COMPUTER VISION METHODS FOR UNDERWATER PIPELINE SEGMENTATION

Roberto Esteban Campos Ruiz

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientadores: Mariane Rembold Petraglia
José Gabriel Rodríguez Carneiro Gomes

Rio de Janeiro
Março de 2018

COMPUTER VISION METHODS FOR UNDERWATER PIPELINE SEGMENTATION

Roberto Esteban Campos Ruiz

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. Mariane Rembold Petraglia, Ph.D.

Prof. Julio Cesar Boscher Torres, D.Sc.

Prof. Diego Barreto Haddad, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2018

Ruiz, Roberto Esteban Campos

Computer vision methods for underwater pipeline segmentation/Roberto Esteban Campos Ruiz. – Rio de Janeiro: UFRJ/COPPE, 2018.

XII, 43 p.: il.; 29, 7cm.

Orientadores: Mariane Rembold Petraglia

José Gabriel Rodríguez Carneiro Gomes

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2018.

Referências Bibliográficas: p. 38 – 41.

1. Underwater pipeline. 2. Object detection. 3. Image segmentation. I. Petraglia, Mariane Rembold *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Pedro and Rosita, because
everything I am, I owe to you.

Acknowledgment

First, thanks to God, for giving me health and for guiding me through the right path all this time.

Thanks to my family, because this achievement is also yours. And because despite the distance, I always count on you.

Thanks to my advisers: prof. Mariane and prof. José Gabriel for the help and the trust they gave me, for all the teachings and all the patience they had for me.

Thanks to my friends, for the experiences lived and the shared moments, not only academically but also personally. It would not be fair to name you, because each one of you are just as important as the others.

Thanks to all the institutions that gave me the opportunity and facilities to complete this work. The Analog and Digital Signal Processing Laboratory (PADS), the Electrical Engineering Program (PEE) and the Federal University of Rio de Janeiro. And also to the "Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES)" for the financial support.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

MÉTODOS DE VISÃO COMPUTACIONAL PARA SEGMENTAÇÃO DE DUTOS SUBMARINOS

Roberto Esteban Campos Ruiz

Março/2018

Orientadores: Mariane Rembold Petraglia
José Gabriel Rodríguez Carneiro Gomes

Programa: Engenharia Elétrica

O processo de inspeção de tubulações submarinas é geralmente realizado por Veículos Operados Remotamente (ROVs) equipados principalmente com sensores óticos e acústicos. Durante longos periodos de inspeção e em condições de baixa visibilidade, o processo de inspeção visual torna-se cansativo e sujeito a falhas de interpretação por parte do operador. Portanto, a automação desse processo apresenta uma melhoria na manutenção das tubulações.

Este trabalho apresenta um sistema de segmentação de tubulações rígidas submarinas usando uma câmera monocular. Um detector de bordas baseado na cor foi proposto aproveitando as restrições da geometria das tubulações e informações de rastreamento. Tubulações segmentadas foram transformadas em uma representação de vista superior 2D. O sistema foi avaliado com um conjunto de dados de 7808 imagens, anotados manualmente, obtidas em diferentes tarefas de inspeção reais. O sistema obteve 96.5% na taxa de detecção e 96,3% de acurácia na segmentação.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

COMPUTER VISION METHODS FOR UNDERWATER PIPELINE SEGMENTATION

Roberto Esteban Campos Ruiz

March/2018

Advisors: Mariane Rembold Petraglia
José Gabriel Rodríguez Carneiro Gomes

Department: Electrical Engineering

Underwater pipeline inspection is usually conducted by Remotely Operated Vehicles (ROVs) equipped mainly with optical and acoustic sensors. During long inspections periods and low visibility conditions, traditional visual inspection becomes a tedious job and can lead to operator misinterpretations. Therefore, the automation of this process involves an improvement in the maintenance of the pipelines.

This work presents an underwater pipeline segmentation system for rigid pipelines using a monocular camera. A color based edge detector was proposed, taking advantage of the pipeline geometry restrictions, besides tracking information. Segmented pipelines were transformed into a 2D top view representation. The system was evaluated with a dataset containing 7808 images, manually annotated, acquired during real inspection tasks. The system reached 96.5% of detection rate and 96.3% of segmentation accuracy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

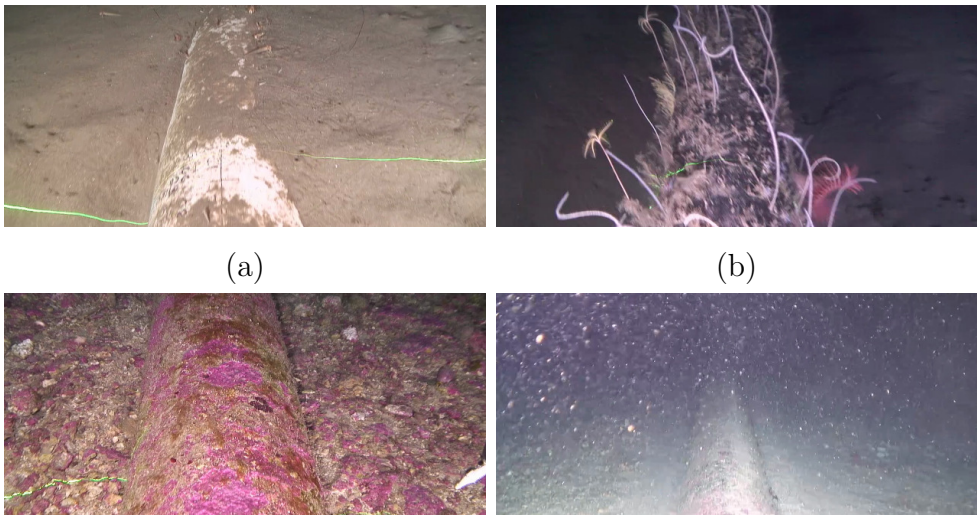
Underwater operations are carried out using Remotely Operated Vehicles (ROVs), which are controlled remotely by an operator located on a ship, and equipped mainly with optical and acoustic sensors [2]. They operate under the sea surface performing inspection, search, recovery, repair and maintenance operations of man-made structures and oceanographic and engineering research [3]. On the particular case of underwater pipeline inspection, ROVs move along the pipeline acquiring information to determine the overall condition and ensure the conservation of these structures.

Umbilical/tether cables, used for energy supply, signal transmission and communication [4], connect the ROV to the control unit. Signals are transmitted from sensors that help to keep control of the vehicle and from sensors installed for inspection. Cameras are the principal elements to provide vehicle orientation and position feedback with respect to the pipeline, and also to perform visual inspection. In low visibility conditions and long range locations, multibeam and sidescan sonars are used to produce acoustic images and a map of the surroundings. Magneto-metric sensors are used to detect internal and external changes on a pipeline by analyzing its cross section [5]. In a similar way to multibeam sonar, laser imaging system can be used to produce 3D high resolution images [5].

Traditionally, the detection of pipeline damages (corrosion, cracks, biological contamination and leaks) is performed through visual inspection by a specialized person, at the same time of the mission or later using the recorded information. In both cases, the extension of this operation for a long time represents a tedious job and can lead to an erroneous analysis due to the lack of the operator concentration. Therefore, the automation of this process entails an improvement in the maintenance of these structures.

In visual inspection, well defined features presented in recently installed pipelines, such as color and shape, facilitate the detection process. When the pipelines get older, the influence of marine environment becomes visible. Sometimes, the sand covers the pipeline or algae growth may deform the pipeline contours or makes the

pipeline apparency similar to the seabed appearance. Non-uniform artificial lighting, blurring and turbulence produced by the lack of stability on the ROV movement [2] make the inspection processes more complicated. The described situations are shown in Figure 1.1.



(a) (b)

(c) (d)

Figure 1.1: Sample pipeline images. Sand (a), algae (b), seabed appearance (c) and non-uniform lighting (d).

This work presents a system for the detection and segmentation of underwater pipelines - using image processing and computer vision algorithms - as the first step in the process of developing an automatic inspection system where traditional methods do not work properly. Image sequences acquired during ROV inspection missions are used as input to the system. As an output, for each image, the segmented pipeline is obtained.

1.1 Objectives

The main objective of this work is to develop a monocular vision-based system for underwater pipeline segmentation with application in real scenarios. To accomplish this, some specific objectives are proposed:

- Generate a manually segmented dataset from real image sequences to evaluate the system performance;
- Include the color information from the input image to improve the pipeline detection process;
- Develop an automatic initializer that is activated at the beginning of the tracking and every time the tracking is lost.

- Compare different matching strategies and number of frames used for tracking;
- Transform the segmented pipeline into a 2D top-view representation.

1.2 Text organization

The proposed method for pipeline segmentation is described in Chapter 3 as well as the theoretical background of the algorithms used. Chapter 4 presents the results of experiments with real image sequences and in Chapter 5, concluding remarks and some hints for possible improvements are given. Finally, in Appendix A, information about the database is provided.

1.3 Related work

Many systems have been used in underwater pipeline/cable operations. Acoustic system composed by forward looking sonar [6], side scan sonar [7] or echo sounders [8] were used for pipeline detection from long distances and challenging visual conditions. A magnetic system was used in [9] to detect narrowed cables, by applying electrical stimulation to the cable and detecting it by magnetometers situated in an AUV (Autonomous Underwater Vehicle). Vision systems based on cameras are used in good visibility conditions and closeness to the pipeline [2], [10], [11] [12], [13], [14]. Finally, some systems combine acoustic and visual information [15] or side scanning sonar, sub-bottom profiler and a magnetometer [16], to increase their robustness and the results reliability.

Since most of the systems developed for the visual inspection of underwater pipelines are carried out by private companies, not much of work has been published in the literature. Then, the most relevant vision-based approaches are reviewed below.

In [11], Hough Transform (HT) is applied after edge detection, the two longest and parallel lines are selected, distance from the pipeline to ROV is calculated using calibration parameters from the camera and pipeline width. That information is used to validate detection on future frames. The pipeline is expected to be parallel to the camera plane and a previous knowledge of the target width is required. Tests were conducted in a non-marine controlled environment.

To avoid selecting a fixed threshold for edge detection, [13] proposed a histogram clustering of the gradient magnitude with three classes: background, pipeline edges and ripple patterns. Edges with more intensity are assumed to correspond to the pipeline borders and once thresholding was applied, the two longest lines obtained

from HT are selected as pipeline borders. Then, optical flow is used to predict the pipeline location over a Region of Interest (ROI) for the next frame.

The method presented in [2] is based on image segmentation followed by a straight line segment fitting and co-linearity analysis. Segmentation is done using the minimal-spanning tree algorithm over a bidimensional histogram (gray level intensity vs. gradient magnitude) using three classes. Then, contours are calculated and straight lines are found by grouping adjacent pixels according to a preference matrix (vertical lines have preference over horizontal). Next, using *total least squares* [17], a line is fitted to the pixels corresponding to each segment. Co-linearity analysis is applied to merge segments that are possible parts of the same pipeline border. A segment is taken as reference, and all segments that are close and have similar orientation to the reference one are merged. The same criteria applied in [11] is used to select the final borders. A Kalman Filter (KF) is used to predict a ROI, and when there is no valid detection the system is reset and the ROI is set to the whole image. This system was tested with a real inspection dataset containing 1457 images. The system achieved a 92% average detection rate.

A particle filter based system for underwater telecommunications cable tracking was presented in [12]. Unlike other methods and due to the reduced cable diameter, only the distance and orientation of the central line with respect to the image origin was used to represent the cable. For these parameters, a constant velocity model was adopted considering a slow vehicle motion in a 2D plane parallel to the ground. The observation model was built from the convolution of a 1D *mexican-hat* function and the rows of the input image, where peaks in the resulting signal represent a cable location hypothesis. Results showed an average error of 3.11 pixels and 1.83° for cable distance and angle, respectively, over a dataset composed by 10000 manually annotated images.

Shape and color image segmentation were employed in [10] for pipeline detection, where shape information was obtained by applying a canny edge detector on the grayscale input image. Color segmentation is obtained by thresholding the Hue, Saturation and Value channels (HSV). Shape and color information are merged by a pixelwise AND operation, an adjustment mechanism taken from [18] and the pipeline parameters from previous frames. Finally the central line is estimated as in [12], and the new thresholds are set for the next frames.

In [14], a visual classification system based in a Hierarchical Neural-Tree Classifier (HNTC) for pipeline recognition is presented. The input image is divided into small regions called "macro-pixels" from where features are extracted. Then, each macro-pixel classification is validated by analyzing their neighborhood classification. If the mean result of the surrounding macro-pixels is less than a lower threshold, then the macro-pixel corresponds to the sea-bottom. If the mean result is greater

than an upper threshold, then the macro-pixel corresponds to the pipeline. When the mean result is between the lower and upper thresholds, an uncertain pixel mechanism is activated. The mechanism divides the macro-pixel into smaller regions and repeats the process. Results show the system performance in real scenarios where the pipeline shape is well defined.

