Automatic Detection of Stork Nests on Very-High Voltage Towers

João Pedro Ribas Fernandes

Dissertation to obtain a Master Degree in Electrical and Computer Engineering

Jury
President: Prof. Fernando Duarte Nunes
Supervisor: Prof. Maria Paula dos Santos Queluz Rodrigues
Co-supervisor: Prof. Tomás Gomes da Silva Serpa Brandão
Members: Prof. Francisco de Moura e Castro Ascensão de Azevedo
Prof. Luís Eduardo de Pinho Ducla Soares

October 2012
Acknowledgments

First of all, I would like to thank my grandparents, the main pillars of my education, for the moral (and financial) support they gave me during my academic life.

I would like to show my gratitude to my supervisors, Paula Queluz and Tomás Brandão for always being available and for all their help.

I would also like to thank Albatroz Engenharia, specially Eng. Gomes Mota and Professor Francisco Azevedo, for the opportunity to work on such an interesting matter.

I also want to give a very special thanks to the most dedicated and hard-working person I know, Manuela Ribas who, during my childhood, always treated me as an adult; and to Inês Sampaio who has always been motivating me to grow up.

Finally, I would like to thank all my friends, they make my life worth living; with a special thank you to David Castro, my course mate, who has stood by me during some difficult times.
Abstract

This dissertation proposes a tool for the automatic detection of stork's nests on the top of very-high voltage towers. The developed method uses common image processing tools, and is applied to video data resulting from a regular inspection of aerial power lines.

The method starts with a search for towers on video sequences, followed by a search for nests on each tower. Both searches are, at first, applied independently to the frames extracted from the videos. Then, the temporal correlation between frames is exploited in an effort to improve the results, decreasing the number of incorrect detections. Towers detection relies on the search for straight lines that represent the well defined tower structure. The search for nests is based on their characteristics, like color and shape.

Performance evaluation of the developed method takes into account the number of successful detections and the number of false detections. On the final experiments, the method was able to detect 83% of the towers, with a false positive rate of 14%. From the detected towers, 79% of towers in risk (i.e. with nests) were identified, with a false positive rate of 43%.

The implementation of such tool targets to dispense dedicated human intervention on the count for nests. Its practical implementation would have great impact on the electrical power transportation maintenance, bringing economic and environmental benefits and helping preventing power failures and loss of quality of service.

Keywords

Very-high voltage tower detection, nest detection, edge detection, straight lines detection, motion vectors.
Resumo

Na presente dissertação propõe-se uma ferramenta para a detecção automática de ninhos de cegonha no topo de postes de muito-alta tensão. O método desenvolvido recorre a ferramentas básicas de processamento de imagem e é aplicado a sequências de vídeo obtidas durante uma inspeção regular às linhas aéreas de transporte de energia elétrica.

O método começa por procurar postes nas sequências de vídeo e, em seguida, procura ninhos de cegonha em cada um dos postes encontrados. Cada uma das procuras inicia-se por uma primeira análise trama a trama e prossegue com a exploração da relação temporal existente entre tramas consecutivas, com vista a melhorar os resultados da detecção. A detecção dos postes recorre à procura das linhas rectas que representam a estrutura bem definida do poste. A procura por ninhos é baseada nas suas características, como a cor e a forma.

A avaliação do método desenvolvido teve em conta o número de detecções bem sucedidas e o número de detecções falsas. Nos testes finais, o método detectou 83% dos postes, com uma taxa de detecções falsas de 14%. Dos postes detectados, 79% dos postes que estavam em risco (i.e. que continham ninhos) foram bem identificados, com uma taxa de falsos positivos de 43%.

A implementação deste tipo de ferramentas poderá dispensar a intervenção humana dedicada à detecção e contagem de ninhos. A sua implementação prática terá um grande impacto na manutenção da rede de transporte de energia, através da prevenção da ocorrência de falhas ou perdas de qualidade de serviço, com vantagens a nível económico e ambiental.

Palavras Chave

Detecção de postes de muito-alta tensão, detecção de ninhos, detecção de contornos, detecção de linhas rectas, vectores de movimento.
Contents

1 Introduction ............................................................. 1
  1.1 Motivation .......................................................... 3
  1.2 Problem Formulation ............................................... 3
  1.3 Main Contributions ............................................... 4
  1.4 Video Data Characterization ..................................... 5
    1.4.1 Video Acquisition ........................................... 5
    1.4.2 Training Data ................................................ 5
  1.5 Dissertation outline ............................................ 7

2 Overview of Related Work ........................................... 9
  2.1 Image Data Characterization ..................................... 11
  2.2 Tower Detection .................................................. 11
  2.3 Nest Detection .................................................... 13
  2.4 Results and Comments .......................................... 14

3 Tower Detection ...................................................... 17
  3.1 Tower detection in Still Images ................................ 19
    3.1.1 Edge Detection .............................................. 20
    3.1.2 Line Detection .............................................. 25
    3.1.3 Evaluating Probable Tower Regions ....................... 29
    3.1.4 Tower Configuration ....................................... 38
    3.1.5 Diagram of the developed system ......................... 42
  3.2 Tower Detection in Video Sequences ........................... 43
    3.2.1 Neighborhood Confirmation ................................. 43
    3.2.2 Improving Tower Configuration ............................ 48
    3.2.3 Diagram of the developed system ......................... 54
  3.3 Results and Comments ........................................... 55
    3.3.1 Still Images Analysis .................................... 55
    3.3.2 Video Sequences Analysis ................................. 56
Contents

4 Nest Detection 59
  4.1 Nest Detection in Still Images ................................. 61
    4.1.1 Nests color ........................................... 61
    4.1.2 Nests shape ........................................... 65
    4.1.3 Tower Region Concealment ................................. 67
    4.1.4 Diagram of the developed system ......................... 70
  4.2 Nest Detection in Video Sequences ............................ 71
    4.2.1 Relating Nests Detections ............................... 71
    4.2.2 Diagram of the developed system ......................... 74
  4.3 Results and Comments ........................................ 75
    4.3.1 Still images analysis .................................. 75
    4.3.2 Video sequences analysis ............................... 75

5 Final Experiments and Conclusions 79
  5.1 Final Experiments ............................................ 81
  5.2 Main conclusions ............................................. 83
  5.3 Future work ................................................ 83

A Tower Model 87

B Tower Detection In Still Images - Results 91

C Tower Detection In Video Sequences - Results 99

D Nest Detection In Still Images - Results 105

E Nest Detection In Video Sequences - Results 109
List of Figures

1.1 Tower model adopted in this dissertation, with the area of interest for nests detection surrounded by red (horizontal structure) and blue (vertical structures) lines ............................ 4
1.2 Manually selected points of interest: tower region (a) and exact tower configuration and nests location (b) .......................................................... 6

2.1 Tower with the possible nests positions outlined by dashed lines [2] .......................... 11
2.2 An example of a set of lines that would be considered as a tower ............................ 12
2.3 Detected tower (a) and its selected clustered regions (b) (area of interest in white) 13
2.4 Example of a group with 4 pixels and (a) the associated ellipse, (b) the ellipse’s major and minor axis ............................................................... 14

3.1 Original image (a); edge map generated using Otsu’s 2-level threshold ($T_1 = 0.05$ and $T_2 = 0.11$), with edges in white (b) .................................................. 20
3.2 Original Image (a); Edge Map generated using $T = 0.4$, with edges in white (b) . 21
3.3 Training results using different values of $size_{close}$ (a) and $\alpha$ (b) ................. 22
3.4 Edge map before (a) and after (b) the edge thinning ........................................ 23
3.5 Results for different values of $\beta$, where the best result is achieved with $\beta = 0.02$. . . 23
3.6 Original images (first column); Edge maps generated with initial threshold (second column) and with the adapted threshold (third column) .......................... 24
3.7 Straight line (in red) represented by the Hough Transform’s parameters .......... 25
3.8 Original Image (a); edge map generated by the method presented in section 3.1.1 (b); lines found by the Hough Transform, marked in blue (c) ....................... 26
3.9 Training results for different values of $max_{gap}$, where the best result is achieved with $max_{gap} = 0.06$ ............................................................. 26
3.10 Straight lines found in an edge map before (a) and after (b) the edge thinning . 27
3.11 Tower Structure: expected horizontal lines (in red) and vertical lines (in blue) .... 27
3.12 Example of a selected image region ............................................................. 28
3.13 Original image (a); edge map with detected lines and selected regions (b); selected regions on the original image (c) ........................................... 28
3.14 Training results for each parameter ............................................................. 30
List of Figures

3.15 Analysis of the selected regions from figure 3.13: original image inside each region (first column); edge maps (second column); lines found (third column) 31
3.16 Tower configuration, explained in table 3.1 32
3.17 Normalized scoring function 34
3.18 First test, where the best region scored 46% 35
3.19 Second test, where the only selected region scored 59% 36
3.20 Third test, where no regions were selected 36
3.21 Training results for the \( \min \) score threshold, where the best result is achieved using \( \min_{\text{score}} = 0.65 \) 37
3.22 Fourth test: false positive which scored 68% 37
3.23 Tower configuration, with a score of 66% 40
3.24 Tower configuration, with a score of 88% 40
3.25 False tower detected with a configuration score of 45% 41
3.26 Diagram of the developed algorithm for tower detection in still images 42
3.27 Original frames with the tower region selected in yellow (first row), and edge maps produced from each frame, with the detected lines (second row) 44
3.28 Frame \( n \) with the selected region in yellow (a) and Frame \( n + 1 \) with confirmation area in dashed yellow lines (b) 45
3.29 Video sequence frame by frame (first row) and as a sequence (second row) 46
3.30 Video analyzed frame by frame (first row) or as a sequence (second row) 46
3.31 Configuration of the same tower found in two frames 49
3.32 Correspondence (b) of selected points (a) found 50
3.33 Comparison between using only 2 (first row) and 5 (second row) points 51
3.34 Automatically selected points on frame 1 and their correspondence on frame 2 53
3.35 Left structure moved from the configuration in frame 1 to frame 2, same example as the figure 3.31 53
3.36 Diagram of the system developed to detect tower in video sequences system developed 54
4.1 Histogram of: tower region (top) and the hand-selected pixels (bottom); Red (first column), Green (second color) and Blue (third column) 62
4.2 Histogram of: tower region (top) and the hand-selected nest pixels (bottom); Hue (first column), Saturation (second color) and Value (third column) 62
4.3 Pixels classified as nests in white, and as background in black 63
4.4 Probability density functions for background colors (in blue) and nest colors (in red) 63
4.5 Pixels classified as nests in white 64
4.6 Map of detections (b) where isolated groups of connected white pixels represent nests candidates of the original image (a) 65
List of Figures

4.7 Example of a group with 4 pixels and the associated ellipse (a) and the ellipse’s major and minor axis (b) .................................................. 65
4.8 Map of candidates with their respective ellipse .................................. 66
4.9 Values of nests eccentricity, detected on the training sequence ............... 66
4.10 Map resulted from applying the eccentricity filter (a) and the final detection (b) with detected nests inside blue squares ......................... 67
4.11 Tower configuration (a) searching area (b) obtained from the tower configuration . 67
4.12 Tower configuration (a) and searching area (b) .............................. 68
4.13 Image with two detected nests ...................................................... 68
4.14 Example with a detected nest, a false positive and an undetected nest ........ 69
4.15 Diagram of the system developed for nest detection in still images .......... 70
4.16 Representation of the tower referential, composed by three axis: horizontal detection’s referential (in red), vertical left detection’s referential (in blue) and vertical right detection’s referential (in green) ................................. 72
4.17 Example of a list of detections on an horizontal structure .................. 73
4.18 Diagram of the developed algorithm for nest detection in a sequence of frames . 74

