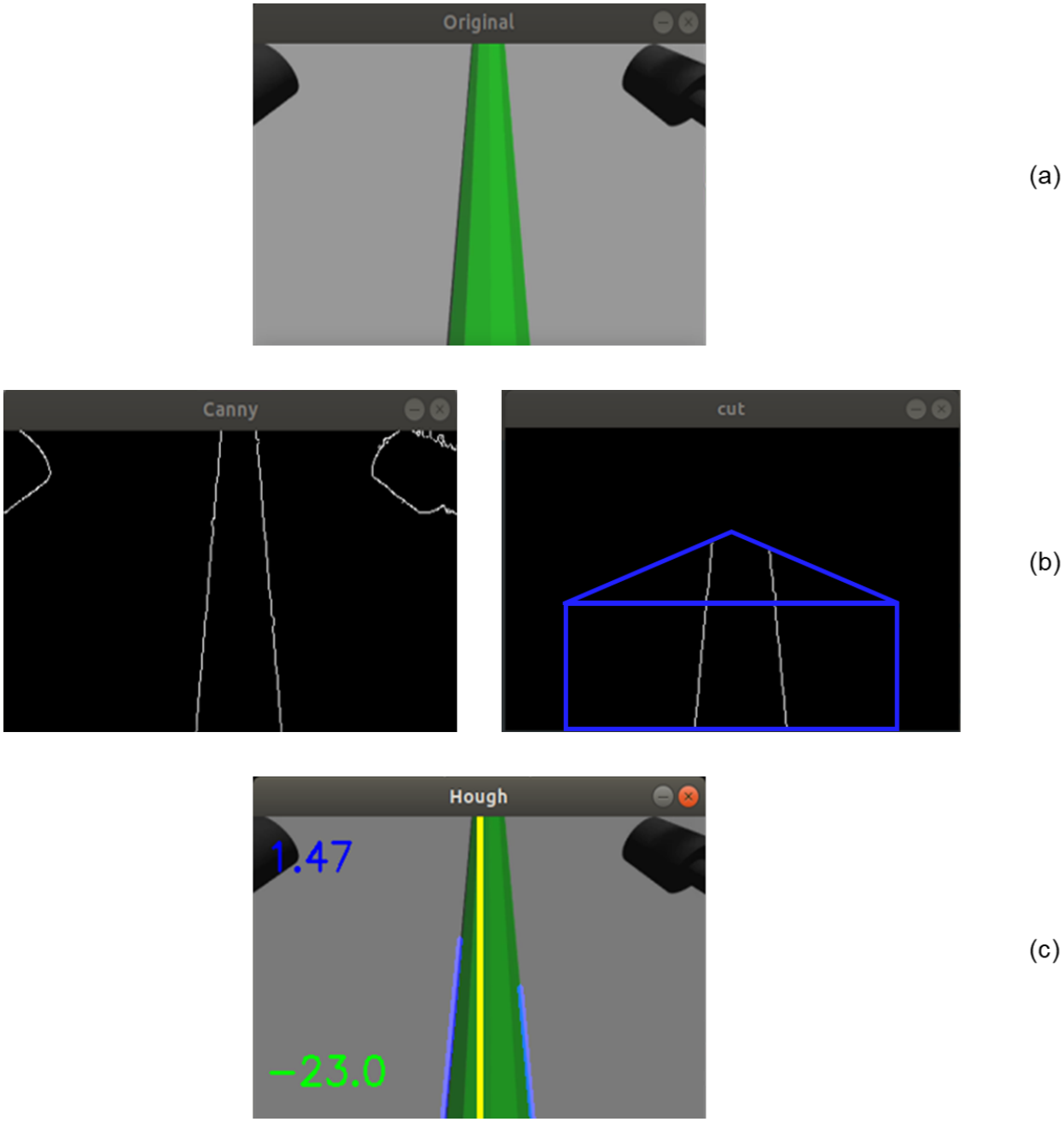


Figure III.8: Line follower outputs. (a) Raw image. (b) Canny filter without ROI on the left and with ROI on the right. (c) Hough image with the reference of the life follower.

In addition to the line follower controller, this work proposes object detection with YOLO algorithm to guarantee the pipeline tracking performance and to ensure that the UAV is capable to make curves. Figure III.9 (a) illustrates the pipe detection during the straight parts, the turn and another straight on the left of the image (taking the red dotted line as reference), indicating that the drone must turn counter-clockwise, as well as after the curve it remains going forward.

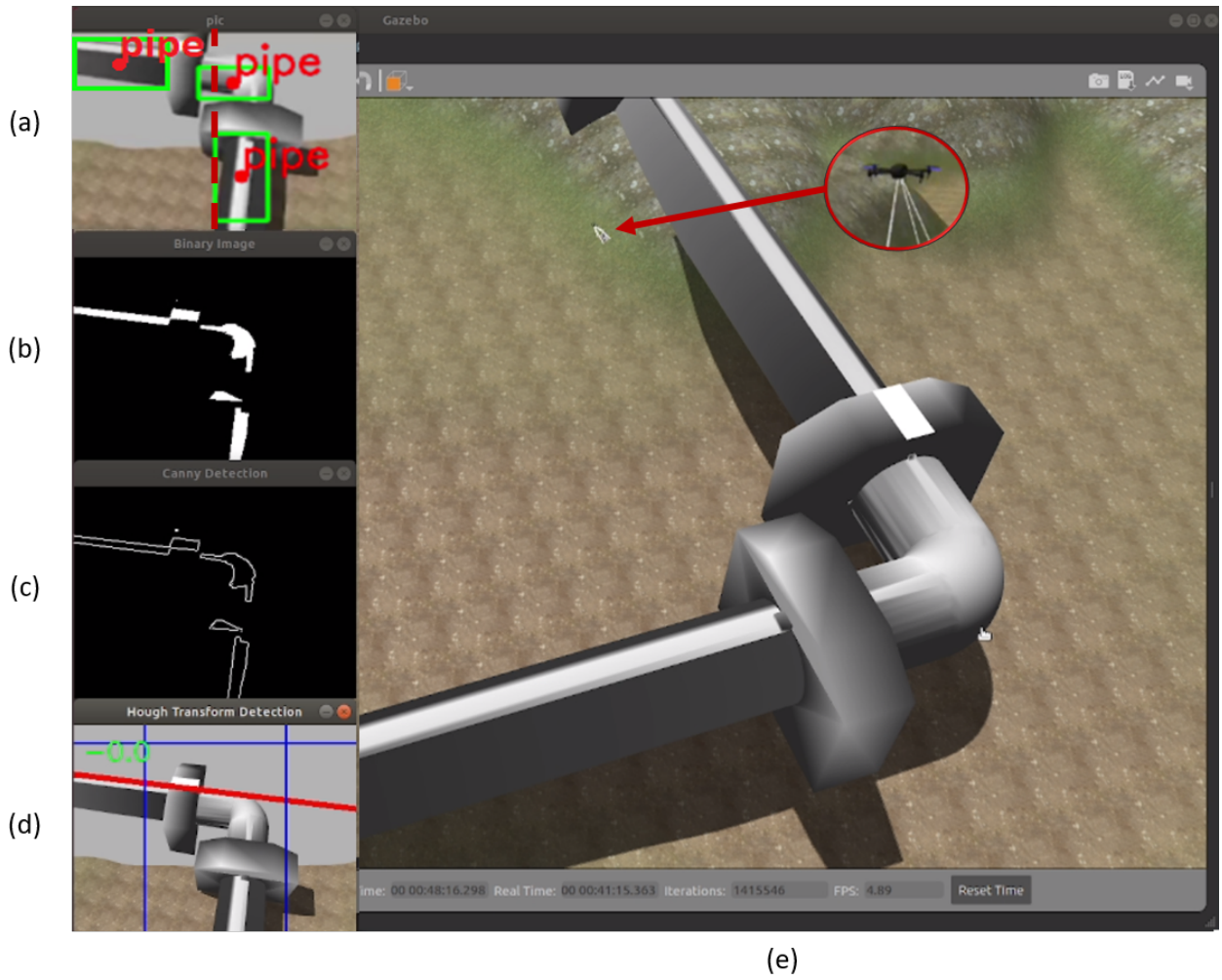
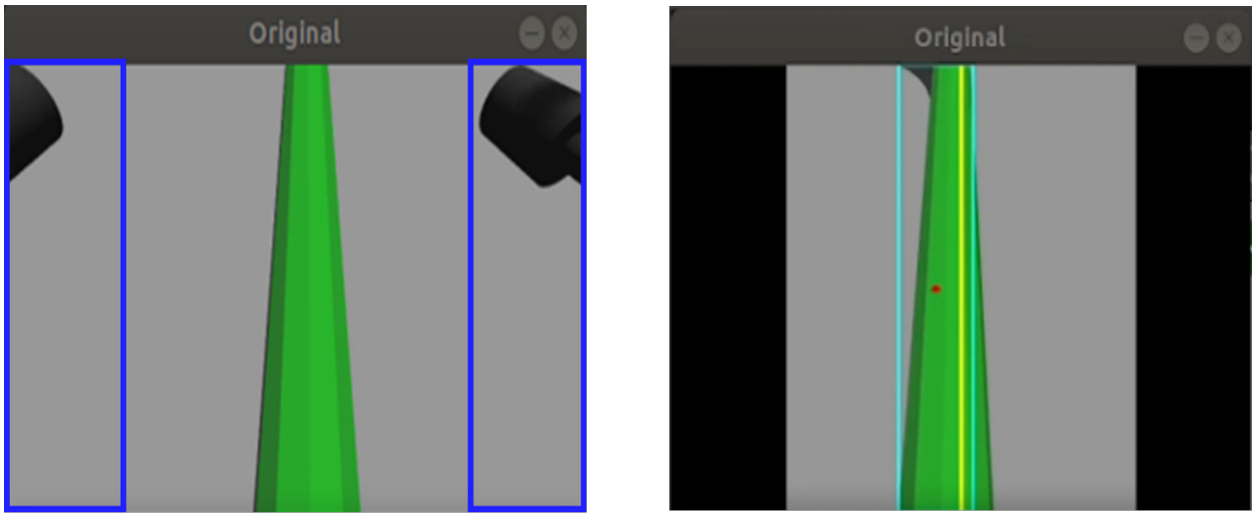


Figure III.9: Navigation Representation. (a) Pipe detection with the reference line. (b) Binarized image. (c) Canny kernel. (d) Hough transform. Part (e) Simulated environment. Image adapted from (1)

Note that this work at first used the Iris quadrotor drone, which has no landing gears. Then, the vehicle was changed to the hexacopter Typhoon 480, for some reason in the simulation setup, the author was not able to retract the landing gears in the off-board mode. Because of that, it was used a ROI solution for the landing gear not be seen during the YOLO image processing. 20% of each frame side was ignored, as shown in Figure III.10

With all these algorithms, the UAS searches for the pipeline using YOLOv5 strategy. Once the bounding box of the pipe is found, the system receives the probabilistic Hough transform lines, the line follower method is active to guide the drone. The UAV moves with the speed of 0.5 m/s forward, if the Hough lines have an angle difference to the yellow reference line +/- 1 degree (Figure III.8 (c)),



(a) (b)

Figure III.10: ROI for YOLO usage. (a) Raw image. (b) YOLO running with ROI set up.

there is a yaw-axis (twist.angular.z axis) correction with the speed of +/- 0.3 m/s. When the x_1 and x_2 center is displaced from +/- 30 meters (i.e., parameter set to the simulation environment, it must be adjusted to real-world testing, related to altitude set point and another field fine-tuning) from the yellow center line (Figure III.8 (c) green left lower number), the system corrects the UAV in the y-axis with speed 0.5 m/s to guarantee the centralization. Once the UAS finds a curve, based on the center of the bounding box and the reference line, the system can identify for each side, counter clock or clockwise (yaw axis/twist.angular.z axis) the UAV must turn. This logic is resumed in Figure III.11, note that in case of the drone misses the pipeline it will remain in the position sending to the ground station the current position, this parameter could change according to the real testing conditions, and assumptions.