Chapter 2

System architecture and description

The proposed system for solving the pipeline segmentation problem is presented in this chapter. Fig. 2.1 shows the system architecture. It is composed by five blocks: image conditioning, pipeline detection, initialization, tracking and segmentation and 2D top view transformation.

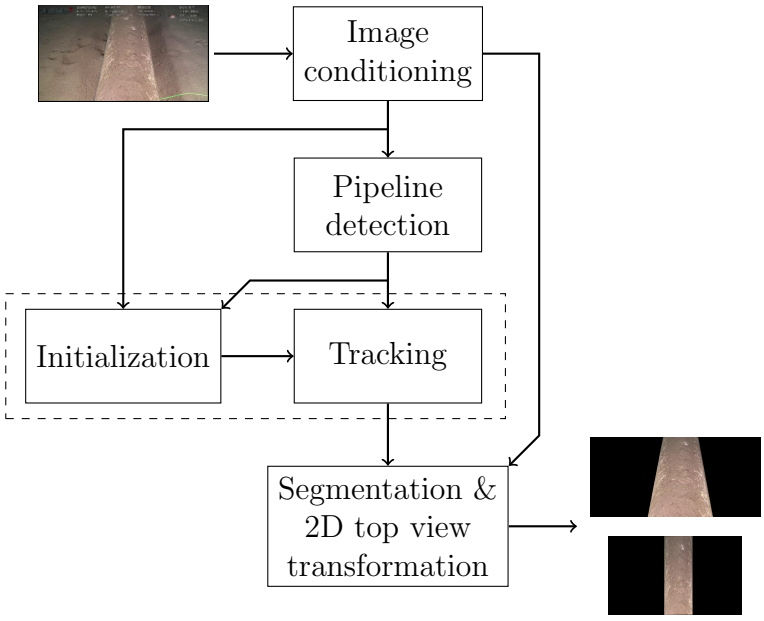


Figure 2.1: System architecture.

The input image, $\mathbf{I}$, passes through the image conditioning block where basic image processing operations are performed, generating $\mathbf{I_1}$ and $\mathbf{I_2}$. $\mathbf{I_2}$ goes to the pipeline detection block. Here, a set of hypotheses that represent possible pipeline locations is generated. $\mathbf{I_2}$ also goes to the initialization block together with the hypothesis set. In this block a more complex process for pipeline detection is performed. This block only works at the beginning of the tracking. The tracking block

uses information from previous detections to facilitate the selection of an hypothesis from the current set. Segmentation and 2D top view transformation block uses the information of the detected pipeline to isolate it from its environment and to generate an image where the pipeline edges are vertical and parallel. $\mathbf{I_1}$ is used for these operations.

2.1 Image conditioning

Three operations are performed in this process as illustrated in Fig. 2.2. First $\mathbf{I_0}$ is generated from $\mathbf{I}$ by applying cropping, as shown in Figure 2.3, to remove alphanumeric information corresponding to the inspection process. Then, $\mathbf{I_1}$ and $\mathbf{I_2}$ are generated by resizing $\mathbf{I_0}$ by a factor of 0.5 and 0.125, respectively. The images $\mathbf{I_2}$ will be used during the detection task, and $\mathbf{I_1}$ will be used for segmentation and top view representation. Finally, a noise reduction operation is applied over $\mathbf{I_2}$.

Commonly used noise filtering approaches tend to smooth characteristics of interest such as borders, making the detection process more complicated. Bilateral filter was used iteratively, because of its edge-preserving feature.

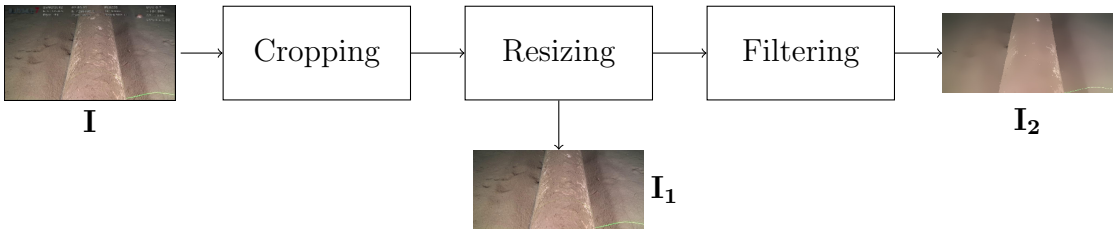


Figure 2.2: Image conditioning process.

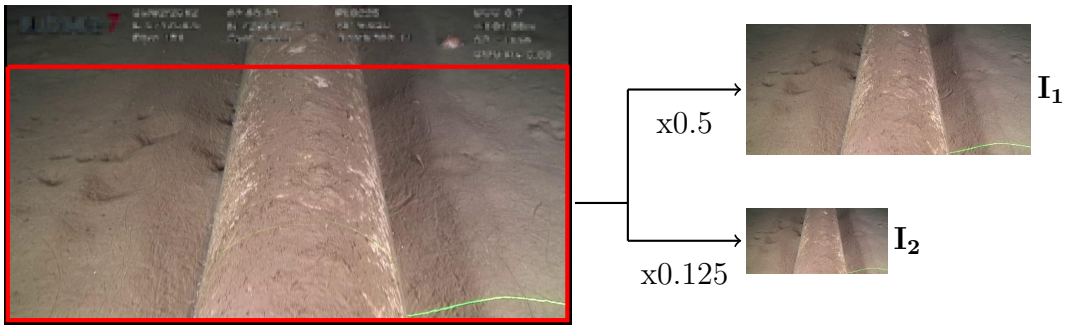


Figure 2.3: Input images $\mathbf{I}$ and $\mathbf{I_0}$ (in red) from where $\mathbf{I_1}$ and $\mathbf{I_2}$ are extracted.

Bilateral filter

Bilateral filter is a non-linear edge preserving technique to smooth images [19], and has been used in image denoising [20], multispectral fusion [21], tone mapping [22]

and mesh smoothing [23]. The algorithm output is a weighted average of the interest pixel neighbors. To preserve edges, the weighted average takes into account pixel intensity variations. If two pixel have similar spatial location and similar intensity levels, they are considered close to each other.

The bilateral filter, $BF[\cdot]$, applied to the pixel $\mathbf{p}$ from image $\mathbf{I}$ is defined as:

$$BF[\mathbf{I_p}] = \frac{1}{W_p} \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) G_{\sigma_r}(|\mathbf{I_p} - \mathbf{I_q}|) \mathbf{I_q}, \tag{2.1}$$

where the normalization factors W_p and $G_\sigma(x)$ are given, respectively, by:

$$W_p = \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) G_{\sigma_r}(|\mathbf{I_p} - \mathbf{I_q}|), \tag{2.2}$$

and

$$G_\sigma(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right). \tag{2.3}$$

In Eq. (2.1), the spatial Gaussian weighing G_{σ_s} and the Gaussian range weighing G_{σ_r} penalize the influence of distant pixels and the difference in intensity values between $\mathbf{I_p}$ and $\mathbf{I_q}$, respectively. The effect of the spatial σ_s and range σ_r parameters are shown in Figure 2.4.

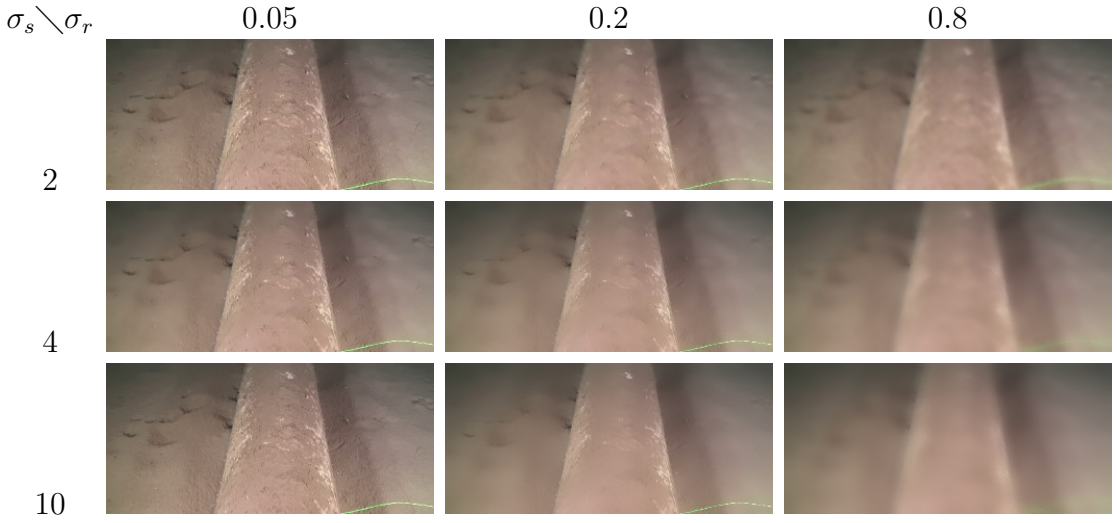


Figure 2.4: Effect of σ_s and σ_r parameters. There is no smoothing when σ_r is close to 0 independent of σ_s. When σ_r and σ_s are increased, the output effect passes from an edge preserving smoothing to blurring.

A different effect from modifying σ_s and σ_r parameters can be reached by iterating the bilateral filter. This leads to smoother images as shown in Fig. 2.5. Strong edges remain when the bilateral filter is iterated, in comparison to the blurring effect generated when σ_r and σ_s are increased.

A more detailed explanation about bilateral filtering can be found in [24].

1 iteration 2 iteration 4 iteration

Figure 2.5: Bilateral filter iterations with $\sigma_s = 6$ and $\sigma_r = 0.1$.

2.2 Pipeline detection

The pipeline detection consist of extracting all possible locations of the pipeline from $\mathbf{I_2}$. The operations required for this task are shown in Fig. 2.6, and explained next.

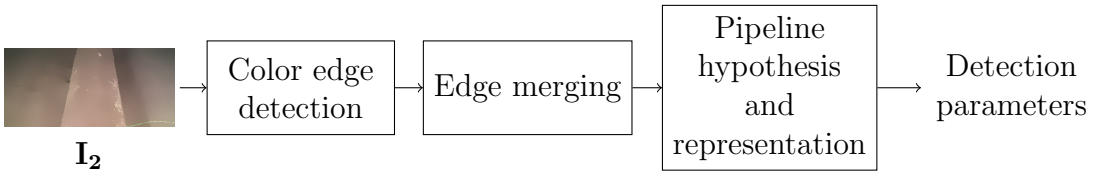


Figure 2.6: Pipeline detection process.

2.2.1 Color edge detection

The proposed color edge detection process is composed by three operations as it is shown in Fig. 2.7. A color edge detection algorithm based on the color tensor of Eq. (2.5) was proposed in [1]. That algorithm applies non-maximum suppression on λ_1, which is shown in Eq. (2.7), along the local prominent direction θ.

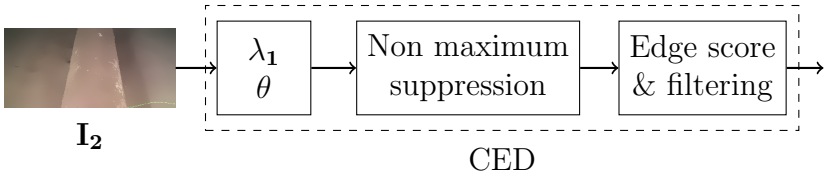


Figure 2.7: Color edge detection (CED).

A variation is proposed by setting a fixed $\theta = \frac{\pi}{2}$, since we assume that the ROV motion is realized along the pipeline during inspection, which means that pipeline edges in $\mathbf{I_2}$ will always appear vertically.

$\mathbf{I_{nms}}$ is obtained after computing λ_1 from $\mathbf{I_2}$, followed by non-maximum suppression. Then, each edge i in $\mathbf{I_{nms}}$ is labeled and its points are stored in the vector $\mathbf{l_i}$. Then, a score s_i is computed:

$$s_i = n_i \sum \mathbf{I_{nms}}(\mathbf{l_i}), \tag{2.4}$$

where n_i is the length of $\mathbf{l_i}$ and $\sum \mathbf{I_{nms}}(\mathbf{l_i})$ is the energy contained in $\mathbf{l_i}$. Higher values of this score indicate the presence of a long and well defined edge.