A.1 Tower model used and their most important measures ....................... 89
B.1 Results achieved by analyzing the training sequence of still images (part 1) .... 93
B.2 Results achieved by analyzing the training sequence of still images (part 2) .... 94
B.3 Results achieved by analyzing the training sequence of still images (part 3) .... 95
B.4 Results achieved by analyzing the training sequence of still images (part 4) .... 96
B.5 Results achieved by analyzing the training sequence of still images (part 5) .... 97
B.6 Results achieved by analyzing the training sequence of still images (part 6) .... 98
C.1 Sequence 19 analyzed as a group of still images (col. 1), as video sequence before (col. 2) and after (col. 3) the motion prediction ......................... 101
C.2 Sequence 16 analyzed as a group of still images (col. 1), as video sequence before (col. 2) and after (col. 3) the motion prediction ......................... 102
C.3 Sequence 9 analyzed as a group of still images (col. 1), as video sequence before (col. 2) and after (col. 3) the motion prediction ......................... 103
C.4 Sequence 10 analyzed as a group of still images (col. 1), as video sequence before (col. 2) and after (col. 3) the motion prediction ......................... 104
D.1 Results achieved by analyzing the training sequence of still images (part 1) .... 107
D.2 Results achieved by analyzing the training sequence of still images (part 2) .... 108
E.1 Excerpt from sequence 7, both nests detected ............................... 111
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.2</td>
<td>Excerpt from sequence 9, both nests detected</td>
<td>112</td>
</tr>
<tr>
<td>E.3</td>
<td>Excerpt from sequence 14, three detected nests and a false positive</td>
<td>113</td>
</tr>
<tr>
<td>E.4</td>
<td>Excerpt from sequence 17 with detected nest</td>
<td>114</td>
</tr>
</tbody>
</table>
List of Tables

1.1 Video data characteristics ......................................................... 5
1.2 Video sequences used on tests ..................................................... 6

2.1 Results obtained on the evaluation of the 8 provided video sequences ........ 15

3.1 Labels of lines and points of figure 3.16 ......................................... 32
3.2 Values used in the normalized scoring function of each feature .................. 34
3.3 Score of each region from figure 3.15 ............................................. 35
3.4 Values used in the normalized scoring function of each feature .................. 39
3.5 Points from frame 1, their correspondence on frame 2 and the resulting motion vectors 50
3.6 Results obtained in the test sequence (visual results are presented in appendix B) 55
3.7 Results obtained using the video sequences (*excerpt presented in appendix C) . 56
3.8 Results obtained using 10 video sequences without towers ........................ 57

4.1 Results obtained testing the sequence (presented in appendix D) ............... 75
4.2 Results obtained on the video sequences analysis (*excerpt presented in appendix E) 76
4.3 Results obtained on the test video sequences ..................................... 76

5.1 Tower detection results ................................................................. 81
5.2 Nest detection results ................................................................. 82
List of Acronyms

VHV  Very-High Voltage
RNT  Rede Nacional de Transporte
REN  Redes Energéticas Nacionais
HSV  Hue-Saturation-Value
RGB  Red-Green-Blue
PDF  Probability Density Function
FP   False Positive
FPR  False Positive Rate
HR   Hit-Rate
MVND Multivariate Normal Distributions
1.

Introduction

<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Motivation</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Problem Formulation</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Main Contributions</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Video Data Characterization</td>
<td>5</td>
</tr>
<tr>
<td>1.5 Dissertation outline</td>
<td>7</td>
</tr>
</tbody>
</table>
1. Introduction
This chapter presents the influence of storks’ nests in the electrical power transportation, as the main motivation for this dissertation. The problem to be solved and the main contributions to its resolution are explained. Finally, the data used in the development of the work related to this dissertation is described.

1.1 Motivation

Electrical power transportation throughout the national grid is based on lines that are, in more than 95% of the cases, aerial lines. Those lines are supported by different models of Very-High Voltage (VHV) towers that are subject to faults, thus requiring regular maintenance, resorting to video cameras mounted on helicopters. Storks are an example of an important risk factor to the electrical grid that may lead to power failures or quality of service loss - the nests or the stork itself, may cause short-circuits, damaging the line and injuring the stork [1].

To avoid the risks created by storks, Redes Energéticas Nacionais (REN) is required to move nests to specially designed structures, like platforms on towers or, if possible, platforms on posts placed next to towers, to attract storks. To search for nests, an inspection is performed by REN, covering 20% to 25% of the national grid. Nests inspection is currently performed by a dedicated operator in a highly time consuming operation. Ideally, it could be automatically done from the video sequences produced during a regular inspection, avoiding a human driven nests inspection. The main objective of this dissertation is to develop a system capable of automatically detecting and counting the nests in each VHV tower, using videos recorded in a regular power line inspection.

1.2 Problem Formulation

This dissertation attempts to develop an algorithm based on common image processing tools, able to automatically identify the presence of nests on the top of VHV towers. The algorithm is intended to be applied to video data acquired during a regular inspection of power lines, by cameras fixed on helicopters. The expected output is a report containing the number of towers that appear in the video, identifying which of them are in risk of malfunction due to the presence of nests. Since the video data is georeferenced, correspondence may later be made between a detected tower and its position on a map.

Although Rede Nacional de Transporte (RNT) includes different tower models, this dissertation adopted the model presented in figure 1.1. The search for nests is reduced to the most critical part of the tower, represented in figure 1.1 by the regions surrounded by the blue and red lines. This is the most critical part of the tower, since it is where the nest and the stork are closer to the isolators which may cause short-circuits.

The performance evaluation of the method is based on the Hit-Rate (HR) – percentage of towers in risk well detected – and on the False Positive Rate (FPR) – percentage of towers with
1. Introduction

no nests that were identified as in risk.

Figure 1.1: Tower model adopted in this dissertation, with the area of interest for nests detection surrounded by red (horizontal structure) and blue (vertical structures) lines

1.3 Main Contributions

A first approach to a method able to automatically identify towers and to count the number of nests present on each tower was already developed in [2]. In that work, video data was treated as a sequence of independent frames, assuming each frame as a still image. The main contribution of the presented dissertation is to explore the temporal correlation between each frame of a video sequence. It is shown that, even though this adds complexity to the algorithm, better results are actually obtained, comparatively to those presented in [2]. Secondly, even though the developed method is based on a similar approach as the previous work, some improves to the method itself were implemented. Those improves aimed to obtain a more general solution using, for instance, parameters normalized to the spatial resolution of the images and video or to the size (in image pixels) of a detected tower.
1.4 Video Data Characterization

This section describes the data used in the development of the towers and nests detection tools and in the tests performed to their validation. All of the material here mentioned was kindly provided by Albatroz Engenharia, only for the purpose of this work.

1.4.1 Video Acquisition

Video data was acquired by two different hand-operated cameras, in a moving helicopter, at a frame rate of 25 frames/s, with a spatial resolution of $1440 \times 1080$ pixels, and compressed with the H.264/AVC encoding standard [3]. Table 1.1 shows the duration and the number of towers obtained by visual inspection, presented in each video.

<table>
<thead>
<tr>
<th>File name</th>
<th>Duration (min:seg)</th>
<th>Number of towers</th>
<th>Towers with nests</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC_0138.mov</td>
<td>24:19</td>
<td>64</td>
<td>14</td>
</tr>
<tr>
<td>PIC_0140.mov</td>
<td>04:35</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>00006.mts</td>
<td>10:35</td>
<td>42</td>
<td>21</td>
</tr>
<tr>
<td>00007.mts</td>
<td>13:44</td>
<td>48</td>
<td>4</td>
</tr>
<tr>
<td>00014.mts</td>
<td>04:04</td>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1.1: Video data characteristics

Although the original videos had a frame rate of 25 frames/s, a temporal sub sampling of 5:1 was applied, in order to reduce the detection algorithm processing time. Whenever along this document two frames are told to be consecutive, they are actually 5 frames apart.

To train the algorithm parameters, only parts of the video sequences were used, which will be specified on the following subsection. The five videos will only be used on their total length (although maintaining a sub-sampling of 5:1) on the last chapter of the work, in order to assess the detection performance.

1.4.2 Training Data

Training data was extracted from the mentioned video sequences, creating a group of still images and short duration video sequences, that represent a wide range of possible scenarios. This data is intended for performing small tests, validating each step of the method and their parameters.

As still images, a group of 165 frames were extracted; they contain towers or only background landscape, as follows:

- 50 frames with background;
- 115 frames with a tower, in which:
  - 65 of them have no nests;
  - 50 of them have at least one nest, summing a total of 97 nests.
1. Introduction

In order to build the "ground truth", in each frame the existing tower was manually selected, either by using a rectangle, as shown in figure 1.2 (a), or through its exact configuration (the area of interest) and nests location, as figure 1.2 (b) shows. The "ground truth" is used to train each parameter of the detection algorithm, in which the rectangle will be used to validate a detected tower and the configuration will be used to validate the exact contour of the detected tower.

![Tower region on the image](image1)

(a) Tower region on the image

![Tower configuration and nests positions](image2)

(b) Tower configuration and nests positions

Figure 1.2: Manually selected points of interest: tower region (a) and exact tower configuration and nests location (b)

As video sequences, 20 short duration sequences showing a tower were considered; 13 of the 20 towers have at least 1 and no more than 4 nests, summing a total of 23 nests. Table 1.2 summarizes the characteristics of the used video shots.

<table>
<thead>
<tr>
<th>Sequence Number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of frames</td>
<td>41</td>
<td>7</td>
<td>15</td>
<td>81</td>
<td>15</td>
<td>11</td>
<td>41</td>
<td>41</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>Number of nests</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sequence Number</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of frames</td>
<td>18</td>
<td>12</td>
<td>55</td>
<td>11</td>
<td>63</td>
<td>41</td>
<td>47</td>
<td>51</td>
<td>71</td>
<td>7</td>
</tr>
<tr>
<td>Number of nests</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1.2: Video sequences used on tests

Contrary to what would happen if the full length video files were used, these short duration video sequences allow a fast training and evaluation of the algorithm, maintaining the diversity of scenarios.
1.5 Dissertation outline

The present dissertation is organized in the following chapters:

Chapter 1 explains the problem that motivated the work, presents the main contributions of the dissertation and describes the data used in the development of the work.

Chapter 2 overviews the work already performed on the same subject.

Chapter 3 presents and explains the algorithm developed to detect towers on both still images and video sequences. The chapter ends with some results obtained for the two cases, emphasizing the improvement obtained when using video.

Chapter 4 describes the algorithm developed to detect the presence of nests on towers, using the output from the previous chapter. Similarly to chapter 3, it ends with the results obtained in the detection of towers containing nests, using both still images and video sequences.

Chapter 5 ends the dissertation with the performance assessment of the developed detection method, using the video files at their full length. Main conclusions and suggestions for a future work are also stated at the end of the chapter.

Appendix A shows the normalized size of a tower. These values were estimated by measuring the proportions of the given model and are not exact values.

Appendices B to E present the visual results of the tests performed on the end of chapter 3 to 5, and are referred along the work.
1. Introduction
Overview of Related Work

Contents

2.1 Image Data Characterization ..................................... 11
2.2 Tower Detection .................................................. 11
2.3 Nest Detection ..................................................... 13
2.4 Results and Comments ............................................. 14
2. Overview of Related Work
This chapter overviews related work, performed by Wei Liu at Albatroz Engineering. The method developed to detect towers and nests is briefly described, as well as the results obtained using the data provided with the work.

2.1 Image Data Characterization

On the development of her work [2], Wei Liu has used frames extracted from video sequences acquired at 25 frames/s, with a spatial resolution of $704 \times 576$ pixels. The duration of the videos vary from 1 to 6 seconds. Although the frames were extracted from videos, they were treated as still images, disregarding the temporal correlation between them.

Towers were of the same model presented in figure 1.1, but the search for nests was only performed on the vertical structures. Nests were expected to be found on the positions exemplified in figure 2.1, where the searching areas are outlined by the dashed lines.

![Tower with the possible nests positions outlined by dashed lines](image)

2.2 Tower Detection

The tower detection in an image was based on the detection of straight lines that presumably form the tower structure. The line detecting method is summarized in the following steps:

1. Image transform to the Hue-Saturation-Value (HSV) color space, with each component normalized to a value between 0 and 1.

2. Split the image in two clusters:
   - area of interest, if $S < 0.12$ and $V > 0.80$;
   - area to neglect, otherwise.

3. Edge detection on the area of interest.

4. Straight lines detection on the resulting edge map.
2. Overview of Related Work

Steps 1 and 2 are justified by the fact that for a set of training images, the towers had, in the HSV space, values of S lower than 0.12 and values of V higher than 0.80. Therefore, analyzing only parts of the image that are inside that cluster removes part of the background, reducing the detection of false tower’s edges.

The edge detection was performed on the areas of interest, with the Prewitt operator [4], with a threshold given by the Otsu’s method [5], creating an edge map from which lines are detected using the Hough Transform [6]. A set of, at least, 3 straight lines would be assumed as a tower if the lines have the following characteristics:

- Two vertical lines with angles between $\frac{\pi}{4}$ rad and $\frac{3\pi}{4}$ rad, with the horizontal direction; if more than a pair of lines is found, the pair with the most symmetrical slope is chosen.