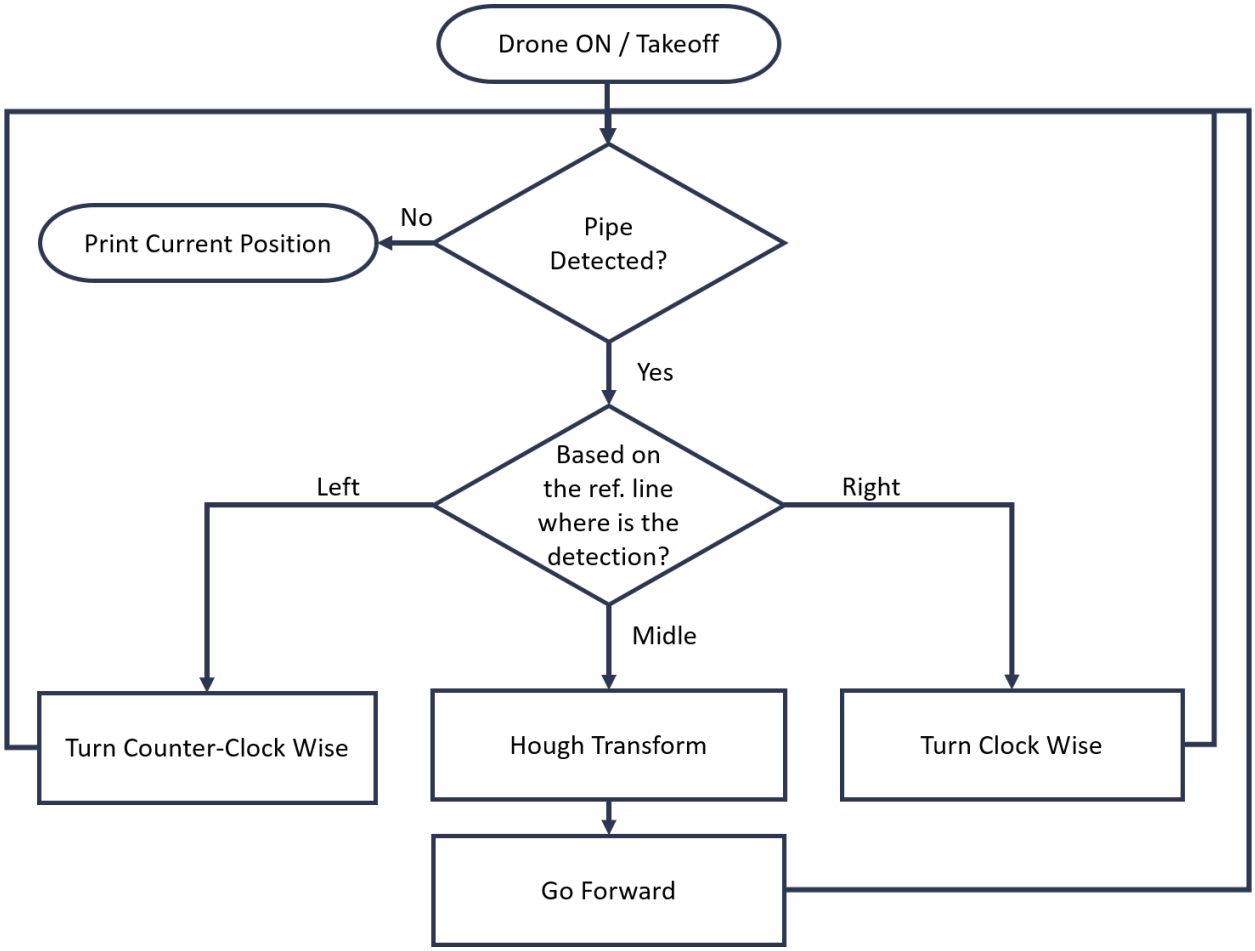


Figure III.11: Navigation logic diagram.

Chapter IV Results and Discussion

All the simulations are performed on a desktop computer with 16GB RAM, Intel® Xeon® CPU E5-2640 v3 2.60 GHz processor, and Nvidia RTX 2060. The operating system distro is a 64 bits Ubuntu 18.04.6 LTS (Bionic Beaver). The algorithms that control the UAV using Python 3 as shown in Chapter III, controls (ROS/PX4 FCU) and YOLO image processing.

In resume, the UAV approaches the pipeline through the YOLO pipe detection, actives the line follow strategy while a tube is being detected. To make the curves the CNN detects the tube in the quadrant left or right of the framed image, which commands the drone to turn (yaw axis) counter clock or clockwise, as illustrated in Figure III.9 and III.11.

Most of the results of this research were published in the article (1), which is also one of the main work contributions, including the data sets posted in IEE DataPort (90). After the article the main 2 objectives were to try to improve the dataset, such as Roboflow repository (91) (92), the implementation and test functionalities related to the UAV and YOLO usage with ROS.

IV.1 Simulation Environment

For safety and economic reasons, the simulation environment setup is one of the key points to this research's success. The community is in continuous development, and it is possible to find a couple of worlds ready to use, such as the world available in Github (93). Figure IV.1 illustrates an example of this mentioned Github figure. This world comprises straight yellow lines, 90 degrees corners, and walls. It is the base case for the line followers concept. This was the first world used to test the line follower, proposed in this work for straight pipes line parts and implemented in the article (2).

Two world files were developed to make the simulation closer to the real world and simulate the behavior of pipeline tracking. At first, during the learning phase, a world was created based on the standard Gazebo library Winding Valley Height for the ground base and a Solidworks pipeline model with two big 3D models. The first one is completely straight, and the other has turns and big partitions, Figure IV.2 (a). The second world, lighter to be processed with Gazebo, was modeled in Solidworks as one simple tube, with straight, 45, and 90 degrees turns, Figure IV.2 (b).

These two models were designed with Solidworks' basic functionalities: parts and assemblies. They were exported to the URDF file (i.e., file compatible with ROS and Gazebo models) using the URDF

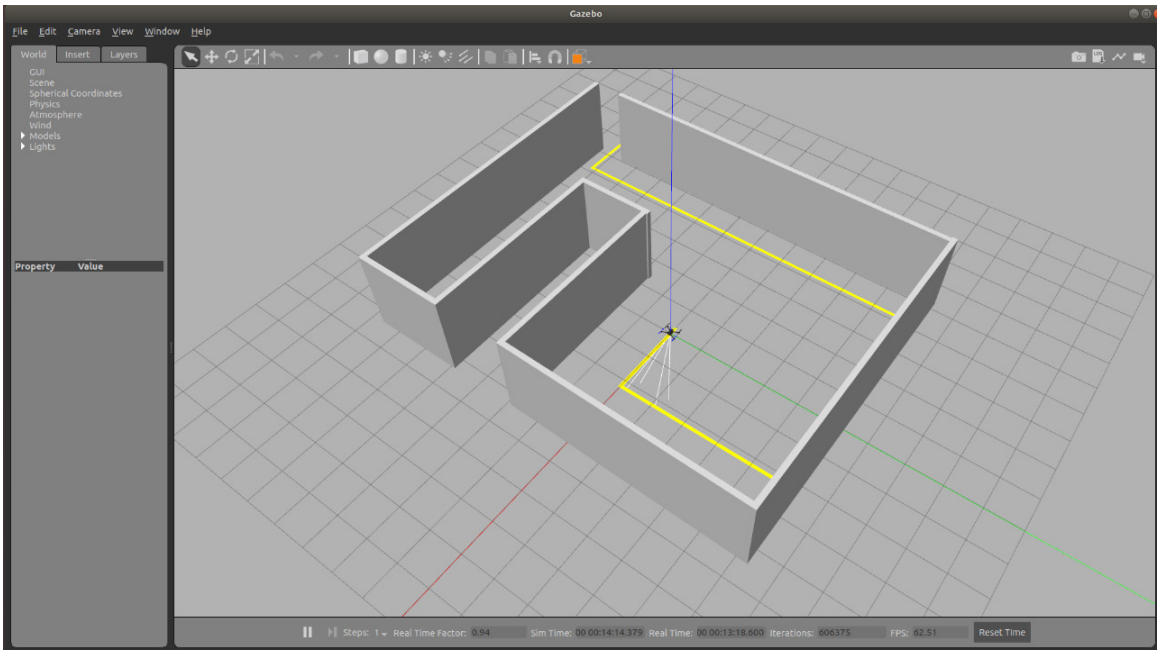
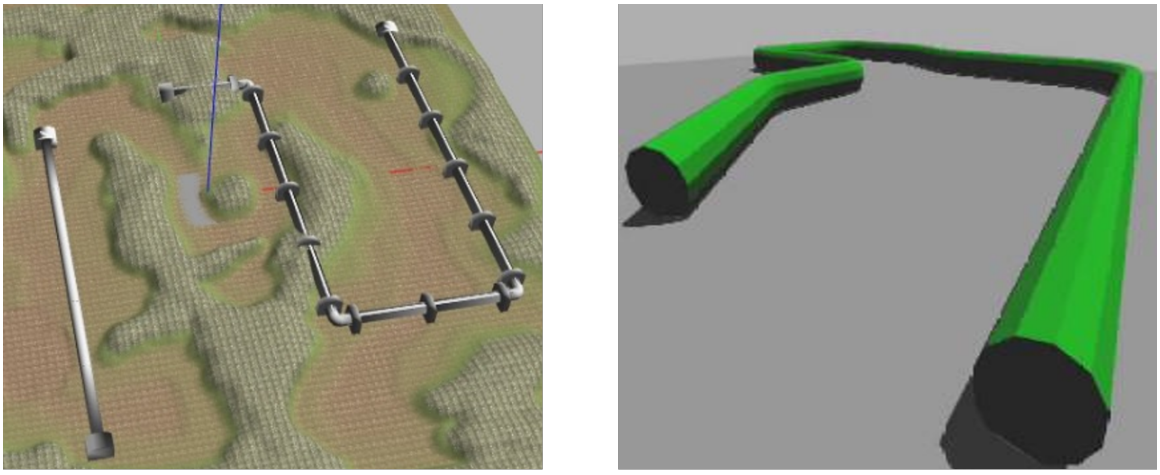


Figure IV.1: Line Follower World. Image made by Lucas Silva Sousa and Yago Melo.