Edges with $n_i \leq \tau_1$ and $s_i \leq \frac{1}{n_s}\sum s$ are eliminated (n_s is the length of the vector of scores s), because they are not expected to be part of the pipeline edges. Finally, the labels are re-indexed as $\mathbf{l_j}$, taking into consideration the ones eliminated in the previous step. An example of the result obtained with the proposed color edge detector is shown in Figure 2.8. Notice that, by making θ fixed, the pipeline orientation is taken into consideration, resulting in better edge detection.

Figure 2.8: Color edge detection. From left to right: input image, color edge detector [1], color edge detector [1] combined with edge score, proposed color edge detector with $\theta = \frac{\pi}{2}$. All pixel values > 0 were set to 1 for visualization.

Color tensor

Proposed in [25], color tensor is applied to color gradient computation. In this work is stated that, for color images, a simple sum of the image derivatives ignores the correlation between the color channels. Color tensor has been used in texture analysis [26], feature extraction [27], [1] and segmentation [28].

The color structure tensor $\mathbf{S}$ of an image $\mathbf{I} = (\mathbf{R}, \mathbf{G}, \mathbf{B})^T$ is given by:

$$\mathbf{S} = \begin{pmatrix} \overline{\mathbf{S_{xx}}} & \overline{\mathbf{S_{xy}}} \\ \overline{\mathbf{S_{xy}}} & \overline{\mathbf{S_{yy}}} \end{pmatrix}, \tag{2.5}$$

where $\overline{\mathbf{S}}_{(\cdot)}$ indicates a convolution of the spatial derivatives with a Gaussian filter. And

$$\begin{aligned} \mathbf{S_{xx}} &= \mathbf{R_x^2} + \mathbf{G_x^2} + \mathbf{B_x^2}, \\ \mathbf{S_{yy}} &= \mathbf{R_y^2} + \mathbf{G_y^2} + \mathbf{B_y^2}, \\ \mathbf{S_{xy}} &= \mathbf{R_x R_y} + \mathbf{G_x G_y} + \mathbf{B_x B_y}. \end{aligned} \tag{2.6}$$

The matrix of Eq. (2.5) describes the first order local differential structure of a pixel $\mathbf{p}$. Its eigenvalues are given by:

$$\begin{aligned} \lambda_{\mathbf{1}} &= \tfrac{1}{2}\left(\overline{\mathbf{S_{xx}}} + \overline{\mathbf{S_{yy}}} + \sqrt{\left(\overline{\mathbf{S_{xx}}} - \overline{\mathbf{S_{yy}}}\right)^2 + \left(2\overline{\mathbf{S_{xy}}}\right)^2}\right), \\ \lambda_{\mathbf{2}} &= \tfrac{1}{2}\left(\overline{\mathbf{S_{xx}}} - \overline{\mathbf{S_{yy}}} + \sqrt{\left(\overline{\mathbf{S_{xx}}} - \overline{\mathbf{S_{yy}}}\right)^2 + \left(2\overline{\mathbf{S_{xy}}}\right)^2}\right), \end{aligned} \tag{2.7}$$

where λ_1 represents the energy of the derivative along the most prominent direction, and λ_2 describes the amount of derivative energy perpendicular to the prominent local orientation. As the local prominent orientation (θ) is indicated by the direction of λ_1, then:

$$\theta = \frac{1}{2}\arctan\left(\frac{2\overline{\mathbf{S_{xy}}}}{\overline{\mathbf{S_{xx}}} - \overline{\mathbf{S_{yy}}}}\right). \tag{2.8}$$

A deeper explanation about the color tensor can be found in [29].

2.2.2 Edge merging

Edge merging corresponds to the process of linking non-continuous edges with different j index as belonging to the same original edge. This operation is performed because, during the non-maximum suppression process, some edges can be separated because of external interference over the pipeline edges. The interference may be caused by marine flora, crossing objects, damage etc.

All points in $\mathbf{l_j}$ are fitted to a line using Algorithm 1, with n_{iter} as the maximum number of iterations, and τ_2 as the maximum distance for considering a point as inlier. Distance d from point p to line $L : y = mx + b$ (line 4 of Algorithm 1) is calculated using Eq. (2.9). At the end, all inliers are fitted using Total Least Squares [17]. Therefore, the set of points $\mathbf{l_j}$ are transformed to the set of lines $\mathbf{L_j}$.

$$d(p, L) = \frac{|mp_x - p_y + b|}{\sqrt{m^2 + 1}}. \tag{2.9}$$

The next step is to calculate the distance d from $\mathbf{l_j}$ to $\mathbf{L_k}$ with $k \neq j$, and if most of points of $\mathbf{l_j}$ are in between the region defined by green lines in Fig. 2.9 (the region width is controlled by τ_3), the, they are considered as belonging to the same $\mathbf{l_k}$ set.

An example of the previous step is shown in Fig. 2.9. On the left side, just a few points are inside the region (defined by τ_3), so no merging is produced. In the middle, all points from a segment are inside the region, then these points must belong to the same edge. Finally, the merging result is shown on the right side with merged segments in the same color.

It is also important to mention that the order in which $\mathbf{L_k}$ (red line on Fig. 2.9) were selected depends on the score s_k from the Section 2.2.1. Thereby, lines with higher score were selected first, which represent strong edges that could have been separated during the edge detection step. Also, if two segments are merged, both are not considered during the rest of the merging process. After a merging, a new $\mathbf{L_k}$ is calculated by applying Algorithm 1.

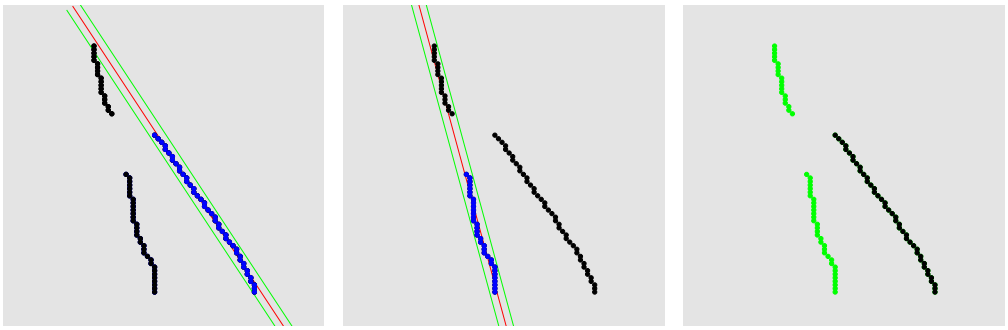


Figure 2.9: Merging process. The reference red line was calculated from the blue points.

Algorithm 1 RANSAC for lines

INPUT: $\mathbf{l_i}$, n_{iter}, τ_2
OUTPUT: $\mathbf{L_i}$ # Fitted line parameters

$m = 0$
while $N > iter$ & $iter < n_{iter}$ **do**
  $\mathbf{p} \leftarrow$ **RandomSample**($\mathbf{l_i}$,2) # Sample randomly two points
  $\mathbf{r} \leftarrow$ **Line**($\mathbf{p}$) # Line parameters
  $\mathbf{d} \leftarrow$ **Distance**($\mathbf{r}, \mathbf{p}$) # Distance from points to line
  inliers $\leftarrow \mathbf{d} \leq \tau_2$
  if $n_{inliers} > m$ **then**
    $m \leftarrow n_{inliers}$ # Store max. number of inliers
    $\mathbf{q} \leftarrow \mathbf{l_j}$(**inliers**) # Store best points
    $e \leftarrow 1 - \frac{m}{n_i}$ # n_i: length of $\mathbf{l_i}$
    $N \leftarrow \frac{\log(0.01)}{\log(1-(1-e)^2)}$
  end if
  $iter \leftarrow iter + 1$
end while
$\mathbf{L_i} \leftarrow$ **TLS**($\mathbf{q}$) # Fitting using total least squares

RANdom SAmple Consensus (RANSAC)

The RANdom SAmple Consensus (RANSAC) algorithm proposed in [30] is an iterative parameter estimator designed to manage a dataset with outliers. It is widely used in computer vision applications [31].

The idea behind the algorithm is to use the minimum number of data samples to estimate the model parameters (unlike other techniques, where the initial solution uses as much data as possible), and then check which elements of the dataset are consistent with the estimated model.

The RANSAC algorithm consists in estimating the model parameters by randomly selecting a minimum number of samples and determining how many samples from the dataset fit the model with a tolerance τ_2 (inliers). If the ratio between the number of inliers and the overall number of samples in the dataset exceeds a threshold m, then the model parameters are re-estimated using all inliers. Otherwise, the

process is repeated N times.

Parameters τ_2 and N must be determined from specific requirements related to the application and to the dataset. The parameter N can be set fixed as n_{iter}, or it can be calculated as:

$$N = \frac{\log(1-a)}{\log(1-(1-v)^n)}, \tag{2.10}$$

where a (usually set to 0.99) is the probability that at least one random sample of n elements is free from outliers, u is the probability that any selected data sample is an inlier and $v = 1 - u$ is the probability that it is an outlier.

For detailed information about this algorithm, see [32].

2.2.3 Pipeline hypothesis and representation

A pipeline hypothesis means possible pipeline location. In this step many hypotheses are generated on the basis that lines $\mathbf{L_1}$ and $\mathbf{L_2}$ could be the pipeline edges if the y coordinate of the intersection point between them is negative. It means that the intersection point should be outside the image as it is shown in Fig. 2.10.

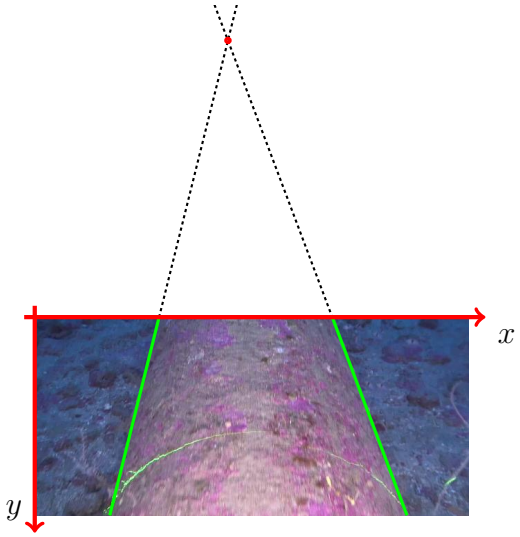


Figure 2.10: Projection of the detected pipeline edges and intersection point outside the image area.

The intersection point p_{int} between $\mathbf{L_1}$ and $\mathbf{L_2}$ is calculated as $x_{int}(\mathbf{L_1}, \mathbf{L_2}) = (b_1 - b_2)/(m_2 - m_1)$ and $y_{int}(\mathbf{L_1}, \mathbf{L_2})$ is calculated by replacing x_{int} on $\mathbf{L_1}$ or $\mathbf{L_2}$.

Each pipeline hypothesis is represented by the geometric parameters δ, α, ω, and β. These parameters are illustrated in Fig. 2.11 and are defined as follows:

- δ: orthogonal distance from the image origin to the pipeline central line.
- α: pipeline center line angle.

- ω: half of the pipeline width.
- β: pipeline edge inclination respect to the center line.

The first three parameters define the pipeline position, orientation and size on the image. The fourth parameter can be considered as the projective effect introduced by the camera inclination (camera plane not parallel to the pipeline).

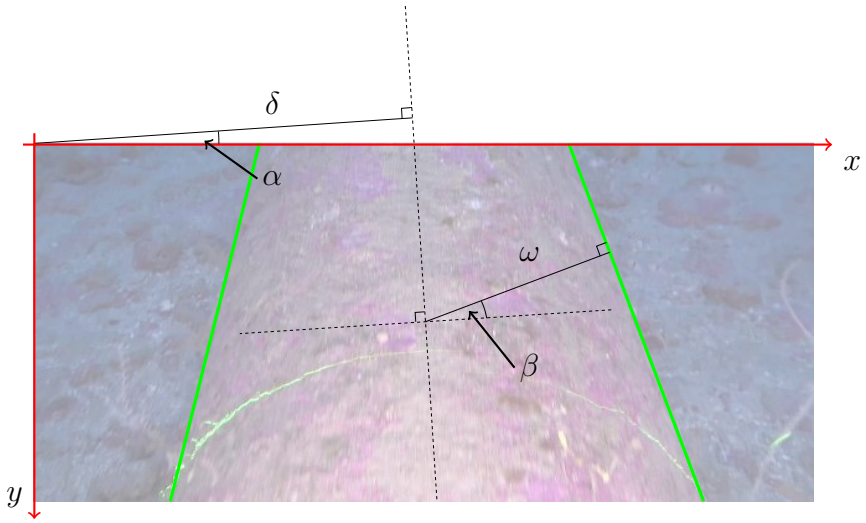


Figure 2.11: Pipeline geometric representation.