- One horizontal line with an angle between $\frac{3\pi}{4}$ rad and $\frac{5\pi}{4}$ rad with the horizontal direction, intercepted by the vertical lines selected in the previous step; if more than one line is found, it is selected the line for which the distance between its center and the interception points with the vertical lines, are more similar.

Figure 2.2 gives an example of a set of lines that would be considered as a tower. After finding the set of lines that better describe a tower, the coordinates of the points highlighted in yellow are obtained. These points are:

- $P_1$ and $P_2$ - top points of the left and right vertical lines, respectively;

- $I_1$ and $I_2$ - interceptions between the horizontal line and the left and right vertical lines, respectively;

- $C_1$ and $C_2$ - half distance points between $P_1$ and $I_1$, and $P_2$ and $I_2$, respectively.

![Figure 2.2: An example of a set of lines that would be considered as a tower](image)
Points $P_1$, $P_2$, $C_1$ and $C_2$, will be the center of the four possible regions, represented by the colored squares in figure 2.1. The size of each region is given by:

- vertical size - $2 \times \frac{P_1P_2}{AB}$;
- horizontal size - $2 \times \frac{P_1C_1}{AB}$ or $2 \times \frac{P_2C_2}{AB}$.

where $AB$ is the length, in pixels, of the line segment linking points $A$ and $B$. A search for nests is then performed on each squared region, using the method presented in the next section.

### 2.3 Nest Detection

After detecting a tower on an image, that image is transformed to gray-level (i.e., only the luminance component is kept). Using Otsu’s method [5], a threshold $th$ is computed, which is the value of luminance that maximizes the separability between higher and lower values of luminance on the image. After selecting the four regions of interest, exemplified on figure 2.3, each pixel from each region is classified as belonging to the "area of interest" or to the "area to neglect" according to:

- If $Y(i,j) > th \Rightarrow (i,j) \in \text{area of interest}$
- If $Y(i,j) < th \Rightarrow (i,j) \in \text{area to neglect}$

where $Y(i,j)$ is the luminance value of the pixel with spatial coordinates $(i,j)$.

![Detected tower and selected regions](image)

(a) Detected tower  (b) Selected regions

Figure 2.3: Detected tower (a) and its selected clustered regions (b) (area of interest in white)

Connected pixels from the area of interest are considered to be a nest candidate. In the example from figure 2.3 (b), the top-left region has two nests candidates, whilst the bottom-right region has one nest candidate. An ellipse is then associated to each candidate, having the same normalized second central moments as the group of connected pixels; the ellipse’s major and minor axis are computed. Figure 2.4 is an example of four pixels, considered as a nest candidate, and the associated ellipse.
2. Overview of Related Work

A candidate is considered to be a nest if the associated ellipse verifies the following conditions:

- Candidate on top regions:
  - the angle between the major axis and the horizontal direction is inside the interval $\left[\frac{8\pi}{9}, \frac{10\pi}{9}\right]$ rad.
  - the length of the minor axis (in pixels) is less than $\frac{iP}{9}$, where $i = 1, 2$.

- Candidate on bottom regions:
  - the angle between the major axis and the horizontal direction is inside the interval $\left[\frac{8\pi}{9}, \frac{10\pi}{9}\right]$ rad.
  - the length of the minor axis is greater than 3 pixels.

The method is not completely general. Restrictions to the size of the minor axis are not normalized neither to the spatial size of the image, nor to the size (in image pixels) of the tower. This means that for different conditions, e.g. in images from different cameras with different resolutions, the method may not achieve the expected results.

2.4 Results and Comments

The proposed algorithm was evaluated using all the frames of 8 video sequences (the same used for the algorithm development). All the video sequences contain a tower, 3 of which with one nest and 5 without nests. The evaluation considered the tower detection Hit-Rate (HR), and the tower "in risk" HR and FPR. A tower is considered to be "in risk" when it contains a nest. These indicators were computed as follows:
2.4 Results and Comments

- Tower detection HR = \( \frac{\text{# detected towers}}{\text{# existing towers}} \);
- Tower "in risk" HR = \( \frac{\text{# towers well detected as "in risk"}}{\text{# towers that are "in risk"}} \);
- Tower "in risk" FPR = \( \frac{\text{# towers wrongly detected as "in risk"}}{\text{# towers detected as in risk}} \);

- when the tower has no nests, the number of False Positive (FP) is shown instead of the FPR

Table 2.1 presents the obtained results.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Frames with a tower</th>
<th>Tower HR</th>
<th>Tower &quot;in risk&quot; HR</th>
<th>FPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>85</td>
<td>8%</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>93%</td>
<td>-</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>142 (with nest)</td>
<td>99%</td>
<td>51%</td>
<td>11%</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>67%</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>28</td>
<td>46%</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>31 (with nest)</td>
<td>6%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>7</td>
<td>38 (with nest)</td>
<td>61%</td>
<td>5%</td>
<td>0%</td>
</tr>
<tr>
<td>8</td>
<td>19</td>
<td>37%</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td><strong>Global</strong></td>
<td>382</td>
<td><strong>59%</strong></td>
<td><strong>35%</strong></td>
<td><strong>24%</strong></td>
</tr>
</tbody>
</table>

Table 2.1: Results obtained on the evaluation of the 8 provided video sequences

As mentioned, the developed algorithm is not general, mainly due to the fact that towers color are expected to be inside a pre-defined color range. Given the variety of video acquisition scenarios, due to different light conditions or towers with colors fade, the method can work better on some cases than in others. For example, it is able to detect the tower on the majority of frames from sequence 2 or 3, whilst on sequence 1 or 6 towers are detected in very few frames.

Regarding the nest detection, the algorithm was unable to detect most of them. Similarly to the tower detection method, the nest detection algorithm is not general, relying on features directly related to a number of pixels: in cases where the tower is further away from the camera, the number of pixels representing a nest is much lower, thus the method is unable to detect them.

In conclusion, these results suggest that the principles of the tower detection algorithm are a good starting point for a new method, as long as a way to generalize them is found. Regarding the nest detection, a completely new approach should be developed in order to get better results. Moreover, the temporal correlation between consecutive frames could be exploited, such as the tower movement prediction. This could be useful to increase the number of frames in which a tower is detected and to decrease the number of FP both on tower and on nest detection.
2. Overview of Related Work
Tower Detection

Contents

3.1 Tower detection in Still Images ........................................ 19
3.2 Tower Detection in Video Sequences ................................. 43
3.3 Results and Comments .................................................... 55
3. Tower Detection
3.1 Tower detection in Still Images

This chapter describes the method developed to detect the presence of a tower in a frame, and to find out its location and configuration. This step is important to the nests detection, which is the primary objective of this dissertation, as it reduces their possible location in a frame.

3.1 Tower detection in Still Images

Object detection in images is a problem deeply discussed and addressed. It has been the focus of many research, as it can be used in a wide variety of applications, such as video surveillance or process automation. Usually, objects detection in images is based on the search for features that best define them, similarly to an object identification by human vision. These features typically include object’s color, size or shape, among others.

On the context of this dissertation, the objects in question are VHV towers, which have a very distinguishing characteristic from the rest of the scenario: a VHV tower is a well defined structure, formed by straight lines. Tower detection could also be based on a color analysis; however, the same color may also be found on different objects of the image, which may lead to false detections. In fact, although all the towers have roughly the same color, the videos were recorded from a moving helicopter, resulting in a variation of the sun light incidence angle on the tower, changing its color perception. Therefore, performing a search for towers based on such feature would lead to less satisfactory results. That said, the developed detection method is limited to the search for lines that are more likely to represent towers in an image.

The developed method attempts to find, in each image, the regions that contain a tower, depending on the straight lines found. It is assumed that only one tower can be present in a frame, which is the case on the available videos. Furthermore, when a tower is detected, its configuration will be sought and saved to the following nest analysis. The method is summarized on the following four steps:

1. **Edge detection** - search for edges in an image;
2. **Line detection** - search for straight edges;
3. **Evaluate probable tower regions** - select and evaluate image regions where lines intersect in a way that resembles a tower configuration. This evaluation will be based on a more detailed detection of edges and lines;
4. **Search for the tower configuration** - if a highly probable tower region is found, a search for the lines that better describe its configuration is performed.

Each one of these steps will be detailed in the next sections.
3. Tower Detection

3.1.1 Edge Detection

For the human vision, objects can be easily detected by their contour lines, which correspond to sharp color or texture variations. These contour lines, called edges, are detected in computer vision through a process similar to the human vision, i.e., by identifying sharply brightness, color or texture changes along an image.

Edge detection is a non-trivial task, and several edge detection methods have been proposed in the literature, showing different levels of complexity and success rates. Most of them start by computing the image gradient, followed by the selection of the most evident edges, which occur when the gradient magnitude is greater than a specific threshold [7].

The Canny Edge Detector [8] is one of the most powerful edge detector, in terms of edge detection rate, edge location and noise robustness. The method starts with a Gaussian filtering, with a certain standard deviation, $\sigma_G$, to smooth the noise on the image. Then, the gradient is computed in each image pixel. The gradient norm is compared to a given threshold, to evaluate each point as an edge or not. The algorithm uses two different thresholds, $T_1$ and $T_2$, where $T_1 < T_2$. Points with gradient value above $T_2$ are considered as "strong edges", whereas points with gradient below $T_1$ are considered "non edges"; points with a gradient between $T_1$ and $T_2$ are considered "weak edges". Finally, the method performs edge linking, which considers as "edges", "weak edges" that are 8-connected to a "strong edge".

Using the Canny Edge Detector, with the 2-level threshold given by Otsu’s Method [5], results in detecting tower edges but also some less contrasting edges on the image. Figure 3.1 shows the result of applying this method to a given image, where pixels considered as "edges" are represented in white and pixels considered as "non edges" are represented in black. As tower edges are often the most contrasting edges on the image, the threshold should be increased, discarding less contrasting edges from the background (e.g. trees) but keeping the most contrasting ones, hopefully edges from a tower.

![Figure 3.1: Original image (a); edge map generated using Otsu’s 2-level threshold ($T_1 = 0.05$ and $T_2 = 0.11$), with edges in white (b)](image)
3.1 Tower detection in Still Images

In order to detect only edges of a possible tower in an image, the Matlab’s Canny Edge Detector built-in function [7] was used, with a standard deviation $\sigma_G = \sqrt{2}$, and thresholds $T_2 = T$ and $T_1 = 0.4 \times T$, starting with $T = 0.4$; afterwards, edge detection is refined with a $T$ value adapted to each image. Thresholds values are normalized using the highest value of the gradient presented in the image. The method used to adapt the threshold $T$ to a specific image will be explained later on. Figure 3.2 shows an example of an edge detection in an image with a tower, using $T = 0.4$. It can be easily seen that most of the tower edges were detected.

![Figure 3.2: Original Image (a); Edge Map generated using $T = 0.4$, with edges in white (b)](image)

After the first edge detection, with $T = 0.4$, and before the threshold adaptation, an edge thinning operation [9] is performed on the edge map, as explained next.

• **Edge Thinning**

In order to achieve better results on the line detection process, an edge thinning operator is applied to the edge map aiming to connect separate edges. Separate edges are connected through the morphological "closing" operator [7], with size $size_{close}$ pixels. Then, connected pixels are considered as a group of pixels and groups with less than $min_{area}$ pixels are neglected, where $min_{area} = \lfloor \alpha \times image_{hr} \rfloor$, and $image_{hr}$ is the image’s horizontal length (in pixels).

In order to learn the optimal values for $size_{close}$ and $\alpha$, the training sequence, mentioned in 1.4.2, was tested with different values, choosing the ones that lead to better results. These tests were only performed after the method was fully implemented, but results are shown here to follow on the explanation. Figure 3.3 summarizes the training results, where:

- Hit-Rate (%) = $\frac{\# \text{detected towers}}{\# \text{existing towers}} \times 100$;

- False Positives (%) = $\frac{\# \text{wrongly detected tower}}{\# \text{detected towers}} \times 100$.  

21
3. Tower Detection

(a) Best results achieved with \( \text{size}_{\text{close}} = 5 \)

(b) Best results achieved with \( \alpha = 0.15 \)

Figure 3.3: Training results using different values of \( \text{size}_{\text{close}} \) (a) and \( \alpha \) (b)

Applying the edge thinning, results in an edge map with less but thicker edges. Figure 3.4 shows a comparison between an edge map obtained before and after applying the edge thinning.