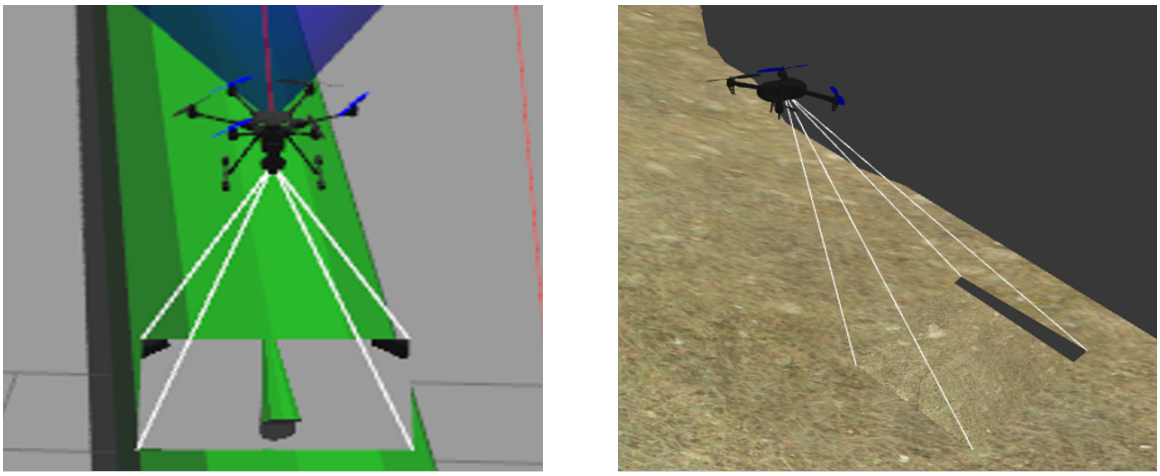
(a) (b)

Figure IV.2: Gazebo Worlds. (a) Heights Valley with big tubes, image from (2). (b) A simple world with a green tube.

Exporter, an add-in developed by ROS Community to improve simulation setups, available in the link (94).

By default, Gazebo has many 3D robots and UAVs that already have the dynamics pre-configured and ready to use. At the beginning of this research, the Iris quadrotor with a static camera was chosen for the simulation tests. Then, the UAV was changed to the hexacopter Typhon H480 because of the stability of camera usage (system/simulation level, Gazebo Gstreamer plug-in is already stable), and the default gimbal. Figure IV.3 shows the UAV Typhon H480 (a) and Iris (b).

In order to simulate the proposed methodology, a simulation environment was created containing the



(a) (b)

Figure IV.3: UAV models. (a) Typhoon 480. (b) Iris with the camera.

pipes and the Iris/Typhoon H480 drone. The UAV experiences in the virtual world to find the pipeline approach and track as a line follower. Figure III.3 illustrates the overall idea.

IV.2 YOLO Training and Results

One of this work's contributions is an image dataset for pipeline detection (90) to make possible the approach stated in this work, the pipeline tracking, and support future works.

The dataset contribution of this dissertation work has around 644 images, among them synthetic from the world created in the Gazebo simulation environment and real images grabbed online and other facilities (90). It is also part of the dataset the *.txt* files with the bounding boxes location and the YOLOv5 weighs (.pt and .onnx files). Before the training, the dataset passed through a data augmentation to flip horizontally and vertically, achieving 1197 images, composed of 1000 for train, 129 for validation, and 68 for testing.

The best result of this work and the dataset are available in the IEE data port (90) and in the Journal Drones from MDPI article dataset (1). Table IV.1 shows the results of different training tries, with different YOLO architecture (small, medium, large and x large), performed during the research with the image quantity, mAP, and confidence.

To improve the YOLO, several training tests were made to understand the network's behavior to this application. Part of the results is listed in Table IV.1, and the line table description is listed below.

1. **MDPI drones article dataset**: It is the dataset built to the article (1) and has around 644. Among them, there are synthetic and real images. During the training, it was used flip horizontal and vertical as data augmentation, transfer learning, and fine-tuning methods described in Chapter III, among others.

2. **Pipeline Tracking**: It is a dataset with the original 644 images from the thesis qualification, 530 images from Dall-e, and the created world with a green tube. This one was used to flip horizontally, vertically, and 30% of images in grayscale, a total of 2100 images.

3. **Pipeline Merge v1**: It is a dataset with the original 644 images from the article (1), and more 805 images from Dall-e, real ones, and images from the world with a green tube. The training used the data augmentation technique: flip, horizontal, and 25% of images in grayscale.

4. **Pipeline Merge v2**: it is the same dataset base from Pipeline Merge v1 with more than 100 background images with data augmentation techniques.

5. **Pipeline Merge v3**: it is the same dataset base from Pipeline Merge v1 with a different data augmentation setup used to flip horizontally and vertically.

6. **Pipeline Merge v2 trained with MDPI article weight**: It is the Pipeline Merge v2 being trained with the weight presented in the article dataset available at (1).

Table IV.1: YOLO training results

#	Dataset - Project	Architecture	Images Qtd	Training Img	Valid Img	Test Img	mAP50	Confidence
1	MDPI drones article dataset	s	1197	1000	129	68	0.721	0.935
2	Pipeline Merge v2 trained with MDPI article weight	s	3029	2600	286	143	0.857	0.848
3	Pipeline Tracking	s	2453	2100	232	121	0.758	0.866
4	Pipeline Tracking	s	2453	2100	232	121	0.794	0.846
5	Pipeline Tracking	m	2453	2100	232	121	0.749	0.870
6	Pipeline Tracking	l	2453	2100	232	121	0.749	0.802
7	Pipeline Tracking	x	2453	2100	232	121	0.749	0.802
8	Pipeline Merge v1	m	2929	2500	286	143	0.778	0.824
9	Pipeline Merge v2	s	3029	2600	286	143	0.792	0.860
10	Pipeline Merge v3	s	2129	1700	286	143	0.777	0.860

The platform used to store this dataset is the IEE Dataport. This is an IEEE repository with 2TB for free to support research and encourage data science. The created dataset is available at (90). The CNN was labeled in the Roboflow website and trained using the Google Colab, a web-hosted Jupyter notebook service that disposes of 12 GB and 16 GB Tensor Processing Unit (TPU) Tesla T4, using the (95) as a guideline.

The dataset model is composed of real and synthetic (from the Gazebo simulation) images. Figure IV.4 shows the learning curve of the YOLO training results for the bounding boxes detection (a) and the object classification (b) during 280 epochs, which took around 4 hours to train in the Tesla T4 from Google Colab.

After the training, the weights are locally stored and used in the ground control station to detect the pipeline with the script.

The Precision-Recall curve presents the trade-off between precision and recall in different thresholds and the mean average precision, mAP. Figure IV.5 illustrates the precision-recall results for this work, the first item of Table IV.1.

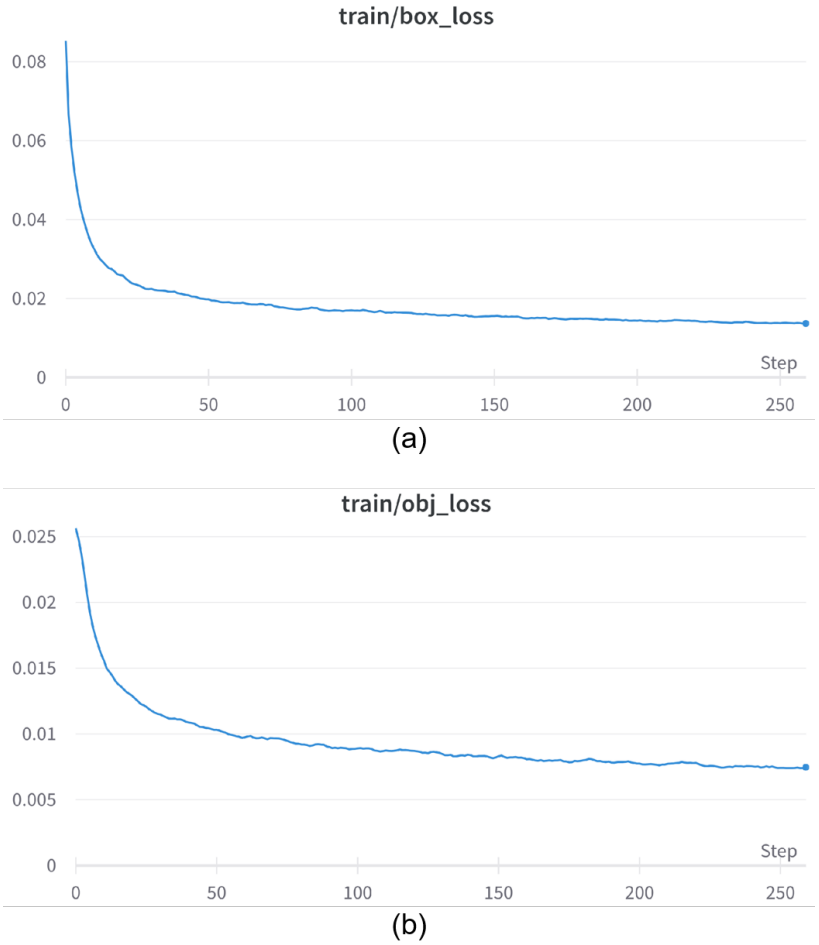


Figure IV.4: Box and object loss through the 280 epochs of training. (a) Bounding box regression loss (Mean Squared Error) X Epochs. (b) Confidence of object X Epochs. Image from (1)

Table IV.2 presents the confusion matrix results from the trained model as evaluation data from YOLOv5 PyTorch configuration. Note that the model identifies the proposed object 64%, and there is no false positive because this dataset has no background images, just like the penguin dataset, available at (79). As this work dataset is small, that is, only one class and around 1000 images, the absence of a background image is a common practice.

Table IV.2: Model confusion matrix (1).

	Pipe (Predicted)	**Background (Predicted)**
Pipe (Actual)	0.64 (TP)	0.36 (FN)
Background (Actual)	0 (FP)	1 (TN)

Figure IV.6 gives examples of an unburied pipeline detection due to the YOLO. It is important to note that the images on the left side (i.e., green synthetic pipe) and the right-down image (i.e., black and white tube) are not part of the trained dataset. This YOLOv5 pipeline detection contribution has significantly good results performing the proposed task.