At the end of the pipeline detection process a set $\mathbf{M} = (\delta, \alpha, \omega, \beta)^T$ of parameters representing the possible pipeline location is obtained.

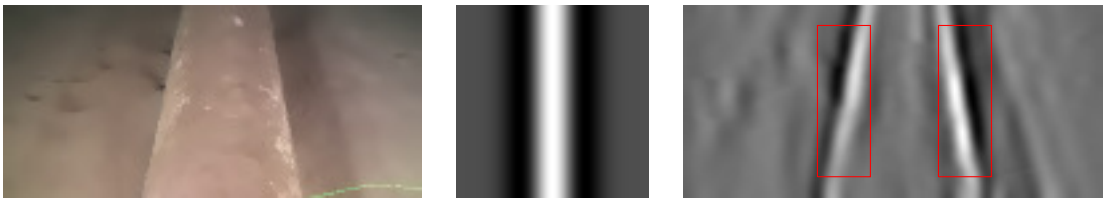
2.3 Initialization

The initialization process consists in automatically detecting the pipeline in the first frame or when the tracking is lost.

The kernel $\mathbf{h}$ consists in a set of mexican-hat functions, having the format shown in Eq. (2.11), arranged vertically on a matrix, as shown in Fig. 2.12 (middle). Experimentally it was possible to identify edge regions close to the local minima of $\mathbf{I_{conv}}$. Therefore, the same procedure of Section 2.2 was used, with $\mathbf{I_{nms}}$ as the non-maximum suppression of $-\mathbf{I_{conv}}$ with $\theta = \frac{\pi}{2}$, thus generating a set $\mathbf{F}$ of initialization hypothesis that represents an approximated pipeline region.

$$f(x) = \left(1 - \frac{x^2}{\sigma^2}\right) \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{2.11}$$

Since each hypothesis in $\mathbf{F}$ is composed by two edges (represented as lines) with scores s_i and s_j $(i \neq j)$, the best hypothesis is selected as $\mathbf{F_k} = \max(s_i s_j)$. This ensures the selection of two strong and long edges. Next, as the initialization



$\mathbf{I}$ $\mathbf{h}$ $\mathbf{I_{conv}} = \mathbf{I} * \mathbf{h}$

Figure 2.12: $\mathbf{I} * \mathbf{h}$. Edge regions marked in red.

hypothesis only gives an approximated region, this region has to be matched to the hypothesis generated from the regular iterative edge detection procedure

The overlap score w, shown in Eq. (2.12), is used to select the best match between the two hypothesis:

$$w(A_{\mathbf{F_k}}, A_{\mathbf{M_i}}) = \frac{A_{\mathbf{F_k}} \cap A_{\mathbf{M_i}}}{A_{\mathbf{F_k}} \cup A_{\mathbf{M_i}}}, \tag{2.12}$$

where A is the area of the hypothesis (see Fig. 2.13). Finally the pipeline edges from the initialization is selected from the maximum w that satisfied the condition $w > \tau_4$. Otherwise, no initialization is considered.

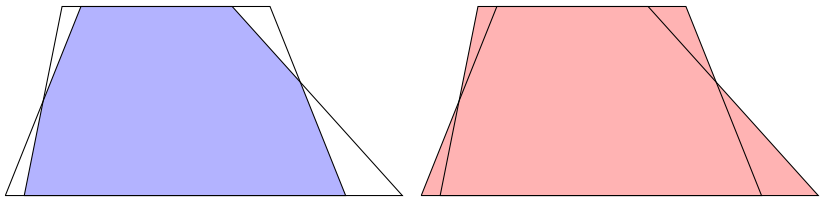


Figure 2.13: Intersection (left) and union (right) of two regions.

2.4 Tracking

The tracking process consist on taking advantage of previous detections to improve the current detection. In this work, two frame and multi frame tracking approaches were developed.

The overlap score from Eq. (2.12) and the relationship between hypothesis magnitudes of Eq. (2.13) were used as matching scores for tracking (columns of $\mathbf{M}$ were normalized to give a similar weight to all variables).

$$z(\mathbf{M}_i^{t-1}, \mathbf{M}_j^t) = |1 - \frac{\|\mathbf{M}_i^{t-1}\|}{\|\mathbf{M}_j^t\|}| \tag{2.13}$$

During the matching score computation, links with weak scores are removed. A weak link is considered when its w score is below a threshold τ_5. Fig. 2.14 shows how weak links in red are removed and an example of an overlap area that generates

a weak link is illustrated in Fig. 2.15. This operation prevents the selection of wrong hypotheses. The best scores are selected as $\max(w)$ or $\min(z)$.

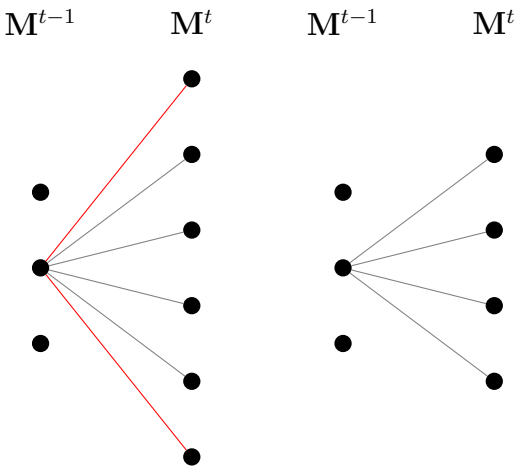


Figure 2.14: Weak links remotion.

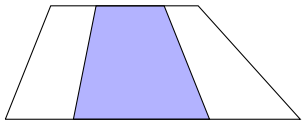


Figure 2.15: Weak link generated by a small area overlap.

2.4.1 Two frame tracking

Two frame tracking consists of continuously selecting a hypothesis from $\mathbf{M^t}$ based on the detection at time $t-1$.

Fig. 2.16 shows how the tracking is performed. The set $\mathbf{M}^t$ in (a) is composed of three hypotheses. The initialization selects the blue node and it is taken as the reference. Then, it is matched with the new hypotheses $\mathbf{M}^t$ (gray links) (b). The hypotheses in $\mathbf{M}^t$ with better link score (maximum w or minimum z) is selected (blue). This process is repeated successively for next hypotheses (c). If there is no link (d) between $\mathbf{M}^{t-1}$ and $\mathbf{M}^t$, the tracking is considered as finalized and a new initialization has to be done at time t.

2.4.2 Multi frame tracking

The idea of multi frame tracking is to maximize (for w) or minimize (for z) the path from $\mathbf{M}^{t-k+1}$ to $\mathbf{M}^t$ (k is the number of frames used). To solve this, dynamic programming was used.

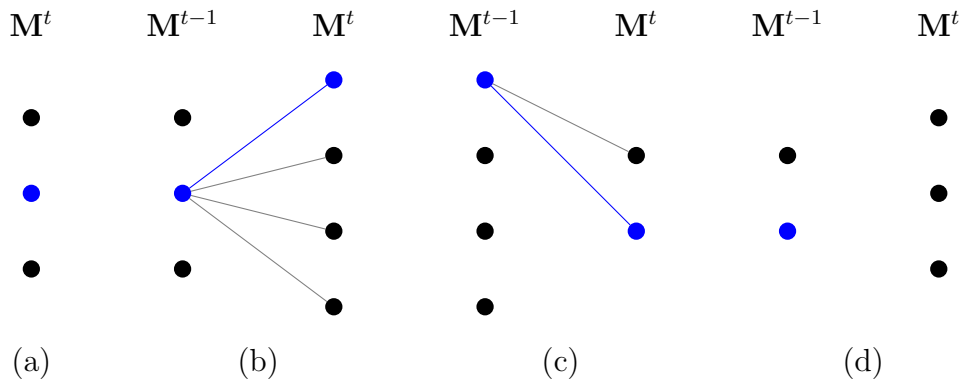


Figure 2.16: Two frame tracking representation. (a) node selected during (blue), (b) matching (gray) and (c) selection (blue) between consecutive hypotheses, (d) tracking finalization.

Dynamic programming is an optimization technique that converts a complex problem into a sequence of simpler subproblems [33]. A detailed explanation of the algorithm used can be found in Chapter 10 of [34].

Fig. 2.17 illustrates an example for $k = 4$. A ghost node (gray) with link score equal to 1 (dashed lines) was added to the end, in order to avoid calculating the best path from the hypothesis in $\mathbf{M}^{t-3}$ to each hypothesis $\mathbf{M}^t$. The result of calculating the best path from the initialized node in $\mathbf{M}^{t-3}$ to the ghost node includes the intermediate detections (Fig. 2.18). Once the path was calculated, the process is repeated for a new set of k frames, with the last detection as the first node. If there is no matching between $\mathbf{M}^{t-n-1}$ and $\mathbf{M}^{t-n}$ $(0 \leq n \leq k-2)$, then tracking is considered as finalized and a new initialization has to be performed in $\mathbf{M}^{t-n}$.

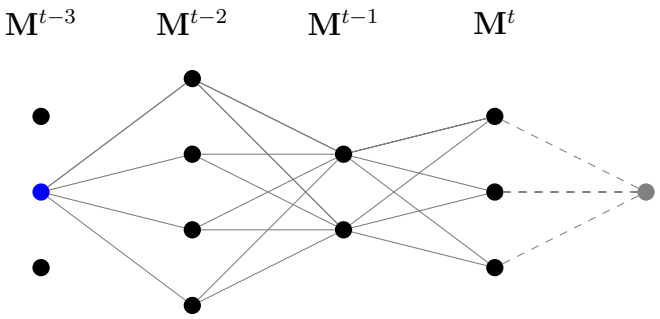


Figure 2.17: Multi frame tracking example. Initial structure with a ghost node (gray) at the end.

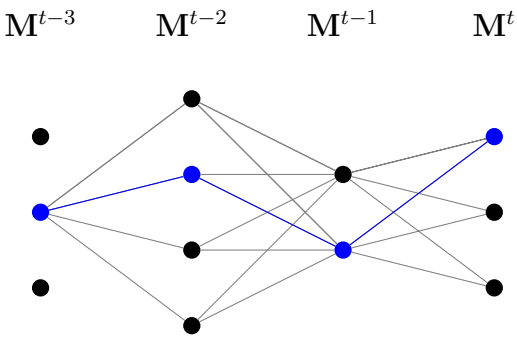


Figure 2.18: Multi frame tracking example. Result of finding the best path. The nodes in blue were selected during the optimization.

2.5 Segmentation and 2D top view transformation

The segmentation task consists on the separation of the pipeline from the image background. And the 2D top view transformation consists of removing the perspective effect in the image, making the pipeline edges parallel. An example of the 2D top view transformation is shown in Fig. 2.19.

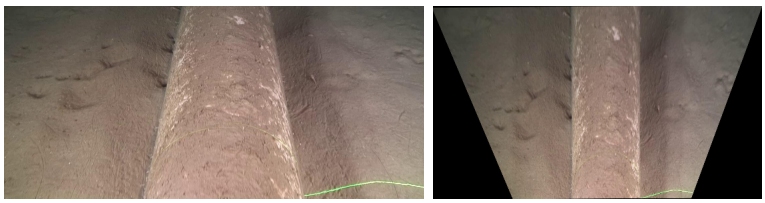
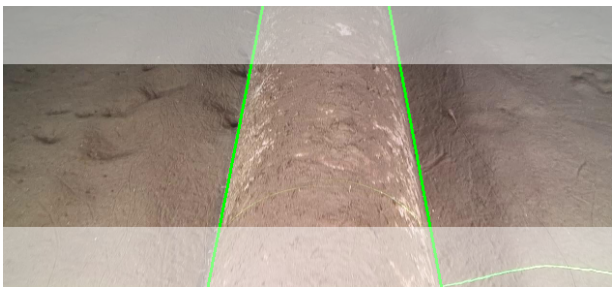


Figure 2.19: Original image (left) and its 2D top view transformation (right).

A region of interest is defined as in Fig. 2.20. During the 2D top view transformation, the upper side has a blurring effect produced by pixel interpolation.

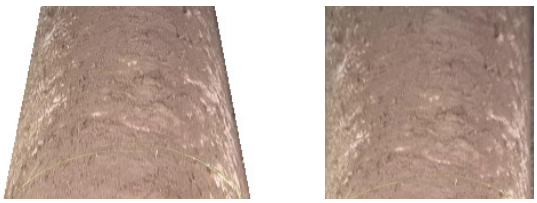
The segmentation process is done over the region of interest in $\mathbf{I_1}$, since $\mathbf{I_2}$ has a low resolution and some details were lost during the filtering. Thus, it is necessary to scale by 4 δ and β parameters. The segmented image $\mathbf{I}_{sg}$ is generated by cropping the pipeline region and removing the background pixels as shown in Fig. 2.21 (left), background pixels were set to 1.