(a) Detected edges without edge thinning  
(b) Detected edges with the edge thinning

Figure 3.4: Edge map before (a) and after (b) the edge thinning

After the edge thinning is applied, the threshold \( T \) will be adapted and the edges will be detected again. This aims to increase the robustness of the method, making it less dependent on the image color contrast.

- **Threshold adaptation**

Although on the example of figure 3.2 the initial threshold, \( T = 0.4 \), has conducted to an adequate detection of the tower edges, it was also verified that it does not work so well on all the images. Using \( T = 0.4 \) will not result in detecting a significant number of edges in images with low contrast. To circumvent this problem, the threshold is reduced until the percentage of image pixels
that are detected as edges is at least $P_{\text{min}}$. The threshold is adapted to each image, according to the following iterative method:

1. define $P_{\text{min}} = \beta \times \text{image\_pixels}$, where $\text{image\_pixels}$ is the total number of pixels in an image.
2. initial threshold $T = 0.4$;
3. edge detection using $T$;
4. edge thinning;
5. verify the percentage of image pixels that were considered as "edges":
   - if above or equal to $P_{\text{min}}$, end;
   - if below $P_{\text{min}}$, change $T$ to $T = T - 0.01$ and return to step 3.

Similarly to the learning of optimal values for other parameters, the training sequence was tested with different values of $\beta$. Training results are plotted on figure 3.5, where:

- Hit-Rate (%) = $\frac{\# \text{detected towers}}{\# \text{existing towers}} \times 100$;
- False Positives (%) = $\frac{\# \text{wrongly detected tower}}{\# \text{detected (true or false) towers}} \times 100$.

![Figure 3.5: Results for different values of $\beta$, where the best result is achieved with $\beta = 0.02$](image)

Figure 3.6 presents some results of the threshold adaptation method with $P_{\text{min}} = 2\%$ of the total number of pixels of an image. In this figure, the first column shows the original images, the second column shows the results obtained from edge detection using the initial threshold, $T = 0.4$, and edge thinning, and the third column shows the results obtained with the edge detection using the adapted threshold and edge thinning.

23
Figure 3.6: Original images (first column); Edge maps generated with initial threshold (second column) and with the adapted threshold (third column)
3.1 Tower detection in Still Images

3.1.2 Line Detection

Ideally, the method described in subsection 3.1.1 would produce a bit map identifying the most strong edges on an image. The next step is to decide which of these edges belong to a tower. As already mentioned, towers have a very uniform structure formed by straight lines, hence filtering the edge map for straight edges seems to be a good approach to follow for tower detection.

The Hough Transform [6] is a very common and efficient method to detect lines in an image. It defines lines using polar coordinates: \( \rho \), lowest distance between the origin of the image and the line; and \( \theta \), the angle of the vector (perpendicular to the line), from the origin to the line. Figure 3.7 shows a line defined by \((\rho, \theta)\).

The Hough Transform is a voting procedure, aiming to select which pairs of \((\rho, \theta)\) are most likely to describe straight lines, depending on the number of points of interest that the line superimpose. Applied to the edge map created with the previous method, the Hough Transform will give to each line a score proportional to the number of edge pixels that are superimposed to that line. The method allows to reduce the search for lines with a certain minimum length, \(\text{min}\_\text{length}\), and to fill gaps, smaller than \(\text{max}\_\text{gap}\) (in detected lines) on the assumption they result from the same line that was not completely detected by the edge detection algorithm. After scoring every possible line on an edge map, one can choose \(N_{\text{lines}}\) lines with the highest score, which will represent the \(N_{\text{lines}}\) edges that are most likely to represent straight lines.

Applying Matlab’s Hough Transform built-in function [7] to an example edge map, with \(\text{min}\_\text{length}\) as 15% of the image’s horizontal length and with \(N_{\text{lines}} = 20\), results in the twenty highest scored lines presented in blue in figure 3.8.
3. Tower Detection

![Original image](image1)

![Edge map](image2)

![Detected lines](image3)

Figure 3.8: Original Image (a); edge map generated by the method presented in section 3.1.1 (b); lines found by the Hough Transform, marked in blue (c)

Again, the method was tested with the training sequence using different values for the \( \text{max}_{\text{gap}} \) parameter, defined as a percentage of the horizontal length of the image. The test was conducted using values of \( \text{max}_{\text{gap}} \) between 2\% and 20\%; the training results are presented in figure 3.9.

![Graph](image4)

Figure 3.9: Training results for different values of \( \text{max}_{\text{gap}} \), where the best result is achieved with \( \text{max}_{\text{gap}} = 0.06 \)

Figure 3.10 shows the difference between searching for lines in an edge map generated with (b), and without (a), the edge thinning. In the latter case, the detected lines more often cover the full length of the tower edges, which confirms the usefulness of the edge thinning procedure.
3.1 Tower detection in Still Images

![Detected lines (in blue) without edge thinning](image1)

(a) Detected lines (in blue) without edge thinning

![Detected lines (in blue) with edge thinning](image2)

(b) Detected lines (in blue) with edge thinning

Figure 3.10: Straight lines found in an edge map before (a) and after (b) the edge thinning

The main objective of searching for lines within an image, is to find a combination of lines that most likely correspond to part of the tower structure. Given the tower structure (figure 3.11), the lines marked in blue and red are expected to be the easiest to be detected, as they define the exterior contour of the tower.

![Tower Structure: expected horizontal lines (in red) and vertical lines (in blue)](image3)

Figure 3.11: Tower Structure: expected horizontal lines (in red) and vertical lines (in blue)

In a typical image, many intersected lines may be found and one cannot tell at first which set of lines, if any, refers to a tower. In a first global search approach, the lines found in the image are divided in two groups:

- **Vertical** - if they have an angle between $\frac{\pi}{4}$ rad and $\frac{3\pi}{4}$ rad with the horizontal direction;

- **Horizontal** - otherwise.

Then, a more detailed search will be performed in the image regions that have an horizontal line intercepted by, at least, one vertical line. For each intersected horizontal line, a rectangular image region is selected, by surrounding the line and by adding a margin proportional to the expected
3. Tower Detection

size of a tower, represented by the size (in pixels) of that horizontal line, \( L_H \). This margin is computed as follows:

- **Horizontal margin** - \( \frac{L_H}{10} \), added to each end of the line;
- **Vertical top margin** - \( \frac{L_H}{3} \), added above the line;
- **Vertical bottom margin** - \( \frac{L_H}{6} \), added below the line.

Accordingly, the selected rectangular image region will be centered on the respective horizontal line center, and will be sized \((L_H + 2 \times \frac{L_H}{10}) \times (\frac{L_H}{3} + \frac{L_H}{6})\). Figure 3.12 shows a fictitious example of a selected image region (in yellow) when an horizontal line (in red) is intersected by a vertical line (in blue).

![Figure 3.12: Example of a selected image region](image)

This selection is an attempt to keep only the interesting region of a tower for further analysis. Figure 3.13 shows an example of the selected image regions. After the edge and the line detection, five horizontal lines, intercepted by vertical lines, were detected, therefore five regions were selected.

![Figure 3.13: Original image (a); edge map with detected lines and selected regions (b); selected regions on the original image (c)](image)

The following analysis, applied to each region, is explained in the next subsection.
3.1 Tower detection in Still Images

3.1.3 Evaluating Probable Tower Regions

The algorithm described in section 3.1.2 selects the image regions which more likely contain a tower, based on the intersected lines. The next step is to decide which of the regions, if any, actually contains a tower. This decision is based on re-analyzing each region, detecting more detailed edges and lines in that region. This re-analysis uses tools similar to those described in sections 3.1.1 and 3.1.2, with some differences listed below:

1. **Edge detection**:
   - threshold adaptation will be performed using $P_{min}$, as before, and also using a $P_{max}$. After the first edge detection, with the initial threshold $T = 0.4$, the percentage of image pixels that are selected as edges is obtained:
     - if that percentage is below or equal to $P_{min} = \beta_1 \times image\_pixels$, $T$ will decrease until the percentage is above $P_{min}$;
     - if that percentage is above or equal to $P_{max} = \beta_2 \times image\_pixels$, $T$ will increase until the percentage is below $P_{max}$ (this aims to avoid appearance of too many edges);
   - the value of $min\_area$, in the edge thinning, is increased.

2. **Line detection**:
   - The Hough Transform is applied 3 times, limiting the values of $\theta$ in three distinct ways:
     - Horizontal lines - $\theta \in [-90^\circ, -60^\circ] \cup [60^\circ, 90^\circ]$ (lines with angles between $\frac{5\pi}{6}$ and $\frac{7\pi}{6}$ rad with the horizontal direction);
     - Vertical left lines - $\theta \in [0^\circ, 45^\circ]$ (lines with angles between $\frac{\pi}{2}$ and $\frac{3\pi}{4}$ rad with the horizontal direction);
     - Vertical right lines - $\theta \in [-45^\circ, 0^\circ]$ (lines with angles between $\frac{\pi}{4}$ and $\frac{\pi}{2}$ rad with the horizontal direction).

If, as expected, a selected region includes a tower, when the edge detector is applied for the second time, more detailed edges from the tower will appear, hence the changes to $min\_area$ referred.

In the line detection, instead of performing a general search for straight lines and then classifying them according to their slope, three specific searches are performed, strictly searching for horizontal, vertical left or vertical right lines. This aims to detect more specific lines from the tower configuration.

As before, the optimum values for parameters $\beta_1$, $\beta_2$, $min\_area$ and $max\_gap$ were learned through the same training procedure, with results shown in figure 3.14.
3. Tower Detection

Figure 3.14: Training results for each parameter

(a) Best results achieved with $\beta_1 = 0.07$

(b) Best results achieved with $\beta_2 = 0.1$

(c) Best results achieved with $\text{min}_{\text{area}} = 0.8$

(d) Best results achieved with $\text{max}_{\text{gap}} = 0.145$

Figure 3.15 presents the results obtained after the re-analysis of each selected region on the image of figure 3.13. The first column represents the part of the original image contained inside each selected region; the second column presents the new edge map; on the third column the lines found are superimposed with the edge map, with horizontal lines in red, vertical lines from the left part of the tower in blue, and vertical lines from the right part of the tower in green.
It is visible on figure 3.15 that the resulting lines represent more accurately towers’ principal contour lines. Although all the five regions from the given example fit the tower, some are more consistent (e.g., the first two regions) than others (e.g., the third region). On the other hand, the last two regions do not include all the tower’s interesting regions. The objective is now to decide which region is more likely to fit a tower: a score will be given to each region that will depend on how similar are the detected lines to the ones presented in figure 3.16.
In each region, the algorithm will give a score to each line, which will represent how similar a line is to a line from the expected set of 4 lines that would best define the tower structure; then, the lines will determine the region score:

- 2 horizontal lines from the horizontal structure, H1 and H2;
- 1 vertical line from the left part of the tower, L, that could be either L1 or L2, intersecting the horizontal lines H1 and H2 at points ITL and IBL, respectively;
- 1 vertical line from the right part of the tower, R, that could be either R1 or R2, intersecting the horizontal lines H1 and H2 at points ITR and IBR, respectively.
3.1 Tower detection in Still Images

The scoring method evaluates three line features Size, Position, and Slope, giving a normalized score according to the function plotted on figure 3.17. The scoring method works as follows, where $A_{\text{center}}$, $A_{\text{left point}}$ and $A_{\text{right point}}$ are the center, the leftmost and the rightmost points of line A:

1. select horizontal line $i$, $H_{1i}$ (it is assumed to be line H1), and obtain $\text{Size}(H1)$, to be used in table 3.2;

2. select horizontal line $j \neq i$, $H_{2j}$, (it is assumed to be line H2);

3. score every $k$ vertical left line, assuming it is referring to the line L1 or L2, according to:
   - $\text{Score}(L_k) = \frac{\text{Position}(ITL_k)+\text{Position}(IBL_k)}{2}$
     - where $\text{Position}(ITL_k)$ is the distance between $ITL_k$ and $H_{1i_{\text{left point}}}$,
     - and $\text{Position}(IBL_k)$ is the distance between $IBL_k$ and $H_{2j_{\text{left point}}}$,
   - the score of the set of vertical lines, $\text{Score}(L)$, will be the mean of $\text{Score}(L_k)$;

4. score every $k$ vertical right line, assuming it is referring to the line R1 or R2, according to:
   - $\text{Score}(R_k) = \frac{\text{Position}(ITR_k)+\text{Position}(IBR_k)}{2}$
     - where $\text{Position}(ITR_k)$ is the distance between $ITR_k$ and $H_{1i_{\text{right point}}}$,
     - and $\text{Position}(IBR_k)$ is the distance between $IBR_k$ and $H_{2j_{\text{right point}}}$,
   - the score of the set of vertical lines, $\text{Score}(R)$, will be the mean of $\text{Score}(R_k)$;

5. score the line $H_{2j}$ according to:
   - $\text{Score}(H_{2j}) = \frac{\text{Size}(H_{2j})/3+\text{Position}(H_{2j})/3+\text{Slope}(H_{2j})/3+\text{Score}(L)+\text{Score}(R)}{3}$
     - where $\text{Position}(H_{2j})$ is the distance between $H_{1i_{\text{center}}}$ and $H_{2j_{\text{center}}}$.