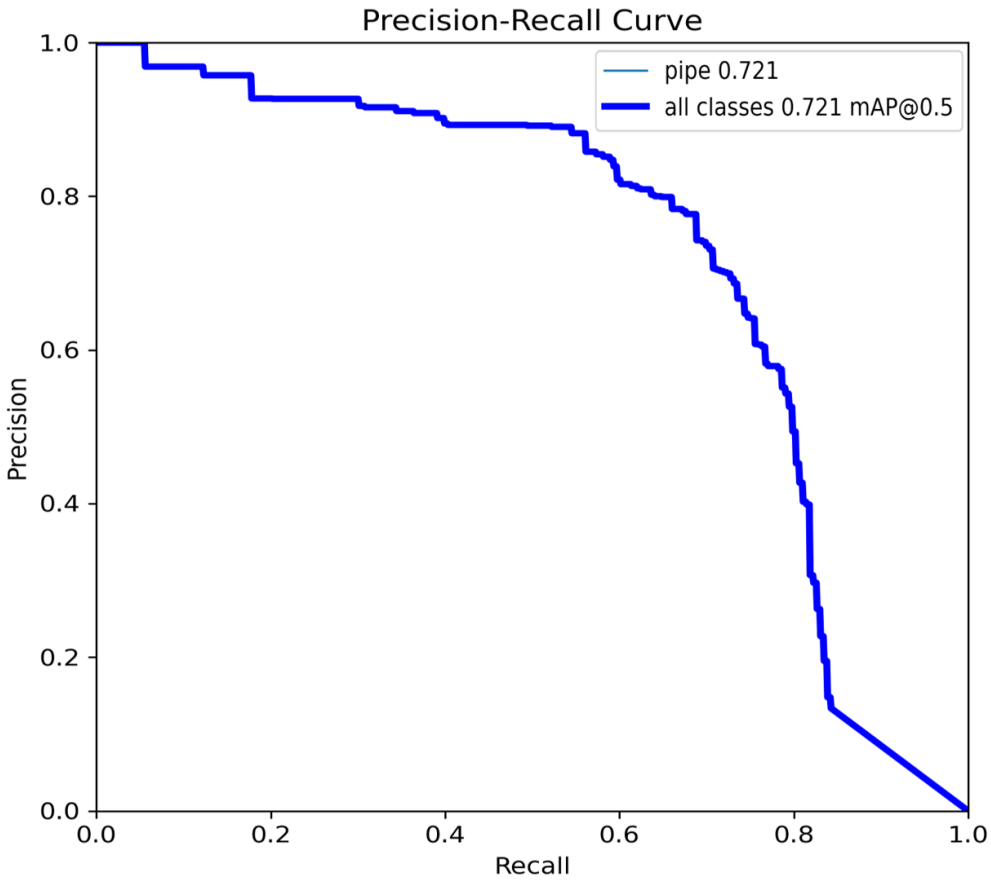


Figure IV.5: Pipe class Precision x Recall. Image from (1).

IV.3 Pipeline Follower Simulation Results

The takeoff height in the simulation was five meters high for safety reasons. Then, the UAV goes to the flight height and maintains it during the tracking. Figure IV.7 presents the UAV approaching the pipeline from above and starting to follow the unburied pipeline using the follow line methodology, same as used in the articles (2) and (1).

In Figure IV.7, it is possible to verify the steps of the pre-processing computer vision technique for the path follower algorithm. Figure IV.7 shows the raw image from the drone in green, the Canny kernel in yellow, and the Hough transform in orange, which uses as a reference the yellow center line inside the blue line limits (i.e., probabilistic Hough line transform outputs). While the yellow line has +/- 1 degree (blue angle number on the upper left) of error and the y-axis error is +/- 30 (green number on the lower left), the UAV goes forward. If the angle offset is missing, the UAV turns (yaw axis) counter clock or clockwise, and if the Hough line center point has a deviation bigger than the offset, the UAV makes the y-axis correction, moving left or right. Note that the blue Hough lines are just drawn in the region of interesting described as in Chapter III, different from the articles (2) and (1) that are not using region of interesting and use standard Hough transform, which traces a big that pass trough 2 points instead of the probabilistic Hough transform that connect 2 points of the detected lines.

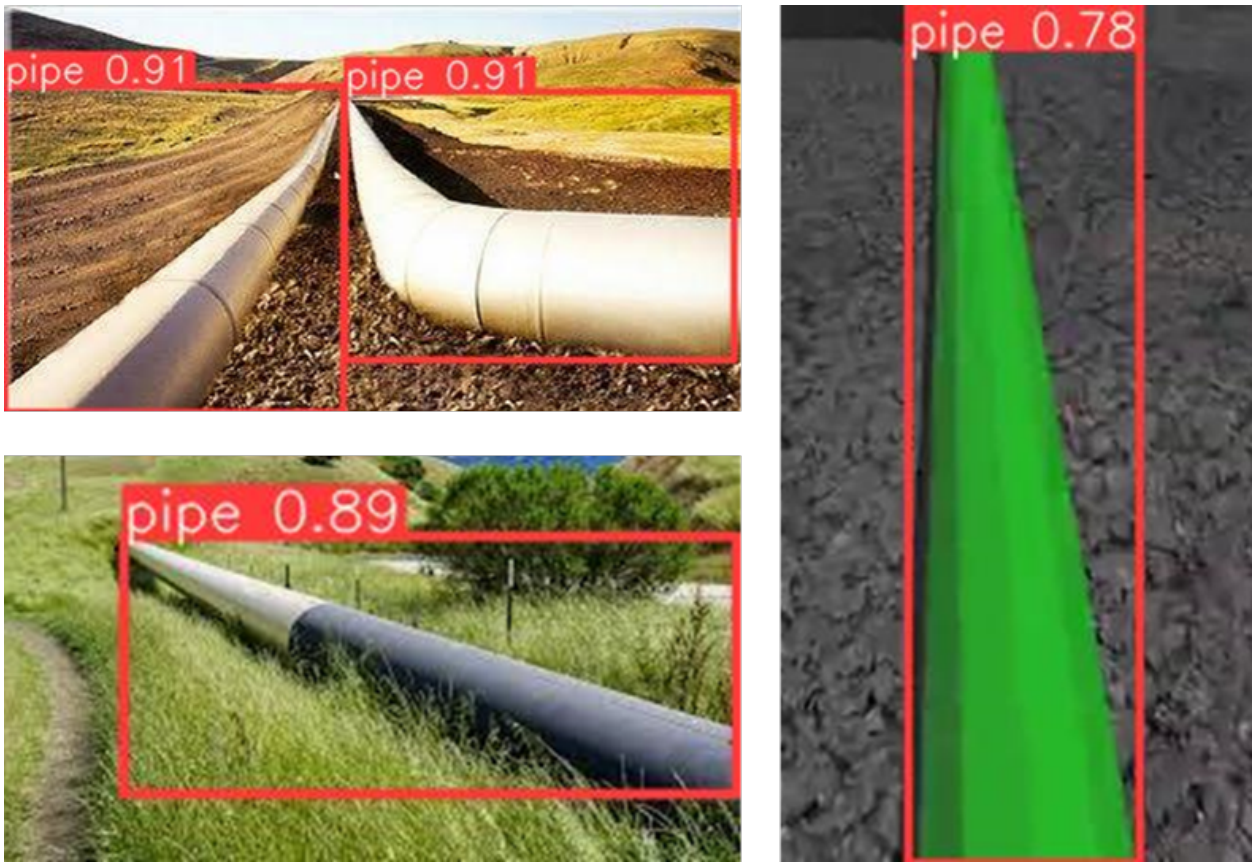


Figure IV.6: Unburied pipeline example used for our dataset.