The 2D top-view transformed image $\mathbf{I}_{tv}$ (right) is generated by using the Algorithm 2. The four points used to calculated the transformation are shown in Fig. 2.22, and the destination points corresponds to the vertices of a rectangle with 2ω wide and the same hight of the region of interest. Black points represent the region of interest limits.



$\mathbf{I_1}$

Figure 2.20: Region of interest.



$\mathbf{I}_{sg}$ $\mathbf{I}_{tv}$

Figure 2.21: Segmented pipeline (left) and its 2D top view transformation.

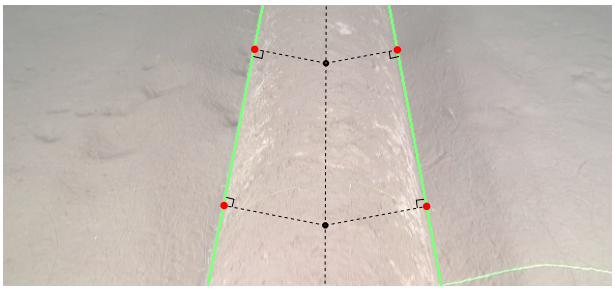


Figure 2.22: Four points (red) used for 2D top view transformation.

Projective transformation

Projective transformation or homography is a non singular linear transformation:

$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix},$$

or in block form:

$$\mathbf{x}' = \mathbf{H}\mathbf{x} = \begin{bmatrix} \mathbf{A}_{2\times 2} & \mathbf{t}_{2\times 1} \\ \mathbf{v}^T_{1\times 2} & v \end{bmatrix} \mathbf{x}, \tag{2.14}$$

where $\mathbf{A}$ is a non-singular matrix composed by two rotations ($\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}$) and two scaling operations (s_x and s_y):

$$\mathbf{A} = \mathbf{R}(\alpha)\mathbf{R}(-\phi) \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \mathbf{R}(\phi), \tag{2.15}$$

$\mathbf{t}$ is the translation vector, and $\mathbf{v} = (v_1, v_2)^T$. It should be noted that, if $\mathbf{H}$ is multiplied by an arbitrary non-zero factor, the transformation is not altered. There are eight independent ratios among the elements of $\mathbf{H}$, thus a projective transformation has eight degrees of freedom [31].

Removing projective distortion

When an image is distorted by a projective transformation, its possible to recover the original representation by applying the inverse transformation to the image. It will result in a new image where the objects are shown in their correct shape [31].

Algorithm 2 is used in order to remove the projective distortion from an image.

Algorithm 2 Projective distortion remotion

1: select four points (x_i', y_i') on the image
2: select the corresponding points (x_i, y_i) in perspective free space
3: find the correspondence between (x, y) and (x', y') as:

$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

We can set $h_{33} = 1$ without losing generalization [31].
4: solve the system:

$$\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 & -x_0x'_0 & -y_0x'_0 \\ 0 & 0 & 0 & x_0 & y_0 & 1 & -x_0y'_0 & -y_0y'_0 \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 & -y_3x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 & -y_3y'_3 \end{bmatrix}^{-1} \begin{bmatrix} x'_0 \\ y'_0 \\ x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \end{bmatrix}$$

5: apply $\mathbf{H}^{-1}$ to the $\mathbf{I}_1$

Chapter 3

System evaluation

Experiments for evaluating the proposed system (Fig. 2.1) performance, as well as the selection of internal parameters, are presented in this chapter. Four experiments were conducted. The first one to evaluate the image conditioning and pipeline detection blocks. The second is to evaluate the initializer. Tracking block is evaluated in the third experiment and the segmentation in the last one.

The evaluation is performed by a comparison of the blocks output and the dataset. Information about the dataset can be found in Appendix A.

For the experiments, a detection is considered as valid if the overlap area (w) between the hypotheses (generated by the system) and the ground truth is greater than 0.9. This value means a high similarity between the output and the ground truth.

The following parameters were set based on qualitative analysis of the results during the system development:

- $\tau_1 = 5$, minimum edge length (pixels);
- $\tau_2 = 5$, maximum distance from point to line on RANSAC (pixels);
- $\tau_3 = 5$, maximum distance from edge points to line during edge merging (pixels);
- $n_{tau} = 30$, maximum RANSAC iterations;

3.1 Experiment 1

The purpose of this experiment is to analyze the image conditioning and pipeline detection processes. Also, the following parameters will be selected:

- bf_{iter}, number of iterations for the bilateral filter;
- σ_s, spatial standard deviation (bilateral filter);

- σ_r, range standard deviation (bilateral filter);
- Color edge detector.

This experiment follows the structure shown in Fig.3.1. $\mathbf{I_2}$ passes through the pipeline detection block (dashed lines) and the generated detections were compared to the ground truth (GT) in order to find the best hypothesis that meets $w \geq 0.9$.

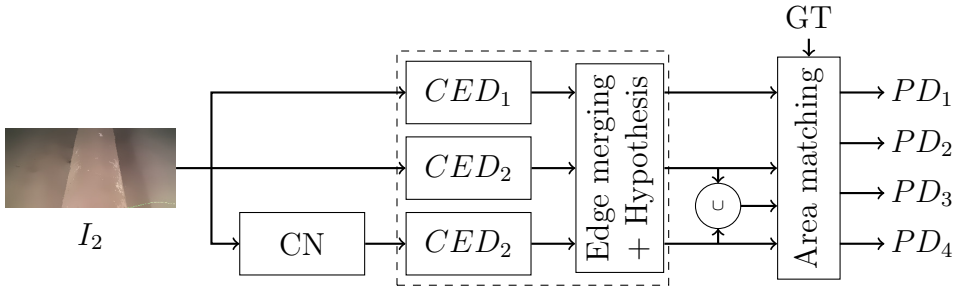


Figure 3.1: Block diagram for Experiment 1. CED (color edge detector), CN (color normalization), GT (ground truth).

Four different pipeline detections (PD) outputs are generated following this structure and are explained next:

- PD_1, output of using the color edge detector CED_1 from in [1];
- PD_2, output of using the color edge detector proposed in Section 2.2.1 (CED_2);
- PD_3, output of using a color normalization (CN) of $\mathbf{I_2}$, followed by CED_2;
- PD_4, output of joining the hypotheses generated by CED_2 and $\text{CN} + \text{CED}_2$.

Color normalization of and RGB image is calculated using Eq. 3.1.

$$R' = \frac{R}{R+G+B}; G' = \frac{G}{R+G+B}; B' = \frac{B}{R+G+B} \tag{3.1}$$

Table 3.1 shows the best score matching average between hypothesis and ground truth for each detector. PD_1 had a result lower than those obtained by the proposed detectors, which means that the edges produced by this detector do not contribute to the generation of good hypotheses. This demonstrates that the pipeline orientation based restriction explained in Section 2.2.1 improves the detection process. Therefore, PD_1 was not used for next test.

PD_1	PD_2	PD_3	PD_4
0.854 $\pm$0.142	0.920 $\pm$0.068	0.956 $\pm$0.042	**0.966$\pm$0.028**

Table 3.1: Best matching average between hypotheses and ground truth.

The bf_{iter} parameter is selected from the detection rate results shown in Table 3.2.

bf_{iter}	PD_2 (%)	PD_3 (%)	PD_4 (%)
1	6026 (77.2)	7383 (94.6)	**7535 (96.5)**
2	5851 (74.9)	7363 (94.3)	**7532 (96.5)**
3	5750 (73.6)	7340 (94.0)	**7527 (96.4)**
4	5571 (71.3)	7332 (93.9)	**7533 (96.5)**

Table 3.2: Bilateral filter iterations with $\sigma_s = 5$ and $\sigma_r = 0.1$.

The increase of bf_{iter} did not mean a significant improvement for PD_4, the difference between its best and worst result was about 0.1%, for PD_2, the difference was 5.9% and for PD_3 it was 0.7%. All cases had better results with $bf_{iter} = 1$, these values were used for next experiments. The variation of σ_s and σ_r did not show significant differences for all cases. Therefore, $\sigma_s = 5$ and $\sigma_r = 0.1$ were selected.

Table 3.3 shows a detailed detection result for each video of the dataset. For videos 1, 2 and 4, the result of PD_3 is considerably higher than PD_2, and opposite in videos 5 and 6. Results were equivalent in video 3. This is due to the difference between scenarios on each video. As PD_4 is a combination between PD_2 and PD_4, this works better for all scenarios on the dataset. It is so, PD_4 is chosen as the color edge detector for the system and was used for next experiments.

Video	PD_2 (%)	PD_3 (%)	PD_4 (%)
1	1352 (75.2)	1685 (93.7)	1685 (93.7)
2	780 (48.8)	1586 (99.2)	1586 (99.2)
3	1755 (97.6)	1759 (97.8)	1759 (97.8)
4	1022 (71.8)	1386 (97.3)	1386 (97.3)
5	221 (78.4)	203 (71.6)	221 (78.4)
6	896 (99.0)	769 (84.6)	896 (99.0)
Total	6026 (77.2)	7383 (94.6)	**7533 (96.5)**

Table 3.3: Pipeline detection for each dataset.

3.2 Experiment 2

The purpose of this experiment is to analyze the initialization processes. Also, the selection of the following parameters:

- h_{size}, kernel size.
- σ_h, mexican-hat standard deviation.

In this experiment, four different initialization (IN) outputs were generated and them are explained next:

- IN_1, red channel of $\mathbf{I_2}$ convolved with h;
- IN_2, green channel of $\mathbf{I_2}$ convolved with h;
- IN_3, blue channel of $\mathbf{I_2}$ convolved with h;
- IN_4, combination of IN_1, IN_2 and IN_3.

For IN_4, the initialization process was performed in each color channel of $\mathbf{I_2}$. Then, one was selected based on the highest w score between the hypotheses from the pipeline detection and the hypothesis from the initializer. τ_4 was set to 0.85 based on qualitative analysis during the system development.

In Table 3.4 shows the initialization detection rate results for different kernel sizes. The first three rows shows that the kernel 10x20 (width > height) has a better performance compared the others. From this result, tests of row 4 to 11 were carried out to determine the best h_{size} for the initializer.

h_{size}	IN_1	IN_2	IN_3	IN_4
10×10	1555 (19.92)	1237 (15.84)	1406 (18.01)	2096 (26.84)
10×20	5241 (67.12)	3932 (50.36)	4669 (59.80)	**5976 (76.54)**
20×10	970 (12.42)	772 (9.89)	759 (9.72)	1229 (15.74)
5×10	1765 (22.61)	1405 (17.99)	1585 (20.30)	2333 (29.88)
10×20	5241 (67.12)	3932 (50.36)	4669 (59.80)	5976 (76.54)
15×30	6281 (80.44)	4717 (60.41)	5641 (72.24)	6763 (86.62)
20×40	7023 (89.95)	5485 (70.25)	6504 (83.30)	**7349 (94.12)**
25×50	6739 (86.31)	5304 (67.93)	6278 (80.40)	7087 (90.77)
30×60	6426 (82.30)	5003 (64.08)	5979 (76.58)	6699 (85.80)
35×70	6272 (80.33)	4833 (61.90)	5881 (75.32)	6528 (83.61)
40×80	6071 (77.75)	4643 (59.46)	5550 (71.08)	6329 (81.06)

Table 3.4: Initialization detection rate for different kernel sizes, with $\sigma_h = 0.2$.

The kernel 20×40 had better detection rate compared to the others. This improvement had as a consequence the increase of the processing time, since the initialization had to be performed three times.

Table 3.5 shows the influence of the σ_h parameter. It can be seen that the best results were obtained with $\sigma_h = 0.200$ for IN_4.

The effect of the relationship between the kernel height and width is shown in Fig. 3.2. 10×10 and 20×10 kernels produced a noise output where the pipeline edges are hardly to see. The kernel 10×20 produce soft and well-defined regions, because it is composed of ten 1D mexican-hat function of 20 points arrangement in a matrix, it tends to emphasize vertical edges, while horizontal edges are smoothed, facilitating the pipeline identification process.

σ_h	IN_1	IN_2	IN_3	IN_4
0.100	4358 (55.81)	3165 (40.54)	3978 (50.95)	5184 (66.39)
0.125	5546 (71.03)	4087 (52.34)	5041 (64.56)	6223 (79.70)
0.150	6406 (82.04)	4809 (61.59)	5767 (73.86)	6882 (88.14)
0.175	6800 (87.09)	5186 (66.42)	6194 (79.33)	7172 (91.85)
0.200	7000 (89.65)	5494 (70.36)	6519 (83.49)	**7349 (94.12)**
0.225	6989 (89.51)	5554 (71.13)	6552 (83.91)	7317 (93.71)
0.250	6853 (87.77)	5525 (70.76)	6460 (82.74)	7197 (92.17)

Table 3.5: Initialization result with $h_{size} = 20 \times 40$ and variable σ_h.