6. repeat from step 2 with the next horizontal line, until all the horizontal lines are tested;

7. after all the horizontal lines are tested as a H2 line, the score of the set of horizontal lines, $\text{Score}(H1_i)$, will be the mean of $\text{Score}(H_{2j})$;

8. repeat from step 1 with the next horizontal line, until all the horizontal lines are tested;

9. after all the horizontal lines are tested as a H1 line, the region score will be:
   - $\text{Score(\text{region})} = \text{Max}(\text{Score}(H1_i)), i = 1...N$
     - where $N$ is the number of horizontal lines on the region.

Each feature is normalized to its expected maximum possible value. To do so, a Normalized scoring function was created. This function is plotted in figure 3.17, with the following points of interest:
3. Tower Detection

- **Max** is the expected value of the feature, obtained from table 3.2;

- $2 \times \Delta_{\text{max}}$ is the size of a trust interval, centered in Max, inside which the feature has the value 1;

- $\Delta_{\text{min}}$ is the size of a penalizing interval, inside which the feature has normalized value varying from 1 to 0;

- $\text{Max} \pm (\Delta_{\text{max}} + \Delta_{\text{min}})$ is the interval in which the feature will be normalized.

![Figure 3.17: Normalized scoring function](image)

The expected values of each feature are proportional to the size of a tower, taken from the model, and having as reference the H1 line. Table 3.2 presents the values used in the normalized scoring function of each feature, where the dependence on H1 is evident. These values are visually represented in appendix A.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Max value</th>
<th>$\Delta_{\text{max}}$</th>
<th>$\Delta_{\text{min}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size(H2)</td>
<td>$\frac{1}{2} \times \text{Size}(H1)$</td>
<td>0</td>
<td>$\frac{1}{4} \times \text{Size}(H1)$</td>
</tr>
<tr>
<td>Position(H2)</td>
<td>$\frac{1}{4} \times \text{Size}(H1)$</td>
<td>0</td>
<td>$\frac{1}{4} \times \text{Size}(H1)$</td>
</tr>
<tr>
<td>Slope(H2)</td>
<td>Slope(H1)</td>
<td>0</td>
<td>$\frac{1}{7} \times \text{Slope}(H1)$</td>
</tr>
<tr>
<td>Position(TTL)</td>
<td>$\frac{1}{4} \times \text{Size}(H1)$</td>
<td>$\frac{1}{4} \times \text{Size}(H1)$</td>
<td>$\frac{1}{4} \times \text{Size}(H1)$</td>
</tr>
<tr>
<td>Position(IBL)</td>
<td>$\frac{1}{4} \times \text{Size}(H2)$</td>
<td>$\frac{1}{4} \times \text{Size}(H2)$</td>
<td>$\frac{1}{4} \times \text{Size}(H2)$</td>
</tr>
<tr>
<td>Position(TIR)</td>
<td>$\frac{1}{4} \times \text{Size}(H1)$</td>
<td>$\frac{1}{4} \times \text{Size}(H1)$</td>
<td>$\frac{1}{4} \times \text{Size}(H1)$</td>
</tr>
<tr>
<td>Position(IBR)</td>
<td>$\frac{1}{4} \times \text{Size}(H2)$</td>
<td>$\frac{1}{4} \times \text{Size}(H2)$</td>
<td>$\frac{1}{4} \times \text{Size}(H2)$</td>
</tr>
</tbody>
</table>

Table 3.2: Values used in the normalized scoring function of each feature

For each one of the selected regions from figure 3.15, table 3.3 presents the scoring values (in percentages) obtained. Each score may be interpreted as a measure of the probability that the
respective region contains a tower. Results suggest that the second region is the more likely to represent a tower on the image.

<table>
<thead>
<tr>
<th>Region</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score</td>
<td>87%</td>
<td>88%</td>
<td>84%</td>
<td>64%</td>
<td>53%</td>
</tr>
</tbody>
</table>

Table 3.3: Score of each region from figure 3.15

The similarity between the scores of the first and the second regions is justified by the fact that they correspond basically to the same area in the image; both scores are high because that area is showing the area of interest of a tower. The third region also scored a high value because all the tower is included in that region. The fourth region still scored an average value because, even though not all the tower is included in the area, most of it is. The last region scored the lowest value, as expected, as it only includes part of the tower. This proves that the scoring system works as expected, selecting the region which is more likely to include a tower.

This scoring system also allows to choose whether an image contains a tower or not, by defining a minimum score, $\text{min} \text{score}$. If the score of the best scored region is below $\text{min} \text{score}$, it will be assumed that the image does not contain any tower. Some examples of image regions that do not contain any tower are shown in figures 3.18 to 3.20.
3. Tower Detection

![Images of original image, edge map, selected region, and region analysis with annotations](image-url)

Figure 3.19: Second test, where the only selected region scored 59%

In the first test, two regions were selected and re-analyzed. Looking at the original image it is clear that those regions are not referring to an area of interest of the tower. Yet, re-analyzing those regions, it is seen that there are lines intersecting each other in a way that are interpreted by the system as being a possible tower configuration. Nevertheless, the best scored region got a score of 46%, which suggests that the threshold $min_{score}$ should be higher than that. In the second test, something similar happens, but this time only one region is selected. Finally, in the third test...
3.1 Tower detection in Still Images

no lines are intersecting, therefore no region was selected and no tower was detected on the image.

Testing the training sequence with different values of $\text{min score}$ (results are presented on figure 3.21) suggests that the minimum score to assume that the image contains a tower should be 65%.

![Training results for the min score threshold](image)

Figure 3.21: Training results for the $\text{min score}$ threshold, where the best result is achieved using $\text{min score} = 0.65$

Even with this threshold, there are cases where a tower is detected in images that do not show any tower (false positive); figure 3.22 is an example of such cases. The occurrence of false positives will be decreased when exploiting the temporal correlation between consecutive frames in a video sequence. This will be presented in section 3.2.

![Fourth test: false positive which scored 68%](image)

(a) Original image  (b) Edge map and lines found  (c) One of the selected areas

Figure 3.22: Fourth test: false positive which scored 68%

The next step is to define the set of 6 lines that best describe the tower configuration, represented in figure 3.16.
3. Tower Detection

3.1.4 Tower Configuration

Selecting the exact position and the limits of the tower is an important step before nests detection, as reducing the nest searching area will also decrease the number of false positives. That said, when assumed that a tower exists in an image, one should try to find the exact set of 6 lines that best describe L1, L2, H1, H2, R1 and R2, drawn in figure 3.16, from the lines found on the selected region of the image. A similar method to the one used to score the regions was developed, in order to find the best configuration of the tower. The method works as follows:

1. select horizontal line $i$, $H_1$, (it is assumed to be line $H_1$) and obtain $\text{Size}(H_1)$ and $\text{Slope}(H_1)$;

2. select horizontal line $j \neq i$, $H_2$ (where it is assumed to be line $H_2$), and compute:
   - $\text{Score}(H) = \frac{\text{Size}(H_2) + \text{Position}(H_2) + \text{Slope}(H_2)}{3}$
     where $\text{Position}(H_2)$ is the distance between $H_{1,\text{center}}$ and $H_{2,\text{center}}$

3. choose the best pair of vertical left lines, assuming they are referring to the pair L1 and L2, by computing the score of every pair, according to:
   - $\text{Score}(L) = \frac{\text{Score}(L_1) + \text{Score}(L_2) + \text{Position}(IL)}{5}$, where:
     - $\text{Score}(L_1) = \text{Position}(IT_{1L}) + \text{Position}(IB_{1L})$
       * $\text{Position}(IT_{1L})$ is the distance between $IT_{1L}$ and $H_{1,\text{left point}}$
       * $\text{Position}(IB_{1L})$ is the distance between $IB_{1L}$ and $H_{2,\text{left point}}$
     - $\text{Score}(L_2) = \text{Position}(IT_{2L}) + \text{Position}(IB_{2L})$
       * $\text{Position}(IT_{2L})$ is the distance between $IT_{2L}$ and $H_{1,\text{left point}}$
       * $\text{Position}(IB_{2L})$ is the distance between $IB_{2L}$ and $IB_{1L}$
     - $\text{Position}(IL) = \frac{\text{Position}(IL)_x + \text{Position}(IL)_y}{2}$,
       * $\text{Position}(IL)_x$ is the horizontal projection of the line segment linking IL to IT_{1L}
       * $\text{Position}(IL)_y$ is the vertical projection of the line segment linking IL to IT_{1L}
       * if there is no interception between L1 and L2, point IL will be the top-most point of L1 and L2.

4. choose the best pair of vertical right lines, assuming they are referring to the pair R1 and R2, by computing the score of every pair, according to:
   - $\text{Score}(R) = \frac{\text{Score}(R_1) + \text{Score}(R_2) + \text{Position}(IR)}{5}$, where:
     - $\text{Score}(R_1) = \text{Position}(IT_{1R}) + \text{Position}(IB_{1R})$
       * $\text{Position}(IT_{1R})$ is the distance between $IT_{1R}$ and $H_{1,\text{right point}}$
       * $\text{Position}(IB_{1R})$ is the distance between $IB_{1R}$ and $H_{2,\text{right point}}$
3.1 Tower detection in Still Images

- Score(R2) = Position(IT2R) + Position(IB2R),
  * Position(IT2R) is the distance between IT2R and H1rightpoint
  * Position(IB2R) is the distance between IB2R and IB1R

\[
Position(IR) = \frac{Position(IR)_{x} + Position(IR)_{y}}{2},
\]
  * Position(IR)_{x} is the horizontal projection of the line segment linking IR to IT1R
  * Position(IR)_{y} is the vertical projection of the line segment linking IR to IT1R
  * If there is no interception between R1 and R2, point IR will be the top most point of R1 and R2.

5. compute the score of line H2_j according to:

\[
Score(H2_j) = \frac{Score(H) + Score(L) + Score(R)}{3}
\]

6. repeat from step 2 with the next horizontal line, until all the horizontal lines are tested;

7. compute the score of the line H1_i according to:

\[
Score(H1_i) = Max(Score(H2_j));
\]

8. repeat from step 1 with the next horizontal line, until all the horizontal lines are tested;

9. after all the horizontal lines are tested, the configuration score will be:

\[
Conf_{score} = Max(Score(H1_i)).
\]

10. save the lines that form the set with score Conf_{score} as the Tower Configuration.