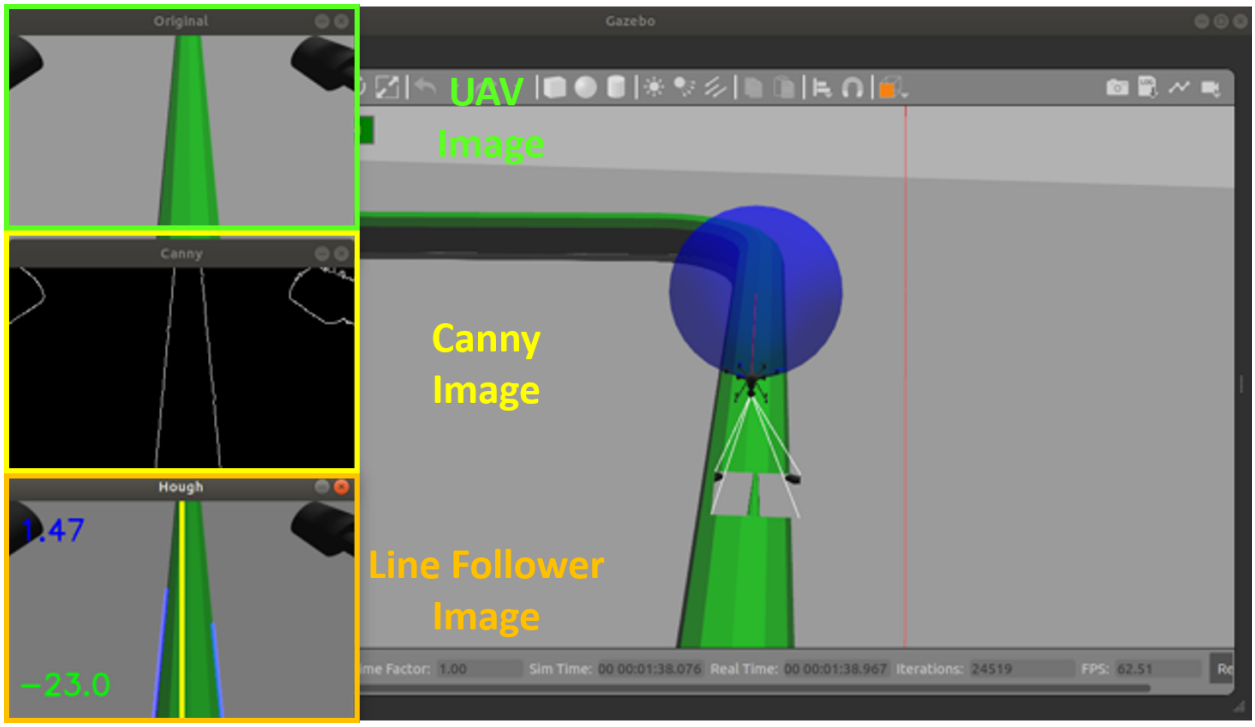


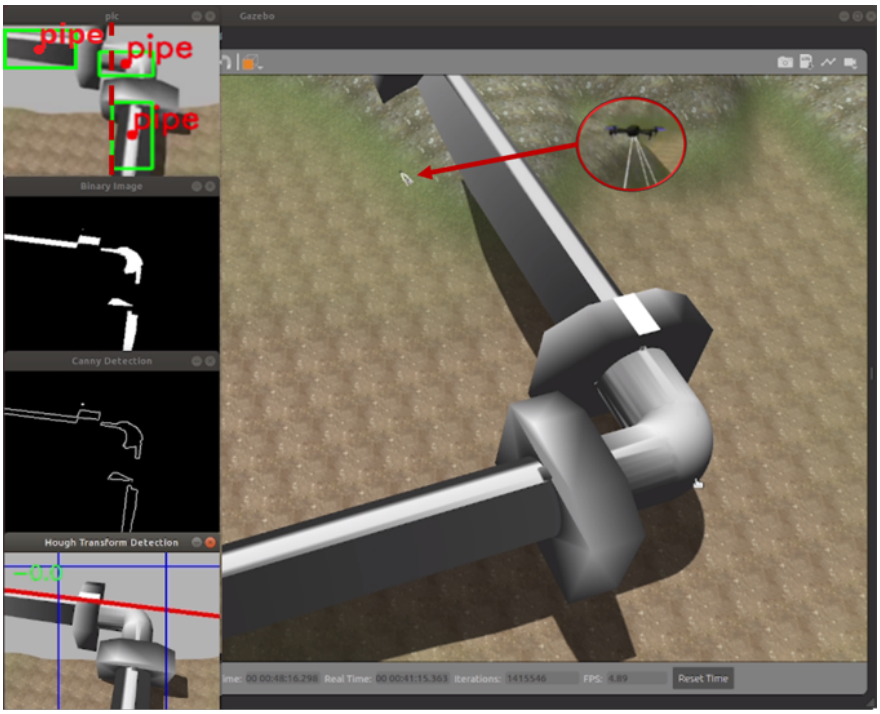
Figure IV.7: UAV performing the line detection.

IV.4 Pipeline Tracking Simulation Results

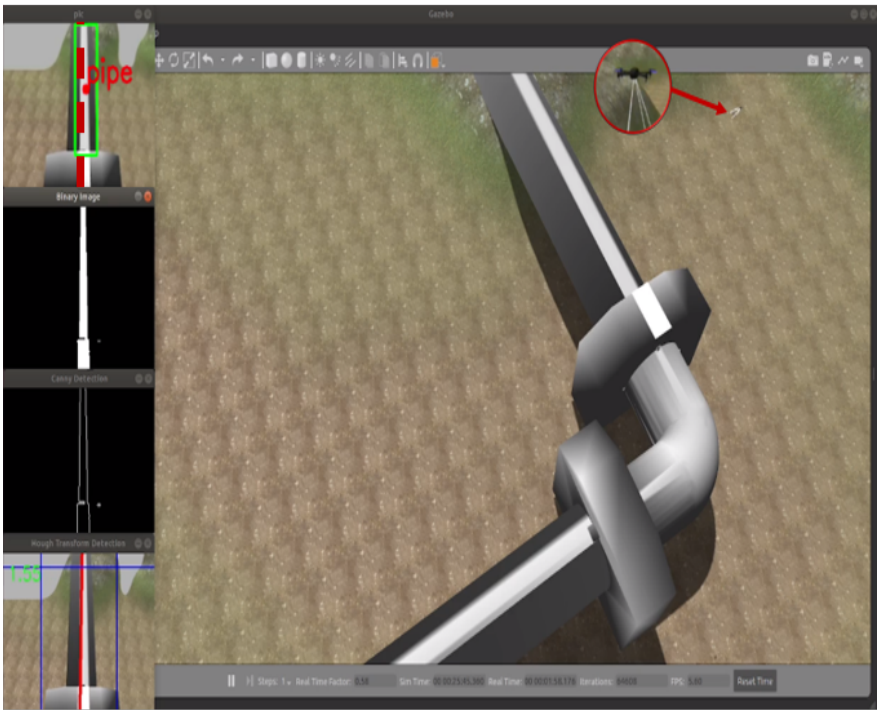
In pipeline tracking, the idea follows almost the same as the pipeline follower. The difference is that in the pipe track, there is YOLO CNN to detect the pipeline, activate the line follower algorithm and

guarantee that the UAV makes the turn.

Table IV.3 gives the error during the simulation test (around 0.0074 meters from the reference in the straight part, with 10 meters of height) and YOLO 87% confidence. The system could satisfactorily perform the tasks, as Figure IV.8.



(a)



(b)

Figure IV.8: UAV performing the pipeline track. Image from (1).

To guarantee that the drone will always make the turn for the correct side, in the final research part, the author focuses on the script improvement that commands the UAV. In order to improve the turning capability in the script, it was used side object detection, which means that if the pipe detection center point is on the right of the yellow reference line, the UAV must turn clockwise. However, if the center of the detection is on the left, the UAV must turn counterclockwise, just as defined in Figure III.11. Figure IV.9 shows the UAV coming from straight tube parts (a) and (b) when the Hough lines are almost none (c) the drone makes the turn counterclockwise, where is the bounding box detection, (d) and (e) and then goes straight again (e).

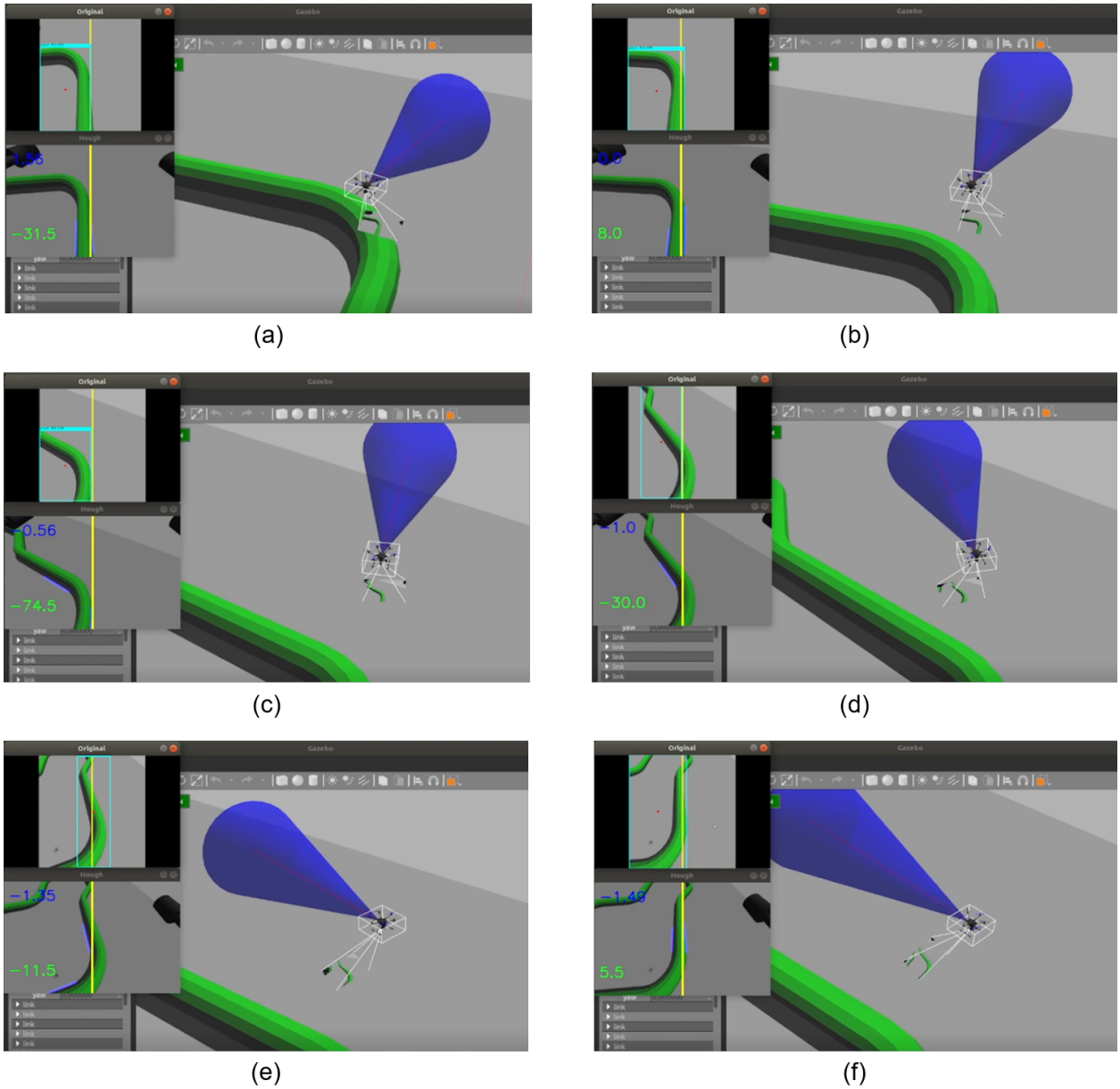


Figure IV.9: UAV performing the pipeline tracking in the simulation environment.

IV.5 Real Flight Results

After testing the system in a simulation environment, the system can be tested in a real-world constrained scenario to show the robustness of the proposed methodology. As a safety point, the UAV had a set point of 10 m high from the pipes in an open space with no strong winds.

For the first test, only a straight pipe section was chosen for safety reasons. The tube has around 6 inches in diameter in the color green. The pipe tracking system results in Figure IV.10, in which the drone finds the pipe (a) and activates the line follower strategy: image binarization (b), Canny Edge Detection (c) and the Hough Transform (d).

As Table IV.3, note that this test resulted pretty accurately with 0.0111 meters of deviation (based on reference line) and the YOLO confidence of 97% during the tracking. The UAV position compared to the reference line and pipe is also illustrated in Figure IV.11 (a) and (b). Note that visually is clear that the drone, on average, has an error tending to zero (Figure IV.11 (b)), showing that the system is accomplishing the proposal.

Table IV.3: Comparison result between simulation and real environments. Table from (1).