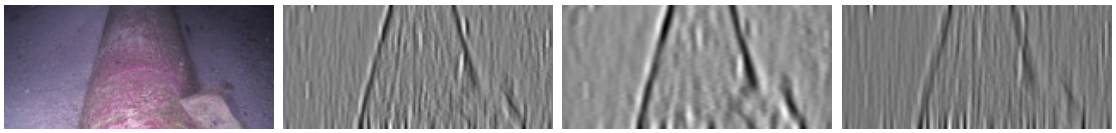


Figure 3.2: Image convolution example for initialization. From left to right, input image, convolution with a kernel: 10×10, 10×20 and 20×10 with $\sigma = 0.2$

3.3 Test 3

Evaluation of the tracking process was the purpose of this experiment. Also, the selection of the number of frames used during tracking.

Since the ground truth images were sampled at 0.5 seconds, it was not possible to test the tracking directly on them. So, the tracking process was performed on the original videos and the result was sampled every 15 frames and then compared to the corresponding ground truth.

During tracking, $\tau_5 = 0.85$ was used for link removing, since it was used to compare pipeline hypotheses in consecutive frames with a different location. In Table 3.6 it is shown the detection rate of the tracking process by increasing the number of frames from 2 to 25 frames. Area overlap w, relationship between hypotheses magnitudes z and a linear combination of w and z (Eq. 3.2) were used as matching score during tracking for comparison.

$$v = 0.5w + 0.5(1 - z) \tag{3.2}$$

Results using z score were inferior compared to w and w, while v results were similar to w, so it it can be inferred that the influence of z over w in v was minimal. The maximum detection rate for w (94.0%) was obtained using 19 frames during tracking. However, the increment from 9 to 19 frames for tracking represents +0.1% of variation. It is also The same goes for v, with +0.1% of variation.

Ideally, the system should be initialized once (at the beginning), but due to the complexity of the scenarios, many re-initializations are performed during the tracking, which mean an increment of processing time. Table 3.7 shows that as

# frames	w	z	v
2	7095 (90.9)	6112 (78.3)	7103 (91.0)
3	7144 (91.5)	6224 (79.7)	7142 (91.5)
5	7201 (92.2)	6310 (80.8)	7132 (91.3)
7	7303 (93.5)	6381 (81.7)	7230 (92.6)
9	7333 (93.9)	6522 (83.5)	7289 (93.4)
11	7292 (93.4)	6557 (84.0)	7302 (93.5)
15	7298 (93.5)	6638 (85.0)	7284 (93.3)
19	**7338 (94.0)**	6658 (85.3)	**7302 (93.5)**
25	7211 (92.4)	**6679 (85.5)**	7191 (92.1)

Table 3.6: Correct pipeline detections during tracking.

the number of frames used for tracking increments, the number of initialization was reduced. This is because the purpose of using more frames is to maximize the path score between them, thus reducing the options of wrong matchings in the middle.

# frames	w	z	v
2	887	1164	901
3	780	965	791
5	641	797	643
7	584	691	589
9	576	618	575
11	535	607	541
15	514	548	505
19	508	539	513
25	**482**	**510**	**488**

Table 3.7: Number of initialization during tracking.

Considering the complete videos for 2, 5 and 19 frames tracking (minimum, intermediate and maximum detection rates), the similarity between the results of w and v are 96.3%, 94.9% and 96.3%, respectively, which once again indicates that the influence of z over w was not strong in v. Then, w score was selected for tracking.

3.4 Test 4

The segmentation accuracy was evaluated in this test. The results from tracking with 2, 5 and 9 frames were used for comparison, 9 frames was selected instead of 19 because there was only 0.1% of difference on the detection rate between them. The area overlap between detections and ground truth were used as metric to evaluate the segmentation accuracy.

Tables 3.8 and 3.9 show the average accuracy. The first one calculated the accuracy using all detections. While the second one only took into consideration the correct detections from Table 3.6.

Video	2	5	9
1	0.946	0.943	0.944
2	0.959	0.961	0.961
3	0.947	0.955	0.958
4	0.920	0.921	0.943
5	0.913	0.908	0.906
6	0.971	0.973	0.974
Total	0.946	0.947	0.952

Table 3.8: Segmentation average accuracy.

Video	2	5	9
1	0.962 ±0.020	0.961 ±0.020	0.961 ±0.021
2	0.967 ±0.017	0.966 ±0.018	0.966 ±0.019
3	0.959 ±0.020	0.960 ±0.020	0.961 ±0.019
4	0.958 ±0.020	0.957 ±0.019	0.954 ±0.021
5	0.956 ±0.020	0.955 ±0.021	0.955 ±0.021
6	0.977 ±0.010	0.978 ±0.010	0.978 ±0.010
Total	0.963 ±0.020	0.963 ±0.020	0.963 ±0.020

Table 3.9: Segmentation average accuracy.

High segmentation accuracies were achieved from both tables, it means that the detections obtained from tracking are good approximations of the real pipeline.

Finally, the 9 frames tracking approach was selected considering its implementation. It avoided buffering too many images and intermediate results while keeping high detection rates and high segmentation accuracy.

3.5 Final system setup

The final system was proposed with the following parameters:

- Image conditioning: $bf_{\text{iter}} = 1, \sigma_s = 5, \sigma_r = 0.1$;
- Pipeline detection: $\tau_1, \tau_2, \tau_3 = 5, n_{\text{iter}} = 30$, combined color edge edge detection from RGB and normalized RGB image input;
- Initialization: $h_{\text{size}} = 20 \times 40, \sigma = 0.2, \tau_4 = 0.85$;
- Tracking: $\tau_5 = 0.85$, 9 frames path with w score for matching;
- Segmentation and 2D top view representation: does not depend on parameters.

3.6 System outputs demonstrations

The following figures show some samples of the system output (without segmentation) using 2, 5 and 9 frames for tracking.

Fig. 3.3 corresponds to the video 1. It is possible to see in row 4 that no one of the compared methods managed to detect the pipeline. This could be due to the illumination conditions and the distance from the camera to the pipeline. Correct detections are observed in the presence of algae in rows 5 and 6. The system detected the pipeline under a partial occlusion of one of its edges, as shown in row 8.

Figure 3.3: Detection example from video 1. From left to right: ground truth (red), tracking using 2, 5 and 9 frames (yellow).

Output samples from video 2 are presented in Fig. 3.4. The presence of sand on the pipeline sides is a characteristic of this video. Tracking with 2 frames presented some wrong detections compared to the others.

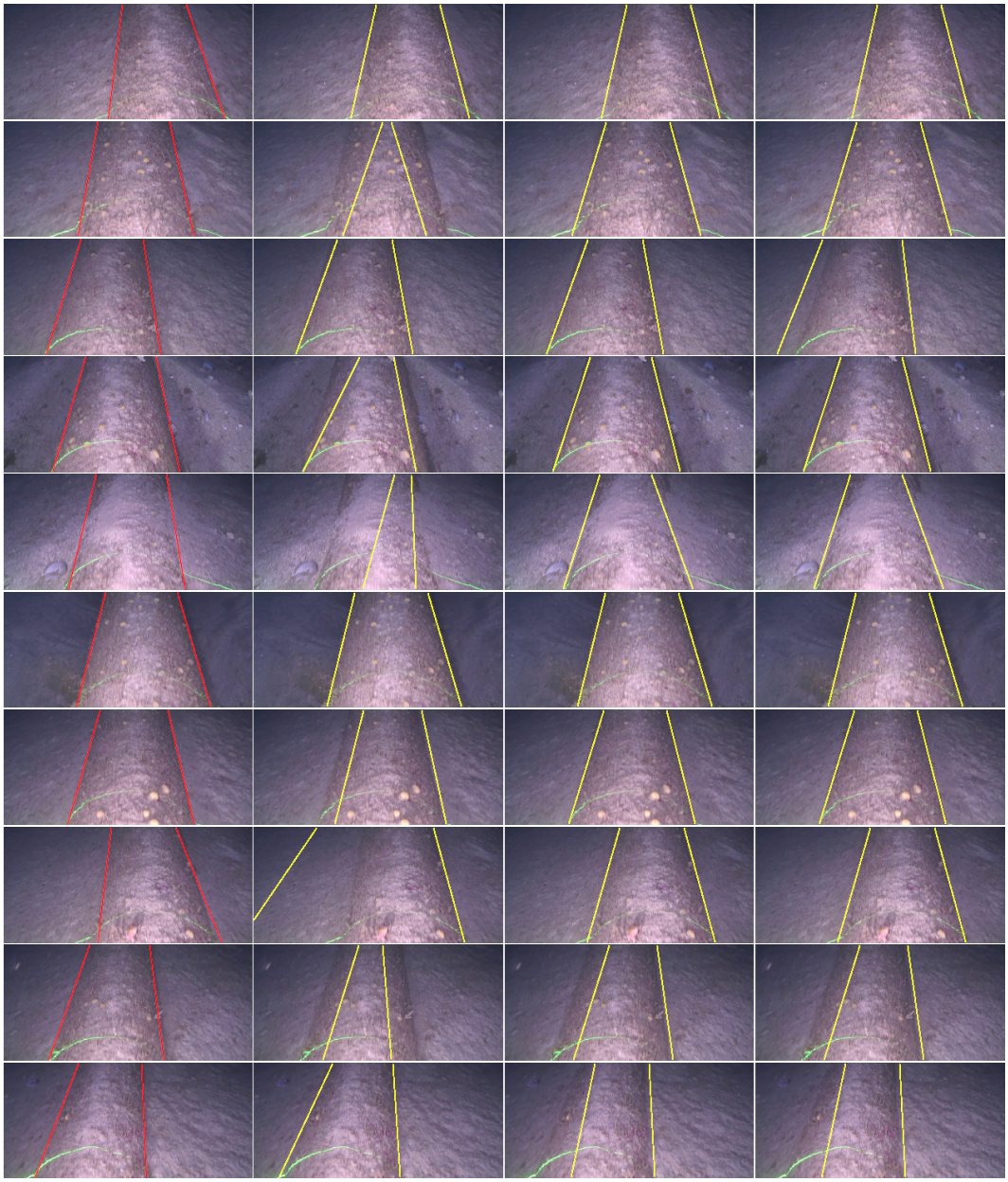


Figure 3.4: Detection example from video 2. From left to right: ground truth (red), tracking using 2, 5 and 9 frames (yellow).

Rows 2 and 6 of Fig. 3.5 shows the detection of the pipeline (video 3) when there is a partial occlusion of the borders. Tracking with 5 and 9 frames presented the same output in all samples.

Figure 3.5: Detection example from video 3. From left to right: ground truth (red), tracking using 2, 7 and 15 frames (yellow).

Also, in Fig 3.6, tracking with 5 and 9 frames had the same result. Characteristics of this video (4) are the similarity of the texture between the pipeline and the seabed, the change in the geometry of the pipeline (row 7), partial occlusion (row 9) and non-uniform illumination (row 10).