Table 3.4 shows the values used in the normalized scoring function (figure 3.17), of each line feature, and are visually represented in appendix A.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Max value</th>
<th>(\Delta_{max})</th>
<th>(\Delta_{min})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size(H2)</td>
<td>(\frac{1}{2} \times size(H1))</td>
<td>0</td>
<td>(\frac{1}{2} \times size(H1))</td>
</tr>
<tr>
<td>Position(H2)</td>
<td>(\frac{1}{2} \times size(H1))</td>
<td>0</td>
<td>(\frac{1}{2} \times size(H1))</td>
</tr>
<tr>
<td>Slope(H2)</td>
<td>Slope(H1)</td>
<td>0</td>
<td>(\frac{1}{2} \times slope(H1))</td>
</tr>
<tr>
<td>Position(IT1L)</td>
<td>0</td>
<td>0</td>
<td>(\frac{1}{2} \times size(H1))</td>
</tr>
<tr>
<td>Position(IB1L)</td>
<td>(\frac{1}{2} \times size(H2))</td>
<td>0</td>
<td>(\frac{1}{2} \times size(H2))</td>
</tr>
<tr>
<td>Position(IT2L)</td>
<td>(\frac{1}{2} \times size(H1))</td>
<td>0</td>
<td>(\frac{1}{2} \times size(H1))</td>
</tr>
<tr>
<td>Position(IB2L)</td>
<td>(\frac{1}{2} \times size(H2))</td>
<td>0</td>
<td>(\frac{1}{2} \times size(H2))</td>
</tr>
<tr>
<td>Position(IL1)</td>
<td>(\frac{1}{2} \times dist(IT1L, IT2L))</td>
<td>0</td>
<td>(\frac{1}{2} \times dist(IT1L, IT2L))</td>
</tr>
<tr>
<td>Position(IL2)</td>
<td>(\frac{1}{2} \times dist(IT1L, IT2L))</td>
<td>0</td>
<td>(\frac{1}{2} \times dist(IT1L, IT2L))</td>
</tr>
<tr>
<td>Position(IT1R)</td>
<td>0</td>
<td>0</td>
<td>(\frac{1}{2} \times size(H1))</td>
</tr>
<tr>
<td>Position(IB1R)</td>
<td>(\frac{1}{2} \times size(H2))</td>
<td>0</td>
<td>(\frac{1}{2} \times size(H2))</td>
</tr>
<tr>
<td>Position(IT2R)</td>
<td>(\frac{1}{2} \times size(H1))</td>
<td>0</td>
<td>(\frac{1}{2} \times size(H1))</td>
</tr>
<tr>
<td>Position(IB2R)</td>
<td>(\frac{1}{2} \times size(H2))</td>
<td>0</td>
<td>(\frac{1}{2} \times size(H2))</td>
</tr>
<tr>
<td>Position(IR)</td>
<td>(\frac{1}{2} \times dist(IT1R, IT2R))</td>
<td>0</td>
<td>(\frac{1}{2} \times dist(IT1R, IT2R))</td>
</tr>
<tr>
<td>Position(IR)</td>
<td>(\frac{1}{2} \times dist(IT1R, IT2R))</td>
<td>0</td>
<td>(\frac{1}{2} \times dist(IT1R, IT2R))</td>
</tr>
</tbody>
</table>

Table 3.4: Values used in the normalized scoring function of each feature
3. Tower Detection

Figure 3.23 shows the configuration of the tower, found on the region 2 of figure 3.15, that results after applying the described method.

Figure 3.23: Tower configuration, with a score of 66%

In this case the configuration found is quite accurate, aside the left part of the tower. The final score given by the system to this configuration was 66%. Figure 3.24 shows an example in which the tower was detected more accurately, with a score of 88%.

Figure 3.24: Tower configuration, with a score of 88%
3.1 Tower detection in Still Images

The global score of the detected configuration and the scores of each part of the configuration, Score(L), Score(H) and Score(R), are saved, and will be used when exploiting the correlation between frames of a video sequence. The score of the tower configuration will also be used later on, to detect false positives, as shown on the example of figure 3.25, where a tower was wrongly detected with a configuration score of 45%.

Figure 3.25: False tower detected with a configuration score of 45%
3. Tower Detection

3.1.5 Diagram of the developed system

Figure 3.26 summarizes, in a diagram, the method developed to detect towers in still images.

Figure 3.26: Diagram of the developed algorithm for tower detection in still images
3.2 Tower Detection in Video Sequences

In this section we extend the method, presented in section 3.1, to the case of video sequences. With this, we expect to improve the results, by increasing the number of detected towers and by reducing the number of false positives: if a tower was detected in a group of consecutive frames, the probability that a tower also exists in the following frame will be high; on the contrary, if in a group of consecutive frames a tower was detected in just one of the frames, that detection is likely to be a false positive.

The tower detection in still images was based on two searches, detailed in section 3.1:

1. search for the most probable tower regions in an image, detecting edges and lines (subsection 3.1.1 and 3.1.2);
2. re-analyze edges and lines in each probable tower region (subsection 3.1.3).

It was also verified that, for most of the undetected towers, the reason was a failed detection at the first search. The method developed to detect towers in video sequences uses the most probable tower region selected on a frame during the first search, to perform the second search on the following frames. As it will be shown, this procedure allows either to detect a miss detected tower in a new frame, either to lower the confidence of detected tower, in previous frames.

The method developed to detect towers in a video sequence can be summarized in the following steps:

1. Analyze the whole sequence as a group of still images, using the method presented in section 3.1.
2. Neighborhood confirmation - Confirm detections in each frame, using the detections on following frames.
3. Improving tower configuration - Improve the tower configuration inside each group of frames containing the same tower.

3.2.1 Neighborhood Confirmation

Analyzing a video sequence frame by frame, as uncorrelated still images, will result in towers detected in some frames but, most likely, not on every frame where that tower appears, as exemplified in figure 3.27; the method failed to detect any probable tower region in both the second and third frames, as no intersection between horizontal and vertical lines was detected. The main objective here is to re-analyze those frames, in an attempt to find miss detected towers, confirming the detection on the first frame; or to find that the detection on the first frame was a false positive.
3. Tower Detection

![Figure 3.27: Original frames with the tower region selected in yellow (first row), and edge maps produced from each frame, with the detected lines (second row)](image)

All the videos were acquired by a moving camera, so the same tower will change its position from frame to frame. If the camera was fixed on the helicopter, the tower "movement" would be quite constant and moving towards the inferior left corner of the image. However, cameras were hand-operated resulting in an unpredictable tower "movement". Although the direction of the tower motion vector cannot be predicted, it is assumed that its norm does not have a high value. Therefore, the detection on Frame $n$, if correct, should be confirmed on Frame $n \pm 1$, and in an image area centered on the same coordinates as the region of interest from Frame $n$. Figure 3.28 shows an example of the confirmation area on Frame $n + 1$, corresponding to the region detected on Frame $n$. In order to cope with the camera movement between frames, the searched area on Frame $n + 1$ is 10% longer, in both horizontal and vertical directions, than the region of interest on Frame $n$. 

---

**Figure 3.27:** Original frames with the tower region selected in yellow (first row), and edge maps produced from each frame, with the detected lines (second row)
3.2 Tower Detection in Video Sequences

![Image](image_url)

Figure 3.28: Frame $n$ with the selected region in yellow (a) and Frame $n + 1$ with confirmation area in dashed yellow lines (b)

When confirming a detection between Frame $n$ and Frame $n \pm 1$, several scenarios may occur:

1. Frame $n \pm 1$ does not have any detected tower. Applying the method from subsection 3.1.3, using the confirmation area, leads to:

   (a) No tower detected, or
   
   (b) A tower is detected.

2. Frame $n \pm 1$ already has a detected tower, which is:

   (a) Outside the confirmation area, or
   
   (b) Inside the confirmation area.

Figure 3.27 is an example of the first case (1a). A tower was detected on the first frame but not on the following frames. As mentioned previously, the major cause of detection failure comes from not detecting any probable tower region on the first search. Therefore, the confirmation area obtained from the region selected on Frame $n$ will be used to find the tower on Frame $n + 1$. Performing the second search in the confirmation area, results in detecting a tower with a score of 88%. Furthermore, using a confirmation area built from the new region found on Frame $n + 1$, on Frame $n + 2$, will also result in detecting the tower. This allows not only to say that detection on the first frame is correct with more certainty, but also to detect towers on frames that were not detected before. Results are shown in figure 3.29.

The first case will also allow to detect false positives. Searching for towers on the frames represented in figure 3.30, resulted in a false detection on the second frame. Using the confirmation area, built from the region selected in Frame 2, to perform the second search on Frame 1 and Frame 3, did not result in the detection of a tower, allowing to discard the false detection on the second frame.
3. Tower Detection

Figure 3.29: Video sequence frame by frame (first row) and as a sequence (second row)

Figure 3.30: Video analyzed frame by frame (first row) or as a sequence (second row)
The second case happens when using Frame \( n \pm 1 \) to confirm a detection on Frame \( n \), and when a tower was also detected in Frame \( n + 1 \) or Frame \( n - 1 \). One of two scenarios may happen:

1. the detection on Frame \( n \pm 1 \) is inside the confirmation area, which is the best case because, more likely, the same tower was well detected in both frames;

2. the detection on Frame \( n \pm 1 \) is outside the confirmation area. If this is the case, a search for a tower will be performed on Frame \( n \pm 1 \), inside the confirmation area. If a new tower is found, with an higher score than the previously detected tower, the previous one is replaced by the new one. If a new tower is found, but with a lower score, detection in Frame \( n \) is discarded.

The following pseudo-code represents the algorithm applied to a video sequence, pre-analyzed as a sequence of still images, in order to confirm or to detect new towers (pseudo-code is presented due to the existence of loops):

```plaintext
Data: Video sequence with N frames, analyzed as a group of independent still images
Result: Video sequence with confirmed towers and new towers detected
% Forward search;
    n=1;
    while n<N do
        if frame n has a detected tower then
            if frame n + 1 has a detected tower then
                if tower on frame n + 1 is inside the confirmation area then
                    tower on frame n is considered valid;
                else
                    search for a new tower inside the confirmation area;
                    if new tower is detected, with an higher score then
                        replace tower on frame n + 1;
                        tower on frame n is considered valid;
                    end
                end
            else
                search for a new tower on frame n + 1, inside the confirmation area;
                if tower is detected on frame n + 1 then
                    save detected tower on frame n + 1;
                    tower on frame n is considered valid;
                end
            end
        end
        n = n + 1;
    end
% (Continues next page)
```
3. Tower Detection

% Backward search;
\( n = N; \)
\textbf{while} \( n > 1 \) \textbf{do}
\begin{itemize}
\item \textbf{if} \( \text{frame} \ n \) has a detected tower \textbf{then}
\begin{itemize}
\item \textbf{if} \( \text{frame} \ n - 1 \) has a detected tower \textbf{then}
\begin{itemize}
\item \textbf{if} tower on \( \text{frame} \ n - 1 \) is inside the confirmation area \textbf{then}
\begin{itemize}
\item tower on \( \text{frame} \ n \) is considered valid;
\end{itemize}
\item \textbf{else}
\begin{itemize}
\item search for a new tower inside the confirmation area;
\item \textbf{if} new tower is detected, with an higher score \textbf{then}
\begin{itemize}
\item replace tower on \( \text{frame} \ n - 1 \);
\item tower on \( \text{frame} \ n \) is considered valid;
\end{itemize}
\item \textbf{else}
\begin{itemize}
\item discard tower detected on \( \text{frame} \ n - 1 \);
\end{itemize}
\end{itemize}
\end{itemize}
\item \textbf{else}
\begin{itemize}
\item search for a new tower on \( \text{frame} \ n - 1 \), inside the confirmation area;
\item \textbf{if} tower is detected on \( \text{frame} \ n - 1 \) \textbf{then}
\begin{itemize}
\item save detected tower on \( \text{frame} \ n - 1 \);
\item tower on \( \text{frame} \ n \) is considered valid;
\end{itemize}
\end{itemize}
\end{itemize}
\item \( n = n - 1; \)
\end{itemize}
\end{itemize}
\textbf{end}
group consecutive frames with detected towers;

The result of the algorithm is a list of groups of consecutive frames containing a tower, separated by frames with no detected towers. Each group will most likely represent frames having the same tower. Next step is to exploit the temporal correlation between frames of the same group, in an attempt to improve the tower configuration.

3.2.2 Improving Tower Configuration

One of the objectives of knowing which frames contain the same tower is to improve the tower configuration, by relating the configurations detected in each frame. This will also improve the nest detection. Figure 3.31 shows an example of two frames, where the tower was well detected, but with a better configuration in one of them.
3.2 Tower Detection in Video Sequences

Figure 3.31: Configuration of the same tower found in two frames

Knowing the tower movement from Frame 1 to Frame 2, one could apply the same movement to the configuration of Frame 1 in order to project it on Frame 2. Block matching [10] is a very common and simple technique used to predict motion in a video sequence: for a given image block defined in Frame $n$, it searches for the most similar block in Frame $n + 1$ (or Frame $n - 1$); the displacement between the two blocks is interpreted as a motion vector. A similar principle is applied here. A block is defined around a point of interest, and its best correspondence is sought on the other frame, inside a searching window. To find the best correspondence, an equal sized block is slid through every position inside the window. In every position the correlation coefficient, $r$, between the block from Frame $n$, $A$, and block selected in the searching window, $B$, is computed using equation 3.1 [11]:

$$r = \frac{\sum_i \sum_j (A_{ij} - \overline{A})(B_{ij} - \overline{B})}{\sqrt{\left(\sum_i \sum_j (A_{ij} - \overline{A})^2\right) \left(\sum_i \sum_j (B_{ij} - \overline{B})^2\right)}}$$  \hspace{1cm} (3.1)

where $X_{ij}$ and $\overline{X}$ are, respectively, the luminance value of block $X$ in coordinates $(i, j)$ and the mean luminance value of block $X$. From the position with the higher correlation we will obtain the block displacement vector from Frame $n$ to Frame $n + 1$.