Parameter	Simulation	Real Environment
Error Deviation (m)	0.0074	0.0111
Mean YOLO Confidence	0.8774	0.9765

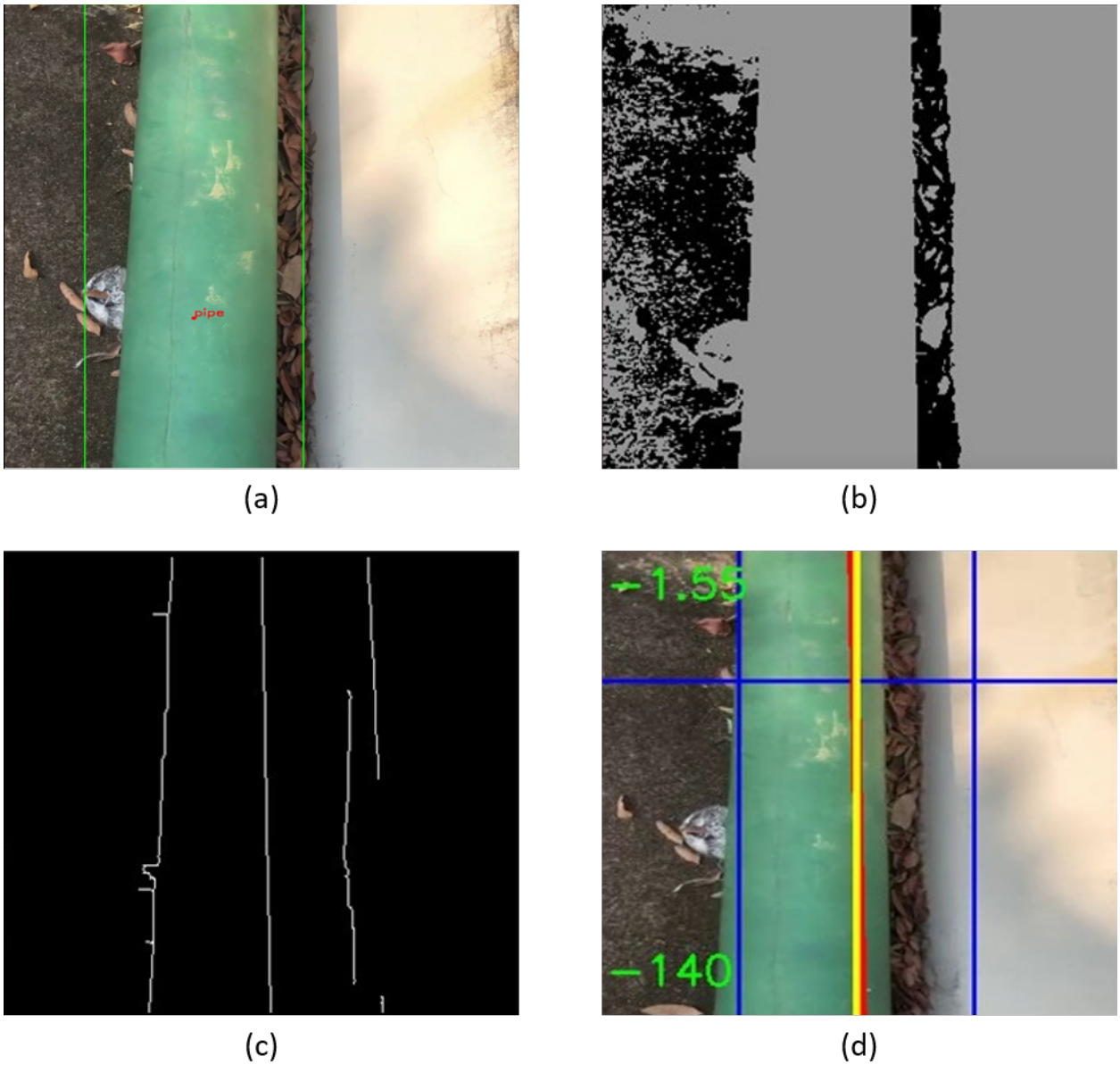
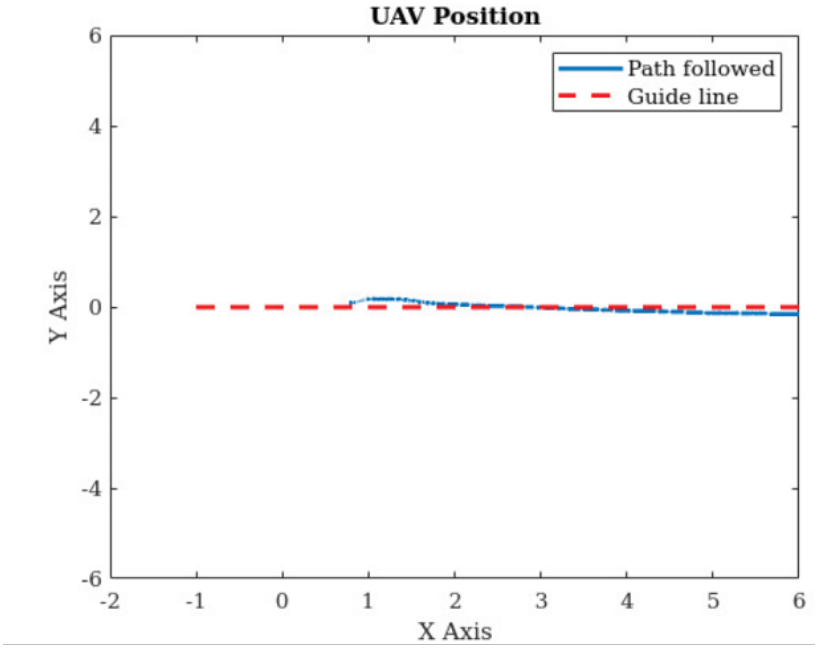
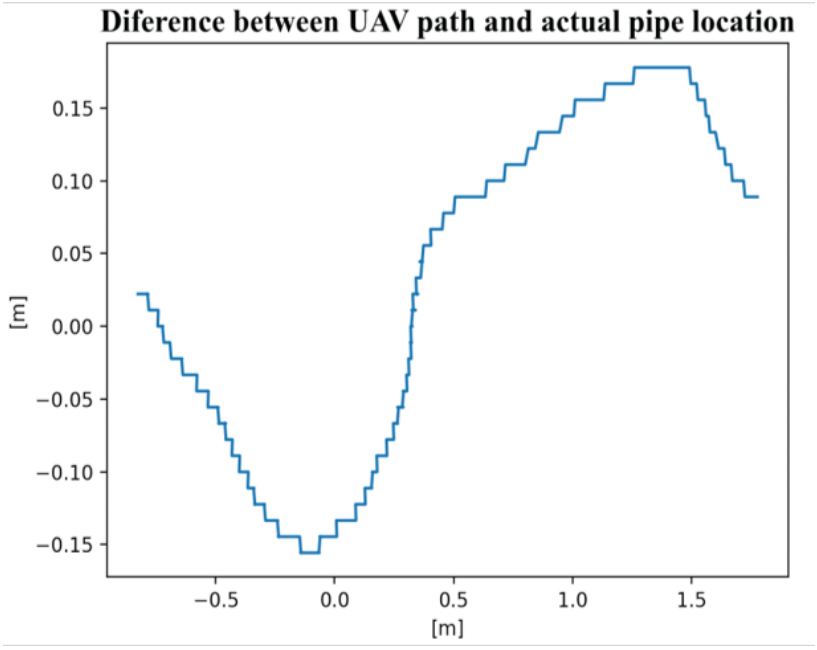


Figure IV.10: UAV performing the pipe tracking in real test. Image adapted from (1).



(a)



(b)

Figure IV.11: UAV positioning during real tests. Image from (1).

Chapter V Conclusions and Future Work

V.1 Conclusions

UAVs have been widely used for industrial inspection in many different fields, because of their versatility and flexibility of flying, reaching places that have high safety risks for humans, such as the O&G areas, work at height, among others.

There is a couple of works using PIG for pipeline inspection, but is not common to see research about this type of UAV inspection as it is in the energy field, which the usage of drone for panel solar, wind turbine and power line inspection is widely using the image processing to perform different inspection activities, putting this work and the article (1) in the highlight of the UAV and CNN inspection strategy.

This work proposed using Hough transform along with YOLO technique to autonomously guide the UAV in unburied pipelines of onshore O&G facilities. The CNN approaches the UAV over the pipe and ensures that the drone will make the curves. The line follower strategy is based on the Hough transform to perform the line tracking during the straight parts, the CNN to guarantee that the UAV is following a tube and to ensure the correct turn where there are curves.

During the development of the research, the author faced many different challenges. Despite this, it is important to mention that the success of this work was not committed, and it generates background for future works.

Starting from the NVIDIA chipset and Ubuntu incompatibility, it is very hard to find a stable GPU driver for Ubuntu 18. In the last test, the computer was very slow because of the driver issue, taking longer than normal to get new results.

During the research conducted in the development of article (1), the CV bridge (it covers the ROS image message to OpenCV array) and the Deep Neural Network (DNN) OpenCV module (it reads the CNN weight file) was running fine. However, after a crash in the Ubuntu system, the author faced difficulties using these features together when it was restored. This is explained by the fact that the DNN module works in a stable version of OpenCV 4, and the CV bridge was discontinued in OpenCV version 3. Thus, the solution to make the system robust was to change the video data acquisition strategy. Instead of reading a ROS message, the GStreamer UDP connection was used to receive the image data and OpenCV 4 in Python 3. OpenCV 4 originally receives data with the GStreamer pipeline. This interface was detailed in Chapter III.

After restoring the Ubuntu 18, the PX4 FCU installed during a flight test had some issues. It was noted that the landing gears were not automatically retracting when Typhoon 480 was in off-board mode. Because of that, a region of interest was implemented in the frame used to run the YOLO to guarantee that the lading gears would not interfere in the YOLO performance. So it was "cut" the 20% in each side of the image before passing through the OpenCV YOLO features.

In the YOLO training, the author did not improve the performance of the ANN besides the several tentatives, as shown in Table IV.1. It could be because of the image quality and resolution, as YOLOv5 expects images with 640x640, and most synthetic images have a resolution of 320x240 and are not good quality. Thus, with the resizing, the feature extracting could be harmed once the best result confidence is the training with the most real pipes, usually bigger than 640x640 and better quality. It is important to mention that the YOLOv5, item one in Table IV.1 published in IEEE Data Port (90), used in this work for testing, has images without the green pipeline, which demonstrates that the training techniques stated in Chapter III are very power full and guarantee a high-performance level for pipe detection.

Although the challenges faced the system performance and the objective of this work were not compromised as the data previously shown. The tests were performed in the Gazebo environment. The outcomes showed that the proposed approach succeeded in accomplishing the mission, and the real test showed the robustness of the proposed system. The UAV flew over the pipeline safely and fully autonomously. The results suggested that the proposed strategy performs excellently in the proposed scenario. Concerning tracking and detection strategies, the outcomes showed that the proposed system could correctly identify pipelines and navigate above them autonomously.

V.2 Future Works

In terms of evaluation, this research opens up several possibilities to develop sophisticated path planning for UAV as pipeline tracking capable of working in global and local environments not restricted to the O&G but also applicable to water treatment facilities and mineral pipelines.

Since neural networks and path planning are fields that the scientific community has widely studied, there is always room for improvement, such as different approaches using the CNN to make UAV moving and other several improvements of the YOLO performance and even other ANN architecture to realize this same task with different performance levels. Related to the Neural Network there is also space to expand the dataset to identify not only the tube but the marks, rust, and other possible failures on the pipe, transforming the current work into a fully automated inspection system.

There is also space for the optimization process to save the battery from improving the area to be covered and motor lifetime, such as the PID controller implemented to analyze the Hough line and bounding box, among other optimal controls techniques that could ensure smooth movements and smarter resource usage.