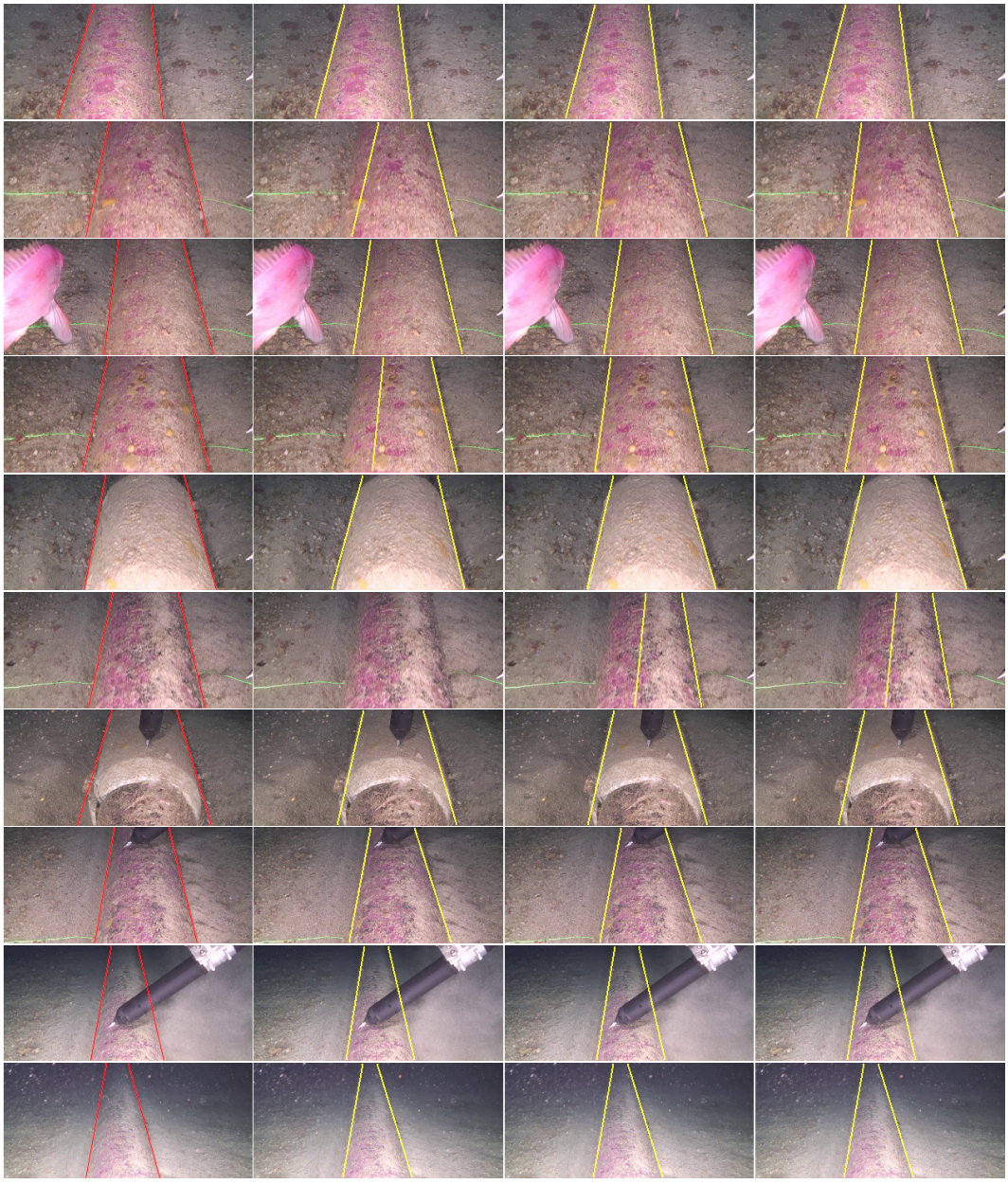


Figure 3.6: Detection example from video 4. From left to right: ground truth (red), tracking using 2, 5 and 9 frames (yellow).

According to Table 3.3, the worst detection rates were obtained on this dataset (Fig. 3.7). The texture of the environment made the detection process more difficult. It is possible to distinguish the pipeline in row 3. However, tracking with 2 and 5 frames had wrong detections, also no detection was obtained in tracking with 9 frames. This can be a result of tracking wrong hypothesis in past frames.

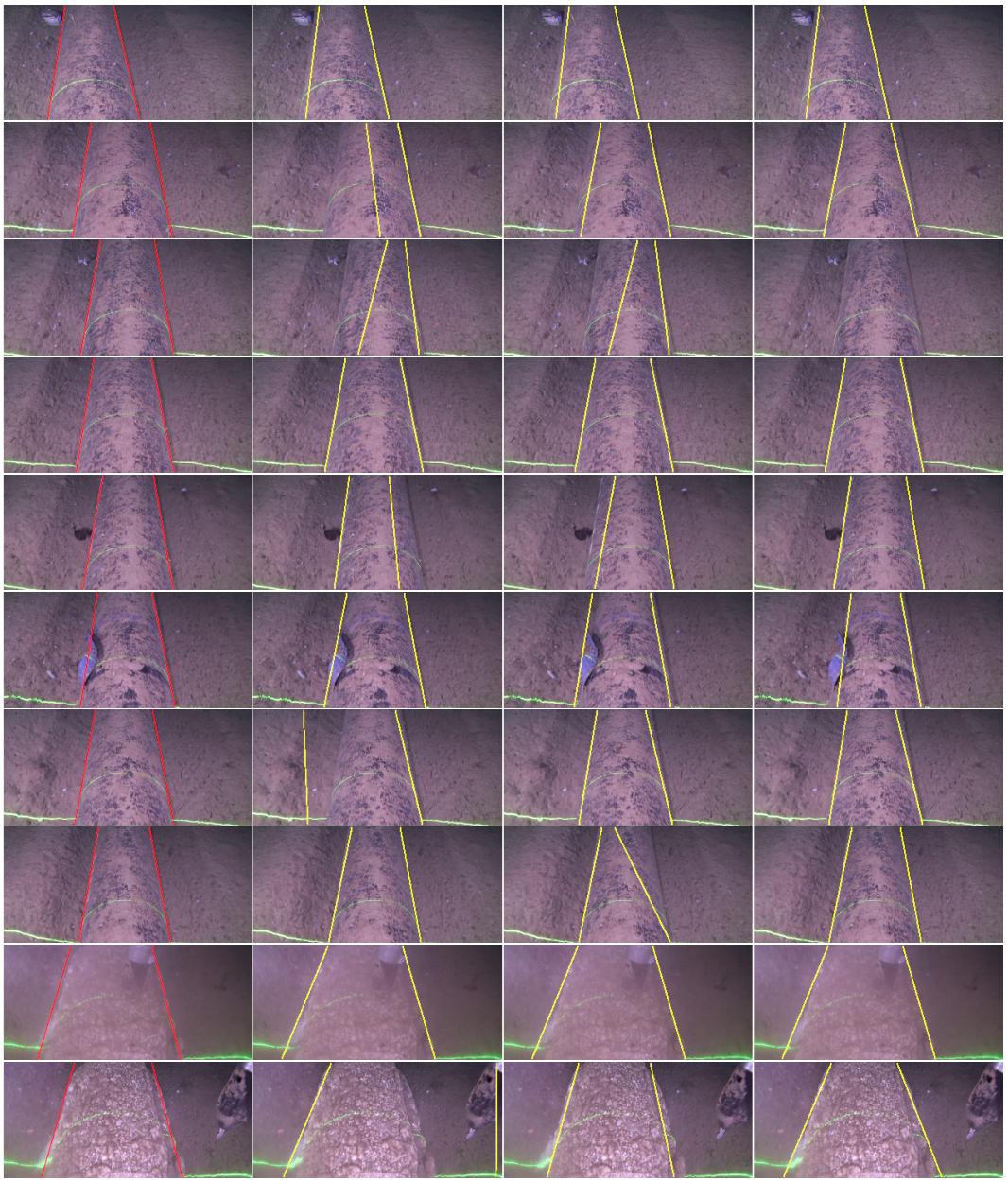


Figure 3.7: Detection example from video 5. From left to right: ground truth (red), tracking using 2, 5 and 9 frames (yellow).

Rows 4 and 5 of Fig. 3.8 show the crossing of two pipelines. It was not possible to detect the pipeline (vertical position) in the first one, but once a part of the pipeline border was visible, the system could detect it. A characteristic of this video (6) is that there is sand on top of the pipeline. The distance from the pipeline to the seabed and the good illumination condition facilitated the detection process making the pipeline edges well defined.

Figure 3.8: Detection example from video 6. From left to right: ground truth (red), tracking using 2, 5 and 9 frames (yellow).

Pipeline detection, segmentation and 2D top view transformation are shown in Fig. 3.9 for a variety of scenarios contained in the whole dataset. The separation of the pipeline from the environment and the the 2D transformation will facilitate further analysis of its conservation state.

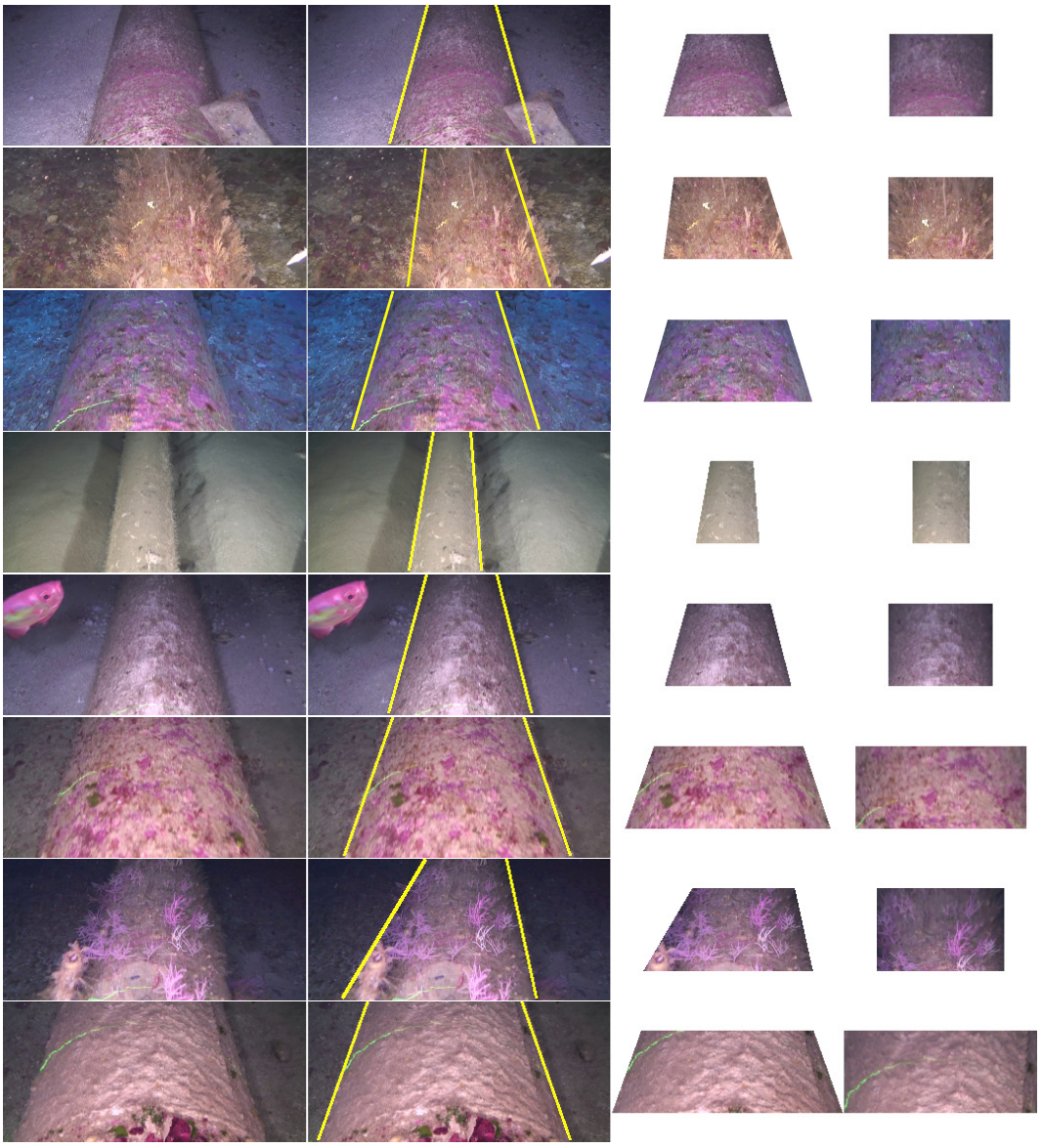


Figure 3.9: From left to right: input image, detected pipeline (yellow), segmented pipeline and 2D top view.

Finally, Fig. 3.10 presents the evolution of the pipeline center for each video over 60 seconds. Some discontinuities can be observed for videos 5 and 6. This means consecutive re-initializations of the tracking due to wrong detections or the absence of these. Smooth trajectories can be seen, it can interpret as correct pipeline detections during tracking over time, which was also corroborated by a qualitative analysis of the output videos.