Given the high resolution of the images used, the blocks were defined with $32 \times 32$ pixels; the size of the searching window was arbitrated to $200 \times 200$ pixels. Figure 3.32 shows an example of three selected points in Frame 1 and their correspondence found in Frame 2. Figure 3.32 also shows, for each point, a unit vector with the same direction as the motion vector. Table 3.5 presents the coordinates of the selected points and the respective motion vector.
3. Tower Detection

(a) Frame 1: Selected points and block (blue), and a unit vector with the same direction as the resulting motion vector

(b) Frame 2: Searching window (yellow) and the corresponding points

Figure 3.32: Correspondence (b) of selected points (a) found

<table>
<thead>
<tr>
<th>Point</th>
<th>Coordinates in frame 1</th>
<th>Coordinates in frame 2</th>
<th>Motion vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point 1</td>
<td>(692, 368)</td>
<td>(676, 389)</td>
<td>(-16, 21)</td>
</tr>
<tr>
<td>Point 2</td>
<td>(641, 614)</td>
<td>(628, 596)</td>
<td>(-13, -18)</td>
</tr>
<tr>
<td>Point 3</td>
<td>(411, 569)</td>
<td>(426, 556)</td>
<td>(15, -13)</td>
</tr>
</tbody>
</table>

Table 3.5: Points from frame 1, their correspondence on frame 2 and the resulting motion vectors

The best points to get correspondence are points from a corner edge or intersections between edges. If for instance, a point from the sky is selected, with a very plain color around it, all the possibilities within the searching window would have a high correlation and the result will be unpredictable. Therefore, the best points to get the correspondence from, are the ending points of the lines from the tower configuration. Ideally, in order to find a line correspondence between frames, it would be sufficient to make the correspondence of the ending points of that line. However, it may happen that a selected line passes the contour of the tower, and the method would end up searching for a correspondence in the background and not on the tower, leading to unsatisfactory results. An example of this situation is shown in figure 3.33 (a) and (b). The line from the initial frame has one of the ends inside the tower, which leads to a correct correspondence; the other end falls on the background and the method will search for a correspondence of a background point. As the tower moves with a velocity different from that of the background, connecting the two points will result in a line that is misaligned with the tower’s contour.

To overcome this issue, more points from each line are used to get the whole line correspondence. Although these points are not points from a corner of the tower, the block formed around them will most likely include intersections of tower and background edges, which may lead to the correct correspondence between frames. The final motion vector, which will be applied to the line, will be the median of the motion vectors found for each point, as an attempt to discard outliers.
3.2 Tower Detection in Video Sequences

Figure 3.33: Comparison between using only 2 (first row) and 5 (second row) points

Looking at the example from figure 3.33, one can see that if only the two ends of the initial line are used to find the correspondent line, the new line would be wrong. On the other hand, if more points are used, the result is clearly improved.

The method developed to improve the configuration of a tower, using the group of frames containing the same tower, is summarized by the following steps:

1. find which of the frames from the group as the highest configuration score, save it has Frame n;

2. if Frame n has a configuration score lower than 60%, neglect the whole group of frames. This will result in discarding false tower detections or towers detected with a wrong configuration (a nest detection in that tower would have meaningless results);

3. compare each element of the tower configuration of Frame n with Frame n+1 independently, meaning left structure with left structure, right with right and horizontal with horizontal;

4. project from Frame n to Frame n + 1 the configuration element that has lower score on Frame n + 1 than on Frame n, as follows:
3. Tower Detection

- for the **vertical** structures:
  - select five equidistant points from each of the two lines (L1 and L2 or R1 and R2) of Frame n and compute their motion vector;
  - apply the median motion vector to both ends of each line, and connect them to compute the new pair of lines (L1’ and L2’ or R1’ and R2’) of Frame n + 1;
  - save the new lines temporarily.

- for the **horizontal** structures:
  - select left end and four more points, spaced by 16 pixels (half block size), from each of the horizontal lines (H1 and H2);
  - compute their motion vector and the resulting median vector, MV1, from Frame n to Frame n + 1;
  - select right end and four more points, spaced by 16 pixels, from each of the horizontal lines (H1 and H2);
  - compute their motion vector and the resulting median vector, MV2, from Frame n to Frame n + 1;
  - apply the motion vector MV1 to left end of lines H1 and H2, and the motion vector MV2 to right end of lines H1 and H2, connecting them to compute H1’ and H2’;
  - save the new lines temporarily.

5. compute the score of the new configuration, formed by the new lines found, using the method presented in section 3.1.4. Replace the old configuration only if the new configuration has a higher score than the old one;

6. repeat from step 2 with the next frame, until the end of the list (always comparing subsequent frames);

7. repeat the process, starting from Frame n and going backwards, until the first frame.

The method aims to improve the configuration of the tower found in each frame, comparing them to the one with the highest score. It ensures that, if by any chance the new configuration found in a frame is worst than the previous one, the latter one is saved and used as comparison for the subsequent frames, preventing errors propagation.

As an example, the method was applied to the left structure of the tower presented in figure 3.31. Five points from each of the lines L1 and L2 in Frame 1 were selected, and their correspondence was found on Frame 2, as figure 3.34 shows. The median of each of the five motion vectors was computed, and applied to each of the lines, resulting in L1’ and L2’. This will result in a better-shaped configuration, scoring 74% against the previous 59%. Figure 3.35 shows the final configuration of the towers on both frames.
3.2 Tower Detection in Video Sequences

Figure 3.34: Automatically selected points on frame 1 and their correspondence on frame 2

Figure 3.35: Left structure moved from the configuration in frame 1 to frame 2, same example as the figure 3.31

This method is applied to each group of frames obtained from the method described in subsection 3.2.1, resulting in the same groups but showing towers with an improved configuration. With this analysis, a more accurate search for nests may be performed.

The method is summarized in a block diagram in the next subsection; more illustrative results and comments about the whole tower detection system are presented in section 3.3.
3. Tower Detection

3.2.3 Diagram of the developed system

Figure 3.36 summarizes in a block diagram, the method developed to improve the configuration of detected towers. Next to each block is a sequence of squares representing the frames, where green squares represent a frame where a tower was detected. Arrows between squares represent the confirmation of a detection using the following frame. Green arrows represent a positive confirmation or a new tower found, whilst a red arrow represent a negative confirmation.

![Diagram of the system developed to detect tower in video sequences](image)

Figure 3.36: Diagram of the system developed to detect tower in video sequences system developed
3.3 Results and Comments

This section presents the results of the tests, performed to evaluate the tower detection method. It is divided in two subsections, where the first is related to the detection in Still Images, described in section 3.1 and the second refers to the detection using Video Sequences, presented in section 3.2.

3.3.1 Still Images Analysis

To evaluate the performance of the method developed to detect towers in still images, the group of still images mentioned in subsection 1.4.2 was used. The group contains 165 frames, 115 of which having a tower. The evaluation is based on two parameters, tower HR and tower FPR:

- **Tower HR (%)** = \( \frac{\# \text{ towers well detected}}{\# \text{ existing towers}} \times 100 \);
- **Tower FPR (%)** = \( \frac{\# \text{ detected towers without real correspondence}}{\# \text{ detected towers}} \times 100 \).

Detected towers were visually split in two groups, accordingly to the tower configuration:

- a tower has an acceptable configuration when all the image pixels from a tower are inside the lines that form the configuration;
- a tower has a wrong configuration when most of the image pixels inside the lines that form the configuration are from the background.

Table 3.6 presents the obtained results:

<table>
<thead>
<tr>
<th>Tower HR</th>
<th>86%</th>
<th>78% with acceptable configuration</th>
<th>22% with wrong configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tower FPR</td>
<td>9%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.6: Results obtained in the test sequence (visual results are presented in appendix B)

Results show that most of the towers were well detected, proving that the method is working as expected. Nevertheless, some of the towers were not detected, or were detected with a wrong configuration, which would lead to unpredictable results in the nest detection. Both the number of false negatives and the number of towers detected with a wrong configuration will be reduced when correlating frames in a video sequence, as will be shown in the next subsection. Moreover, the number of false positives will also be reduced when analyzing video sequences.

These results also show an improvement relatively to the method developed in previous work, and described on chapter 2, was only 59% of the towers were detected.
3. Tower Detection

3.3.2 Video Sequences Analysis

For the video sequences analysis, the tower detection performance was evaluated by the tower HR and by the percentage of towers detected with acceptable configuration, similarly to the previous tests, but now using the 20 short duration video sequences presented in 1.4.2. Each parameter shows two values:

- tower HR:
  - results obtained right after the *Still Images analysis*,
  - results obtained after exploiting the correlation between frames, in the *Video Sequences Analysis*;

- percentage of towers detected with an acceptable configuration:
  - before the motion analysis;
  - after the motion analysis.

With these parameters, one can verify the advantages of exploiting the temporal correlation between frames of video sequences, instead of analyzing them separately. It is also visible the improvement that the motion analysis brings to the tower configurations. Table 3.7 summarizes the results obtained.

<table>
<thead>
<tr>
<th>Seq. #</th>
<th>Nr. of Frames</th>
<th>Tower HR</th>
<th>% detect towers with accept. config.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>41</td>
<td>39%</td>
<td>73%</td>
</tr>
<tr>
<td>2</td>
<td>7 (4 with tower)</td>
<td>75%</td>
<td>100%</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>54%</td>
<td>63%</td>
</tr>
<tr>
<td>4</td>
<td>81 (70 with tower)</td>
<td>17%</td>
<td>80%</td>
</tr>
<tr>
<td>5</td>
<td>34</td>
<td>24%</td>
<td>76%</td>
</tr>
<tr>
<td>6</td>
<td>15 (14 with tower)</td>
<td>79%</td>
<td>100%</td>
</tr>
<tr>
<td>7</td>
<td>11</td>
<td>73%</td>
<td>91%</td>
</tr>
<tr>
<td>8</td>
<td>48</td>
<td>33%</td>
<td>61%</td>
</tr>
<tr>
<td>9*</td>
<td>41</td>
<td>10%</td>
<td>100%</td>
</tr>
<tr>
<td>10*</td>
<td>37</td>
<td>57%</td>
<td>100%</td>
</tr>
<tr>
<td>11</td>
<td>18</td>
<td>28%</td>
<td>94%</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>8%</td>
<td>100%</td>
</tr>
<tr>
<td>13</td>
<td>55</td>
<td>25%</td>
<td>78%</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
<td>36%</td>
<td>82%</td>
</tr>
<tr>
<td>15</td>
<td>64</td>
<td>19%</td>
<td>43%</td>
</tr>
<tr>
<td>16*</td>
<td>41</td>
<td>22%</td>
<td>100%</td>
</tr>
<tr>
<td>17</td>
<td>47</td>
<td>60%</td>
<td>98%</td>
</tr>
<tr>
<td>18</td>
<td>49</td>
<td>59%</td>
<td>100%</td>
</tr>
<tr>
<td>19*</td>
<td>71</td>
<td>11%</td>
<td>34%</td>
</tr>
<tr>
<td>20</td>
<td>7</td>
<td>86%</td>
<td>100%</td>
</tr>
<tr>
<td>Global</td>
<td>699 with tower</td>
<td>33%</td>
<td>78%</td>
</tr>
</tbody>
</table>

Table 3.7: Results obtained using the video sequences (*excerpt presented in appendix C*)
As foreseen, when analyzing video sequences the percentage of frames in which the tower is detected will increase over the still images analysis. In some of the sequences (e.g., seq. 9, 11 or 12), towers were detected in very few frames with the still images analysis, whilst after exploiting the correlation between subsequent frames, the tower is detected in almost every frame.

The tower configuration will also improve in many of the detections after the motion analysis. This will allow the next step of the work to have more information about a tower, resulting in a more accurate nest detection.

A second test was performed, using 10 short duration video sequences with no towers, in order to compare the number of false positives (frames with no tower where a tower was detected) when analyzing the video sequence as a group of still images or as a video sequence. Table 3.8 presents the results obtained.