Bibliography

1 SILVA, Y. M. R. da et al. Computer vision based path following for autonomous unammed aerial systems in unburied pipeline onshore inspection. *Drones*, v. 6, n. 12, 2022. ISSN 2504-446X. Disponível em: <https://www.mdpi.com/2504-446X/6/12/410>. Citado 18 vezes nas páginas x, xi, xii, 23, 26, 32, 34, 36, 39, 41, 42, 43, 44, 46, 48, 49, 50 e 51.

2 SOUSA, L. C. et al. Onshore pipeline inspection using computer vision and unmanned aerial vehicles. In: . [S.l.: s.n.], 2022. Citado 5 vezes nas páginas xii, 28, 39, 40 e 44.

3 PINTO, M. F. et al. Case-based reasoning approach applied to surveillance system using an autonomous unmanned aerial vehicle. In: IEEE. *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*. [S.l.], 2017. p. 1324–1329. Citado na página 14.

4 PINTO, M. F. et al. Ekf design for online trajectory prediction of a moving object detected onboard of a uav. In: IEEE. *2018 13th APCA International Conference on Automatic Control and Soft Computing (CONTROLO)*. [S.l.], 2018. p. 407–412. Citado na página 14.

5 MURRAY, C. C.; RAJ, R. The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. *Transportation Research Part C: Emerging Technologies*, Elsevier, v. 110, p. 368–398, 2020. Citado na página 14.

6 MADRIDANO, Á. et al. 3d trajectory planning method for uavs swarm in building emergencies. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 20, n. 3, p. 642, 2020. Citado na página 14.

7 MELO, A. G. et al. Dynamic optimization and heuristics based online coverage path planning in 3d environment for uavs. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 21, n. 4, p. 1108, 2021. Citado na página 14.

8 MELO, A. G. et al. 3d correspondence and point projection method for structures deformation analysis. *IEEE Access*, IEEE, v. 8, p. 177823–177836, 2020. Citado na página 14.

9 PINTO, M. F. et al. A robotic cognitive architecture for slope and dam inspections. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 20, n. 16, p. 4579, 2020. Citado na página 14.

10 PINTO, M. F. et al. Arcog: An aerial robotics cognitive architecture. *Robotica*, Cambridge University Press, p. 1–20, 2020. Citado na página 14.

11 DIJKSHOORN, N.; VISSER, A. Integrating sensor and motion models to localize an autonomous ar. drone. *International Journal of Micro Air Vehicles*, SAGE Publications Sage UK: London, England, v. 3, n. 4, p. 183–200, 2011. Citado na página 14.

12 TANG, D.; KOU, K.; TANG, Y. Autolanding system of shipborne uav based on fusion of ins, cdgps and vision-based navigation. In: *Advances in Guidance, Navigation and Control*. [S.l.]: Springer, 2022. p. 1609–1618. Citado na página 14.

13 KAKALETSIS, E. et al. Computer vision for autonomous uav flight safety: An overview and a vision-based safe landing pipeline example. *ACM Computing Surveys (CSUR)*, ACM New York, NY, v. 54, n. 9, p. 1–37, 2021. Citado na página 14.

14 COELHO, F. O. et al. Hybrid methodology for path planning and computational vision applied to autonomous mission: A new approach. *Robotica*, Cambridge University Press, v. 38, p. 1000–1018, 2020. Citado na página 14.

15 BIUNDINI, I. Z. et al. Coverage path planning optimization for slopes and dams inspection. In: SPRINGER. *Iberian Robotics conference*. [S.l.], 2019. p. 513–523. Citado na página 14.

16 COELHO, F. O. et al. Ekf and computer vision for mobile robot localization. In: IEEE. *2018 13th APCA International Conference on Automatic Control and Soft Computing (CONTROLO)*. [S.l.], 2018. p. 148–153. Citado na página 14.

17 RAMOS, G. S. et al. Hybrid methodology based on computational vision and sensor fusion for assisting autonomous uav on offshore messenger cable transfer operation. *Robotica*, Cambridge University Press, p. 1–29, 2022. Citado na página 14.

18 KHALOO, A. et al. Utilizing uav and 3d computer vision for visual inspection of a large gravity dam. *Frontiers in Built Environment*, Frontiers, v. 4, p. 31, 2018. Citado na página 14.

19 DAM, J. V.; KRASNE, A.; GABBARD, J. L. Drone-based augmented reality platform for bridge inspection: Effect of ar cue design on visual search tasks. In: IEEE. *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. [S.l.], 2020. p. 201–204. Citado na página 14.

20 KUS, S.; SRINIVASAN, S. Remote, visual inspection and digital analysis for external corrosion characterization in refinery unit applications. In: ONEPETRO. *CORROSION 2021*. [S.l.], 2021. Citado na página 14.

21 NIKOLIC, J. et al. A uav system for inspection of industrial facilities. In: IEEE. *2013 IEEE Aerospace Conference*. [S.l.], 2013. p. 1–8. Citado na página 14.

22 PETILLOT, Y. R. et al. Underwater robots: From remotely operated vehicles to intervention-autonomous underwater vehicles. *IEEE Robotics & Automation Magazine*, IEEE, v. 26, n. 2, p. 94–101, 2019. Citado na página 14.

23 RAMOS, G. et al. Technical and economic feasibility study for implementing a novel mooring-assisting methodology in offloading operations using autonomous unmanned aerial vehicles. *SPE Production & Operations*, OnePetro, v. 37, n. 01, p. 72–87, 2022. Citado na página 14.

24 FREDERIKSEN, M.; KNUDSEN, M. *Drones for Offshore and Maritime Missions: Opportunities and Barriers*. [S.l.]: Center for Integrative Innovation Management, SDU, 2018. Citado na página 15.

25 DURDEVIC, P. et al. Vision aided navigation of a quad-rotor for autonomous wind-farm inspection. *IFAC-PapersOnLine*, Elsevier, v. 52, n. 8, p. 61–66, 2019. Citado na página 15.

26 YU, L. et al. Inspection robots in oil and gas industry: a review of current solutions and future trends. In: IEEE. *2019 25th International Conference on Automation and Computing (ICAC)*. [S.l.], 2019. p. 1–6. Citado 2 vezes nas páginas 15 e 17.

27 TSOUROS, D. C.; BIBI, S.; SARIGIANNIDIS, P. G. A review on uav-based applications for precision agriculture. *Information*, Multidisciplinary Digital Publishing Institute, v. 10, n. 11, p. 349, 2019. Citado na página 17.

28 BIUNDINI, I. Z. et al. Coverage path planning optimization based on point cloud for structural inspection. In: *Frontiers in Nature-Inspired Industrial Optimization*. [S.l.]: Springer, 2022. p. 141–156. Citado na página 17.

29 WANG, C.; CUI, L. The implementation of automatic inspection algorithm for underwater vehicles based on hough transform. In: ATLANTIS PRESS. *2018 7th International Conference on Sustainable Energy and Environment Engineering (ICSEEE 2018)*. [S.l.], 2019. p. 459–464. Citado 2 vezes nas páginas 17 e 19.

30 MAZREAH, A. A.; ALNAIMI, F. B. I.; SAHARI, K. S. M. Novel design for pig to eliminate the effect of hydraulic transients in oil and gas pipelines. *Journal of Petroleum Science and Engineering*, Elsevier, v. 156, p. 250–257, 2017. Citado 2 vezes nas páginas 17 e 19.

31 KAKOGAWA, A.; MA, S. Design of a multilink-articulated wheeled pipeline inspection robot using only passive elastic joints. *Advanced Robotics*, Taylor & Francis, v. 32, n. 1, p. 37–50, 2018. Citado 2 vezes nas páginas 17 e 19.

32 KWON, Y.-S.; YI, B.-J. Design and motion planning of a two-module collaborative indoor pipeline inspection robot. *IEEE Transactions on Robotics*, IEEE, v. 28, n. 3, p. 681–696, 2012. Citado na página 17.

33 IWASZENKO, S. et al. Detection of natural gas leakages using a laser-based methane sensor and uav. *Remote Sensing*, Multidisciplinary Digital Publishing Institute, v. 13, n. 3, p. 510, 2021. Citado na página 17.

34 GÓMEZ, C.; GREEN, D. R. Small unmanned airborne systems to support oil and gas pipeline monitoring and mapping. *Arabian Journal of Geosciences*, Springer, v. 10, n. 9, p. 1–17, 2017. Citado 2 vezes nas páginas 17 e 18.

35 BRETSCHNEIDER, T. R.; SHETTI, K. Uav-based gas pipeline leak detection. In: *Proc. of ARCS*. [S.l.: s.n.], 2015. Citado na página 17.

36 SHUKLA, A.; XIAOQIAN, H.; KARKI, H. Autonomous tracking and navigation controller for an unmanned aerial vehicle based on visual data for inspection of oil and gas pipelines. In: IEEE. *2016 16th international conference on control, automation and systems (ICCAS)*. [S.l.], 2016. p. 194–200. Citado na página 18.

37 YAN, Y. et al. A two-stage optimization method for unmanned aerial vehicle inspection of an oil and gas pipeline network. *Petroleum Science*, Springer, v. 16, n. 2, p. 458–468, 2019. Citado na página 18.

38 MANGAYARKARASI, N.; RAGHURAMAN, G.; KAVITHA, S. Influence of computer vision and iot for pipeline inspection-a review. In: IEEE. *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)*. [S.l.], 2019. p. 1–6. Citado na página 18.

39 MOTAMEDI, M.; FARAMARZI, F.; DURAN, O. New concept for corrosion inspection of urban pipeline networks by digital image processing. In: IEEE. *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*. [S.l.], 2012. p. 1551–1556. Citado na página 18.

40 BONDADA, V.; PRATIHAR, D. K.; KUMAR, C. S. Detection and quantitative assessment of corrosion on pipelines through image analysis. *Procedia Computer Science*, Elsevier, v. 133, p. 804–811, 2018. Citado na página 18.

41 KIRUBAKARAN, A. P.; KRISHNA, I. M. Pipeline crack detection using mathematical morphological operator. In: *Knowledge Computing and its Applications*. [S.l.]: Springer, 2018. p. 29–46. Citado na página 18.

42 SU, T.-C.; YANG, M.-D. Application of morphological segmentation to leaking defect detection in sewer pipelines. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 14, n. 5, p. 8686–8704, 2014. Citado na página 18.

43 SINHA, S. K.; FIEGUTH, P. W. Morphological segmentation and classification of underground pipe images. *Machine vision and applications*, Springer, v. 17, n. 1, p. 21–31, 2006. Citado na página 18.

44 SINHA, S. K.; FIEGUTH, P. W. Segmentation of buried concrete pipe images. *Automation in Construction*, Elsevier, v. 15, n. 1, p. 47–57, 2006. Citado na página 18.

45 SU, T.-C. Segmentation of crack and open joint in sewer pipelines based on cctv inspection images. In: *Proc. AASRI Int. Conf. on Circuits and Systems*. [S.l.: s.n.], 2015. v. 2. Citado na página 18.