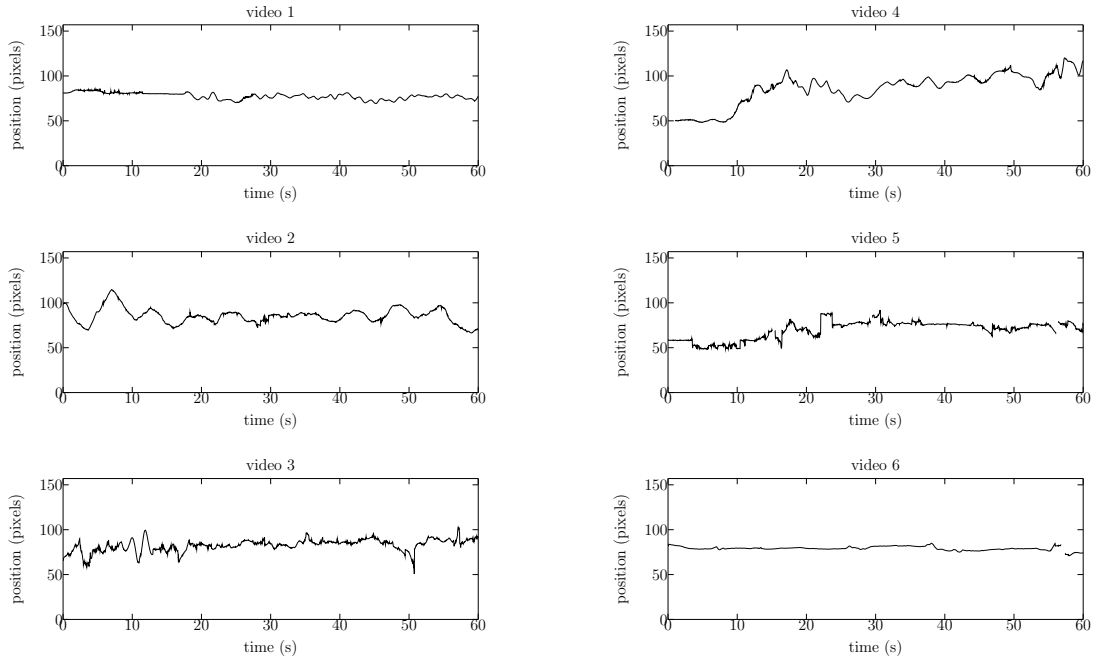


Figure 3.10: Pipeline center evolution over time for all videos.

All the system development was realized in Matlab R2014a. Image processing operations were speedup using MEX functions for OpenCV. Processing a video of 30 minutes takes around 1 hour on a notebook Intel core i7-5500U processor with 8GB of RAM and Ubuntu 16.04 operating system.

Chapter 4

Conclusions

Nowadays, the inspection of underwater pipelines has become a frequent task, which has increased the need for an automatic inspection system. This new system can be built from two subsystems, one that is responsible for isolating the pipeline from its environment, and another that detects and classifies events on the pipeline structure.

An underwater pipeline segmentation system was presented in this work. The system was evaluated on real inspection images, achieving high performance. Different tests and comparisons were made to determine the internal parameters of the system.

The reduction of the input image size allows the approximation of pipeline edges to lines. For example, the presence of small plants on the pipeline edges can be seen as small undulations on the resized image and fitted to a line. The system not handle large deformations.

The pipeline orientation restriction used during the edge detection process increased on 11.2% the average matching compared to the edge detector without restriction. The detection rate of the combined edge detector was 19.3% higher than an individual edge detector.

In the initialization test, 94.1% of detection rate was achieved. A higher detection rate was expected, due to it is a more complex process and wrong initializations can lead to erroneous tracks.

The pipeline detection rate and the number of initializations were calculated in order to select the matching score and the number of frames used for tracking. The best result was obtained with 19 frames and v score. However, w score and 9 frames for tracking were selected, since the difference of the detection rate between them is 0.1%, and fewer frames are buffered.

The segmentation process reached 96.3% of average accuracy. And finally, the segmented pipelines were transformed into a 2D top view representation.

During this work, [35] was published in a international conference proceedings.

Future works include the improvement of the initialization process in order to

achieve higher detection rates. The extension of the system to multiple pipeline segmentation. And the development of an event detection and classification subsystem.

Bibliography

[1] VAN DE WEIJER, J., GEVERS, T., SMEULDERS, A. W. M. "Robust photometric invariant features from the color tensor", *IEEE Transactions on Image Processing*, v. 15, n. 1, pp. 118–127, Jan 2006.

[2] ORTIZ, A., SIMÓ, M., OLIVER, G. "A vision system for an underwater cable tracker", *Machine Vision and Applications*, v. 13, n. 3, pp. 129–140, Jul 2002.

[3] WHITCOMB, L. L. "Underwater robotics: out of the research laboratory and into the field". In: *IEEE International Conference on Robotics and Automation. Symposia Proceedings*, v. 1, pp. 709–716 vol.1, 2000.

[4] NEXANS. "ROV umbilical and tether". Disponível em: <`http://www.nexans.no/eservice/Norway-no_NO/navigate_331785/ROV_umbilical_and_tether.html`>. Accessed on: March 19, 2018.

[5] C, M., S, P., L, H., et al. "Subsea infrastructure inspection: A review study". In: *IEEE International Conference on Underwater System Technology: Theory and Applications (USYS)*, pp. 71–76, Dec 2016.

[6] CHEN, J., GONG, Z., LI, H., et al. "A detection method based on sonar image for underwater pipeline tracker". In: *Second International Conference on Mechanic Automation and Control Engineering*, pp. 3766–3769, July 2011.

[7] VILLAR, S., ACOSTA, G., SOUSA, A., et al. "Evaluation of an efficient approach for target tracking from acoustic imagery for the perception system of an autonomous underwater vehicle", *International Journal Of Advanced Robotic Systems*, v. 11, pp. 1–13, August 2013.

[8] PAVIN, A. M. "The Pipeline Identification Method Basing on AUV's Echo-Sounder Data". In: *OCEANS 2006*, pp. 1–6, Sept 2006.

[9] XIANG, X., YU, C., NIU, Z., et al. "Subsea Cable Tracking by Autonomous Underwater Vehicle with Magnetic Sensing Guidance", *Sensors*, v. 16, n. 8, 2016.

[10] JORDÁN, M. A., TRABES, E. "An Adaptive Vision-based Sensor for Underwater Line Detection Employing Shape and Color Image Segmentation", *Journal of Automation and Control Engineering*, v. 3, n. 6, pp. 480–486, April 2015.

[11] J., P. D., KUHN, V., GOMES, S. "Tracking System for Underwater Inspection Using Computer Vision". In: *2012 International Conference on Offshore and Marine Technology: Science and Innovation*, pp. 27–30, March 2012.

[12] ORTIZ, A., ANTICH, J., OLIVER, G. "A particle filter-based approach for tracking undersea narrow telecommunication cables", *Machine Vision and Applications*, v. 22, n. 2, pp. 283–302, Mar 2011.

[13] CHENG, C. C., JIANG, B.-T. "A robust visual servo scheme for underwater pipeline following". In: *19th International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp. 456–459, April 2012.

[14] FORESTI, G. L., GENTILI, S. "A hierarchical classification system for object recognition in underwater environments", *IEEE Journal of Oceanic Engineering*, v. 27, n. 1, pp. 66–78, Jan 2002.

[15] GORIL M. BREIVIK, SIGURD A. FJERDINGEN, O. S. "Robust pipeline localization for an autonomous underwater vehicle using stereo vision and echo sounder data", *Proc.SPIE*, v. 7539, pp. 7539 – 7539 – 12, 2010.

[16] TIAN, W.-M. "Integrated method for the detection and location of underwater pipelines", *Applied Acoustics*, v. 69, n. 5, pp. 387 – 398, 2008.

[17] DE GROEN, P. "An introduction to Total Least Squares", *Nieuw Archief voor Wiskunde*, v. 14, pp. 237–254, 1996.

[18] JORDAN, M., BUSTAMANTE, J. "Auto-sintonización basada en la optimización de la performance en vehículos subacuáticos guiados adaptivamente". In: *XXI Congreso Argentino de Control Automático*, pp. 1–3, Sep 2008.

[19] TOMASI, C., MANDUCHI, R. "Bilateral filtering for gray and color images". In: *Sixth International Conference on Computer Vision*, pp. 839–846, Jan 1998.

[20] BUADES, A., COLL, B., MOREL, J. M. "A Review of Image Denoising Algorithms, with a New One", *Multiscale Modeling & Simulation*, v. 4, n. 2, pp. 490–530, 2005.

[21] BENNETT, E. P., MASON, J. L., MCMILLAN, L. "Multispectral Bilateral Video Fusion", *IEEE Transactions on Image Processing*, v. 16, n. 5, pp. 1185–1194, May 2007.

[22] DURAND, F., DORSEY, J. "Fast Bilateral Filtering for the Display of High-dynamic-range Images". In: *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pp. 257–266, 2002.

[23] JONES, T. R., DURAND, F., DESBRUN, M. "Non-iterative, Feature-preserving Mesh Smoothing", *ACM Trans. Graph.*, v. 22, n. 3, pp. 943–949, 2003.

[24] PARIS, S., KORNPROBST, P., TUMBLIN, J., et al. "A Gentle Introduction to Bilateral Filtering and Its Applications". In: *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, 2007.

[25] ZENZO, S. D. "A note on the gradient of a multi-image", *Computer Vision, Graphics, and Image Processing*, v. 33, n. 1, pp. 116 – 125, 1986.

[26] BIGUN, J., GRANLUND, G. H., WIKLUND, J. "Multidimensional orientation estimation with applications to texture analysis and optical flow", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 13, n. 8, pp. 775–790, Aug 1991.

[27] VAN DE WEIJER, J., SCHMID, C. "Coloring Local Feature Extraction". In: Leonardis, A., Bischof, H., Pinz, A. (Eds.), *Computer Vision – ECCV 2006*, pp. 334–348. Springer Berlin Heidelberg, 2006.

[28] RITTNER, L., FLORES, F., LOTUFO, R. "New Tensorial Representation of Color Images: Tensorial Morphological Gradient Applied to Color Image Segmentation". In: *XX Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2007)*, pp. 45–52, Oct 2007.

[29] J, V. D. W. *Color Features and Local Structures in Images*. Phd. thesis, Univ. Amsterdam, Amsterdam, Holland, 2004.

[30] FISCHLER, M. A., BOLLES, R. C. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", *Commun. ACM*, v. 24, n. 6, pp. 381–395, 1981.

[31] HARTLEY, R., ZISSERMAN, A. *Multiple View Geometry in Computer Vision*. 2 ed. New York, NY, USA, Cambridge University Press, 2003.

[32] ZULIANI, M. *RANSAC for Dummies*. Technical report, Nov. 2008.

[33] BRADLEY, S. P., HAX, A. C., MAGNANTI, T. L. *Applied Mathematical Programming*. 1 ed. , Addison-Wesley, 1977.

[34] HILLIER, F. S., LIEBERMAN, G. J. *Introdução à Pesquisa Operacional*. 9 ed. Porto Alegre, Brasil, McGraw-Hill, 2013.

[35] PETRAGLIA, F., CAMPOS, R., GOMES, J., et al. "Pipeline tracking and event classification for an automatic inspection vision system". In: *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4, May 2017.

Appendix A

Underwater pipeline dataset

The dataset was obtained from 6 videos of underwater pipeline inspection tasks. Each video has a resolution of 1280x720 pixels and was recorded at 30 fps. 7808 samples were taken as is detailed on Table A.1. One sample was took every 15 frames (0.5 second) to ensure that there are samples from the whole video. Then, the samples passed through the image conditioning process from Section 2.1 (without filtering) with the same scale of I_2. Next, the pipeline borders were manually annotated (ground truth) on all samples.

Video	Number of frames	Number of samples
1	26975	1799
2	26969	1599
3	26984	1799
4	25035	1424
5	7324	282
6	26984	905
Total	140271	7808

Table A.1: Dataset composition.

Some samples are shown in Fig. A.1 and Fig. A.2.

Figure A.1: Sample pipeline images.

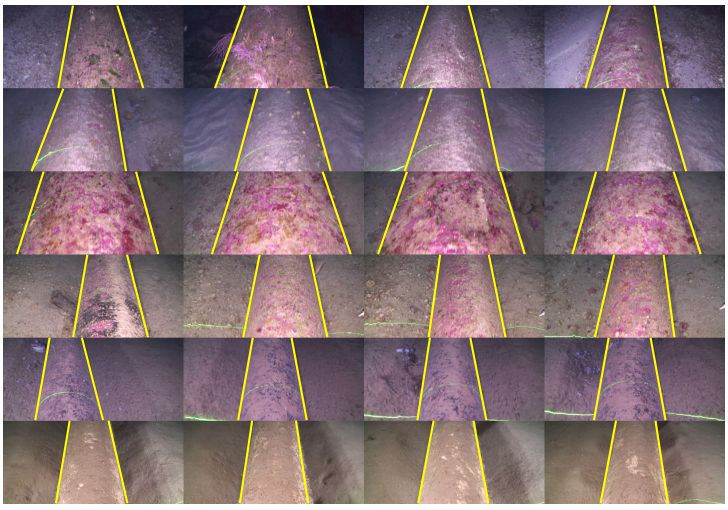


Figure A.2: Manually annotated pipeline borders.