46 TING, L. L. et al. Water leak location based on improved dual-tree complex wavelet transform with soft thresholding de-noising. *Applied Acoustics*, Elsevier, v. 174, p. 107751, 2021. Citado na página 18.

47 HAWARI, A. et al. Automated defect detection tool for closed circuit television (cctv) inspected sewer pipelines. *Automation in Construction*, Elsevier, v. 89, p. 99–109, 2018. Citado na página 18.

48 KUMAR, S. S. et al. Automated defect classification in sewer closed circuit television inspections using deep convolutional neural networks. *Automation in Construction*, Elsevier, v. 91, p. 273–283, 2018. Citado na página 18.

49 YIN, X. et al. A deep learning-based framework for an automated defect detection system for sewer pipes. *Automation in construction*, Elsevier, v. 109, p. 102967, 2020. Citado na página 18.

50 XIAOQIAN, H. et al. Variant pid controller design for autonomous visual tracking of oil and gas pipelines via an unmanned aerial vehicle. In: IEEE. *2017 17th International Conference on Control, Automation and Systems (ICCAS)*. [S.l.], 2017. p. 368–372. Citado na página 18.

51 BASSO, M.; FREITAS, E. Pignaton de. A uav guidance system using crop row detection and line follower algorithms. *Journal of Intelligent & Robotic Systems*, Springer, v. 97, n. 3, p. 605–621, 2020. Citado na página 19.

52 OKOLI, J.; UBOCHI, B. Autonomous robot for gas pipeline inspection and leak detection. *International Journal of Computing and Digital Systems*, University of Bahrain, v. 11, n. 1, p. 811–820, 2022. Citado na página 19.

53 MARIA, T. H. S. et al. Nmpc controller applied to an uav path following problem. In: IEEE. *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*. [S.l.], 2020. p. 1–6. Citado na página 19.

54 ANDRADE, F. A. D. A. et al. Virtual reality simulation of autonomous solar plants inspections with unmanned aerial systems. In: *2021 Aerial Robotic Systems Physically Interacting with the Environment (AIRPHARO)*. [S.l.: s.n.], 2021. p. 1–8. Citado na página 19.

55 WIKIROS. *science article*. 2007. http://wiki.ros.org/. Acessado em 25/06/2022. Citado na página 19.

56 BRADSKI, G. The opencv library. *Dr. Dobb's Journal of Software Tools*, 2000. Citado na página 20.

57 XIE, G.; LU, W. Image edge detection based on opencv. *International Journal of Electronics and Electrical Engineering*, v. 1, p. 104–106, 2013. Citado na página 20.

58 CANNY, J. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8, n. 6, p. 679–698, 1986. Citado na página 20.

59 KIRYATI, N.; ELDAR, Y.; BRUCKSTEIN, A. A probabilistic hough transform. *Pattern Recognition*, v. 24, n. 4, p. 303–316, 1991. ISSN 0031-3203. Disponível em: <https://www.sciencedirect.com-/science/article/pii/003132039190073E>. Citado 2 vezes nas páginas 22 e 34.

60 RATHORE, A. S. Lane detection for autonomous vehicles using opencv library. *International Research Journal of Engineering and Technology*, v. 6, n. 1, p. 1326–1332, 2019. Citado na página 22.

61 ROSSI, A. et al. Real-time lane detection and motion planning in raspberry pi and arduino for an autonomous vehicle prototype. *CoRR*, abs/2009.09391, 2020. Disponível em: <https://arxiv.org/abs-/2009.09391>. Citado 2 vezes nas páginas 22 e 35.

62 DING, D.; LEE, C.; LEE, K.-Y. An adaptive road roi determination algorithm for lane detection. In: *2013 IEEE International Conference of IEEE Region 10 (TENCON 2013)*. [S.l.: s.n.], 2013. p. 1–4. Citado na página 22.

63 BRANDAO, A. S.; MARTINS, F. N.; SONEGUETTI, H. B. A vision-based line following strategy for an autonomous uav. In: IEEE. *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*. [S.l.], 2015. v. 2, p. 314–319. Citado na página 22.

64 MESTER, G. Modeling of autonomous hexa-rotor microcopter. In: . [S.l.: s.n.], 2015. Citado na página 23.

65 SULTANA, F.; SUFIAN, A.; DUTTA, P. A review of object detection models based on convolutional neural network. In: _____. *Intelligent Computing: Image Processing Based Applications*. Singapore: Springer Singapore, 2020. p. 1–16. ISBN 978-981-15-4288-6. Disponível em: <https://doi-.org/10.1007/978-981-15-4288-6_1>. Citado 2 vezes nas páginas 24 e 32.

66 VALUEVA, M. et al. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*, v. 177, p. 232–243, 2020. ISSN 0378-4754. Disponível em: <https://www.sciencedirect.com/science/article/pii-/S0378475420301580>. Citado na página 24.

67 RAZAVIAN, A. S. et al. Cnn features off-the-shelf: An astounding baseline for recognition. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*. [S.l.: s.n.], 2014. p. 512–519. Citado na página 24.

68 KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012. v. 25. Disponível em: <https://proceedings.neurips.cc/paper-/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>. Citado na página 24.

69 LIU, W. et al. SSD: single shot multibox detector. In: . [s.n.], 2015. abs/1512.02325. Disponível em: <http://arxiv.org/abs/1512.02325>. Citado na página 24.

70 COLLOBERT, R.; WESTON, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In: *Proceedings of the 25th international conference on Machine learning*. [S.l.: s.n.], 2008. p. 160–167. Citado na página 24.

71 OORD, A. Van den; DIELEMAN, S.; SCHRAUWEN, B. Deep content-based music recommendation. *Advances in neural information processing systems*, v. 26, 2013. Citado na página 24.

72 KE, Q. et al. Chapter 5 - computer vision for humanmachine interaction. In: LEO, M.; FARINELLA, G. M. (Ed.). *Computer Vision for Assistive Healthcare*. Academic Press, 2018, (Computer Vision and Pattern Recognition). p. 127–145. ISBN 978-0-12-813445-0. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780128134450000058>. Citado na página 24.

73 BANERJEE, A. et al. Chapter 3 - impacts of metaheuristic and swarm intelligence approach in optimization. In: MISHRA, S. et al. (Ed.). *Cognitive Big Data Intelligence with a Metaheuristic Approach*. Academic Press, 2022, (Cognitive Data Science in Sustainable Computing). p. 71–99. ISBN 978-0-323-85117-6. Disponível em: <https://www.sciencedirect.com/science/article/pii/B978032385117600008X>. Citado na página 24.

74 O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015. Citado na página 24.

75 J, L.; X, W. Tomato diseases and pests detection based on improved yolo v3 convolutional neural network. In: *Frontiers in Plant Science*. [S.l.: s.n.], 2020. v. 11. Citado na página 25.

76 REDMON, J. et al. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 779–788. Citado 2 vezes nas páginas 25 e 32.

77 SARDA, A.; DIXIT, S.; BHAN, A. Object detection for autonomous driving using yolo algorithm. In: *2021 2nd International Conference on Intelligent Engineering and Management (ICIEM)*. [S.l.: s.n.], 2021. p. 447–451. Citado na página 25.

78 DEVS, Y. *YOLO: Real-Time Object Detection*. https://pjreddie.com/darknet/yolo/. Accessed on 07/31/2022. Citado na página 25.

79 ARIE, L. G. *The practical guide for Object Detection with YOLOv5 algorithm ="https://rb.gy/w9lqpe*. [S.l.]: Towards Data Science. Accessed on 11/2022. Citado 3 vezes nas páginas 25, 32 e 43.

80 KIM, J.-H. et al. Object detection and classification based on yolo-v5 with improved maritime dataset. *Journal of Marine Science and Engineering*, v. 10, n. 3, 2022. ISSN 2077-1312. Disponível em: <https://www.mdpi.com/2077-1312/10/3/377>. Citado na página 26.

81 SZEGEDY, C. et al. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. Disponível em: <http://arxiv.org/abs/1409.4842>. Citado na página 26.

82 PADILLA, R.; NETTO, S. L.; SILVA, E. A. B. da. A survey on performance metrics for object-detection algorithms. In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. [S.l.: s.n.], 2020. p. 237–242. Citado na página 26.

83 DING, L.; GOSHTASBY, A. On the canny edge detector. *Pattern recognition*, Elsevier, v. 34, n. 3, p. 721–725, 2001. Citado 2 vezes nas páginas 28 e 32.

84 DUDA, R. O.; HART, P. E. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, ACM New York, NY, USA, v. 15, n. 1, p. 11–15, 1972. Citado 2 vezes nas páginas 28 e 33.

85 GONçALVES, J.; PINTO, V. H.; COSTA, P. A line follower educational mobile robot performance robustness increase using a competition as benchmark. In: *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*. [S.l.: s.n.], 2019. p. 934–939. Citado na página 28.

86 ABBYASOV, B. et al. Automatic tool for gazebo world construction: from a grayscale image to a 3d solid model. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.: s.n.], 2020. p. 7226–7232. Citado na página 30.

87 CODE, D. *Joystick Setup*. https://docs.qgroundcontrol.com/master/en/SetupView/Joystick.html. Access on 04/17/2022. Citado na página 31.

88 ULTRALYTICS. *Ultralytics YOLOv5 Github*. https://github.com/ultralytics/yolov5. Accessed on 11/2022. Citado na página 32.

89 MU, Z.; LI, Z. Intelligent tracking car path planning based on hough transform and improved pid algorithm. In: IEEE. *2018 5th International conference on systems and informatics (ICSAI)*. [S.l.], 2018. p. 24–28. Citado na página 33.

90 SILVA, Y. et al. *Pipeline Recognition for Drone Navigation*. IEEE Dataport, 2022. Disponível em: <https://dx.doi.org/10.21227/pncz-sv78>. Citado 4 vezes nas páginas 39, 41, 42 e 52.

91 SILVA, Y. et al. *Pipeline Tracking Dataset*. Roboflow, 2022. Disponível em: <https://universe-.roboflow.com/cefetrj/pipe-tracking-640>. Citado na página 39.

92 SILVA, Y. et al. *Pipeline Merge Dataset*. Roboflow, 2022. Disponível em: <https://universe-.roboflow.com/cefetrj/pipeline-tracking-merge>. Citado na página 39.

93 RAGHUNATHAN, S. *Line Follower TurtleBot*. 2022. https://github.com/sudrag/line_follower_turtlebot. Acessado em 02/04/2022. Citado na página 39.

94 ROS, W. *SolidWorks to URDF Exportern*. https://tinyurl.com/ycwns4tp. Accessed on 07/31/2022. Citado na página 40.

95 BISONG, E. *Building machine learning and deep learning models on Google cloud platform: A comprehensive guide for beginners*. [S.l.]: Apress, 2019. Citado na página 42.