<table>
<thead>
<tr>
<th>Seq. #</th>
<th>Number of false positives using:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Still images</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Video sequence</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Global</td>
<td>41</td>
<td>19</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.8: Results obtained using 10 video sequences without towers

From the results obtained, we can also conclude that the number of false positives will be reduced, as expected, if the correlation between frames is exploited, instead of just analyzing the video sequence as a group of still images.
3. Tower Detection
4 Nest Detection

Contents

4.1 Nest Detection in Still Images ........................................ 61
4.2 Nest Detection in Video Sequences ................................. 71
4.3 Results and Comments ............................................... 75
4. Nest Detection
4.1 Nest Detection in Still Images

Similarly to tower detection, nests detection is based on the search for their most evident features, which in this case are color, shape and position. Nests typically have a brownish color and an oval shape. Moreover, information about the tower is used to reduce nests possible locations.

This section comprehends the following subsections, presenting the study of each feature and its contribution to the nest detection algorithm:

1. **Nests color** - Learn nests general color gamut. In each image, pixels inside that color range are selected.

2. **Nests shape** - Find the shape characteristics of nests, and filter candidates with such shape.

3. **Tower region concealment** - Reduce the list of nest candidates, to those that are located on towers.

### 4.1.1 Nests color

In order to learn the nests color gamut, some seeds were manually selected on nests from the training sequence described in section 1.4.2, retrieving their color both on the Red-Green-Blue (RGB) and Hue-Saturation-Value (HSV) space. Then, a comparison was made against the color of the remaining pixels from the whole tower region. Two data matrices were created, one containing color values of each selected pixel from nests, \( P_{\text{nests}} \), and another containing color values of the remaining pixels, \( P_{\text{back}} \):

\[
P_{\text{nests}} = \begin{bmatrix} R_{P_1} & G_{P_1} & B_{P_1} & H_{P_1} & S_{P_1} & V_{P_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ R_{P_n} & G_{P_n} & B_{P_n} & H_{P_n} & S_{P_n} & V_{P_n} \end{bmatrix}, \quad P_{\text{back}} = \begin{bmatrix} R_{P_1} & G_{P_1} & B_{P_1} & H_{P_1} & S_{P_1} & V_{P_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ R_{P_n} & G_{P_n} & B_{P_n} & H_{P_n} & S_{P_n} & V_{P_n} \end{bmatrix},
\]

where each line represents a pixel and each column represents a color component value.

Plotting separately each column of each matrix, results on the histograms of figure 4.1 and 4.2. Comparing the color histograms of background pixels with those of nests pixels, it can be seen that they are centered at different values. This suggests that some colors tend to appear more often on a nest pixel than on the rest of the image. Next step is to find a way to assess the probability that a given pixel is a pixel from a nest or from the background.
4. Nest Detection

In the first attempt, two Multivariate Normal Distributions (MVND) [12] were used to describe the probability that a given pixel with color $\mathbf{x}$ is a background pixel or a nest pixel, where $\mathbf{x} = [R \ G \ B \ H \ S \ V]$. Equation 4.1 presents the Probability Density Function (PDF) used on each MVND, where $p(\mathbf{x}; \mu, \Sigma)$ is the probability that a given pixel is from a nest or from the background, depending on whether $\mu$ and $\Sigma$ are the mean and the covariance matrix of $P_{\text{nests}}$ or $P_{\text{back}}$, respectively.

$$p(\mathbf{x}; \mu, \Sigma) = \frac{1}{(2\pi)^{6/2} \ | \Sigma |^{1/2}} exp \left( -\frac{1}{2} (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) \right) \quad (4.1)$$

Both probabilities are computed for each pixel on an image, classifying it as background or as nest pixel according to the highest probability.

Using this classifier in the training sequence, approximately 80% of the nests pixels are classified as such, and approximately 20% of the background pixels are missclassified as being nests pixels. Although most of nests pixels are detected, the false positive rate is high enough to not allow the isolation of a nest region in an image, as seen in the three cases presented in figure 4.3. Pixels
4.1 Nest Detection in Still Images

classified as *nest* and *background* are shown in white and in black, respectively.

Figure 4.3: Pixels classified as *nests* in white, and as *background* in black

Given these results, a more selective search was performed, in order to reduce the false positive rate. Looking at the color histograms from figure 4.1 and 4.2, one can easily notice that separately, each background color histogram can be well described by a Gaussian distribution [13] and each nest color histogram by a Rayleigh distribution [13]. Equations 4.2 and 4.3, show the PDF of each distribution, where $\mu$ and $\sigma$ are the mean and standard deviation of the respective color column from $P_{\text{back}}$ or $P_{\text{nests}}$. Figure 4.4 shows the PDF of each color value for both the background, in blue, and nests colors, in red.

\[
\text{pdf}_{\text{back}}(x; \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{4.2}
\]

\[
\text{pdf}_{\text{nest}}(x; \mu, \sigma) = \frac{x}{\sigma^2} e^{\frac{-x^2}{2\sigma^2}} \tag{4.3}
\]

Figure 4.4: Probability density functions for background colors (in blue) and nest colors (in red)
4. Nest Detection

Figure 4.4 also presents a threshold, $T_x$, given by the intersection of nest and background color PDFs. In this new approach, a given pixel, with color $\vec{x} = [R \ G \ B \ H \ S \ V]$, is considered to be from a nest if it respects the following restrictions:

- $R, G, B, V$ below the respective threshold;
- $H$ above the respective threshold;

where $T_R = 105$, $T_G = 98$, $T_B = 95$, $T_H = 106$ and $T_V = 108$.

Saturation is ignored as values from the background and nest are very alike. Applying this classifier, a total of approximately 60% of the nests pixels are classified as such, and approximately 5% of the background pixels are missclassified as being a nest pixel. Even though the number of detected nests pixels has been reduced comparatively to the previous classifier, the false rate is much lower, allowing to better isolate a nest region.

Figure 4.5 compares the result of applying the previous and new classifier.

![Original Image](image1.png) ![Previous classifier](image2.png) ![New classifier](image3.png)

(a) Original Image  (b) Previous classifier  (c) New classifier

Figure 4.5: Pixels classified as nests in white

To compensate the reduction of the detected nests pixels, a morphological closing operator was applied, equal to the one used in the edge thinning from subsection 3.1.1. Then, connected pixels were joined together in groups, where a group of pixels is considered a nest candidate if the number of pixels on that group is:

- greater than the length, in pixels, of the H2 line from the tower detected in that image; and
- lower than 10 times the length, in pixels, of the same H2 line.

These values were arbitrated, as they are only intended to neglect large groups of pixels, most likely background, and meaningless small groups of pixels.

Figure 4.6 shows the result of applying these conditions, where each isolated group of connected white pixels represent a nest candidate. Next step is to analyze the shape of each candidate, filtering them according to the expected characteristics of a nest.
4.1 Nest Detection in Still Images

(a) Original Image  
(b) Map of detections

Figure 4.6: Map of detections (b) where isolated groups of connected white pixels represent nests candidates of the original image (a)

4.1.2 Nests shape

On the map of detections from figure 4.6, apart from the detected trees, nests can easily be differentiated from other candidates by their shape. While nests have an oval shape, parts selected from the tower have a shape closer to a line. Therefore, an ellipse is associated to each candidate, that has the same normalized second central moments as the group of connected pixels, and the ellipse’s major and minor axis are computed. This was performed using Matlab’s `regionprops` [7] function. Figure 4.7 is an example of a group of four pixels considered as a nest candidate, and the associated ellipse.

(a) Group of 4 pixels and the associated ellipse  
(b) An ellipse, with its major and minor axis

Figure 4.7: Example of a group with 4 pixels and the associated ellipse (a) and the ellipse’s major and minor axis (b)

Figure 4.8 shows, in red, the ellipses associated to each candidate on the map of detections of figure 4.6.
In order to decide whether or not a candidate is a nest, the method described so far was applied to the training sequence, associating an ellipse to every candidate. Then, the eccentricity of the ellipses associated to true nests was computed and plotted on figure 4.9. This allowed to conclude that the eccentricity of nests have values between $0.5$ and $0.9$.

Removing candidates with eccentricity lower than $0.5$ or higher than $0.9$ results, as expected, in discarding candidates that were in fact referring to parts from the tower. Figure 4.10 shows the final map of nests candidates, which will be reduced to the ones located on the tower region in the next subsection.
4.1 Nest Detection in Still Images

Figure 4.10: Map resulted from applying the eccentricity filter (a) and the final detection (b) with detected nests inside blue squares

4.1.3 Tower Region Concealment

Knowing the tower configuration, obtained on the previous chapter, the way to greatly reduce the number of false positives is to select the nests candidates which are actually on the tower. Nests are located on the top of the horizontal structures or along one of the vertical structures, so the search is reduced to this area. A map of possible locations is formed by 3 areas:

- A triangle with corners IL, IT1L and IT2L, forming the left structure;
- A trapezoid formed by the lines H1 and H2, representing the horizontal structure;
- A triangle with corners IR, IT1R and IT2R, forming the right structure.

The three areas are then enlarged using a morphological dilatation of \( \frac{\text{size}(H1)}{10} \) pixels, because part of the nest is usually located outside the actual tower. Figure 4.11 shows the configuration of the tower from the testing frame, and the areas to which the search will be reduced.

Figure 4.11: Tower configuration (a) searching area (b) obtained from the tower configuration
4. Nest Detection

There are also some cases, such as the one presented in figure 4.12 (a), in which nests are located on the top of the vertical structures. To take those cases into account, an inverted triangle is placed on the top of each vertical structure, with one third of its size. This is exemplified on figure 4.12 (b).

![Figure 4.12: Tower configuration (a) and searching area (b)](image)

Lastly, candidates from subsection 4.1.2 are assumed as a nest on the horizontal structure if the centroid of the candidate is inside the region marked as red. If not on the horizontal structure, they will be assumed as a nest on the left or right structure if the candidate’s centroid is inside the blue or green region. If the detection is outside any of these areas, it is considered a false positive and is discarded. This ends up on the result presented on figure 4.13, where only nests were selected.

![Figure 4.13: Image with two detected nests](image)
Figure 4.14, shows a false positive and an undetected nest, that happened due to the presence of trees on the background. The false positive occurs because one of the trees, with color and shape very similar to a nest, was in line with the tower, respecting all restrictions to be considered as a nest. On the other hand, the nest on the right structure was mixed with all the trees on the background and the method was unable to isolate it.

![Figure 4.14: Example with a detected nest, a false positive and an undetected nest](image)

Some of these cases may be prevented when relating detections on the same tower from different frames. This will be addressed afterwards.

Next subsection presents, in a diagram, the main steps of the developed method.
4. Nest Detection

4.1.4 Diagram of the developed system

Figure 4.15 summarizes the method developed to detect nests in still images that contain a tower, using the tower configuration found previously.

![Diagram of the system developed for nest detection in still images](image)

Figure 4.15: Diagram of the system developed for nest detection in still images
4.2 Nest Detection in Video Sequences

The method developed on chapter 3 detects groups of frames containing the same tower. The final step is to achieve the main goal of this dissertation, which is to identify which of those towers are in risk of malfunction due to the presence of nests. Analyzing each frame of the group, using the method presented on the previous section, already gives a good estimation about the presence of nests on each frame of the group. Nevertheless, a further analysis exploiting the correlation between frames may lead to a more accurate result, mainly by reducing the number of false nest detections. This subsection suggests a method able to perform this analysis.

Each group of frames shows the tower with increasing size, as the helicopter (and the camera) moves towards the tower, so in the first frames the tower does not occupy a significant part of the image. To avoid misdetections, nests will only be searched for on frames in which the length of the tower’s horizontal structure is, at least, one third of the image horizontal length (in pixels). That search will result in a list of candidates in each frame, where each candidate is classified according to its position on the tower: left, horizontal or right structure. This list is then split in three sub-lists:

- **Left candidates** - candidates detected on a left structures with score higher than 50%.
- **Horizontal candidates** - candidates detected on an horizontal structures with score higher than 50%.
- **Right candidates** - candidates detected on a right structures with score higher than 50%.

Nest detections on towers with an erroneous shape configuration will be meaningless, as they would most likely be located outside the tower region; this justifies why the search is limited to structures with a score higher than 50%.

Each list will be analyzed separately, using the method described on the next subsection, to confirm the presence of a nest on the tower.

4.2.1 Relating Nests Detections

In order to associate nests candidates in different frames that correspond to the same nest, the position of each candidate on the tower referential is computed, as exemplified on figure 4.16. The candidate’s position is given by the distance to a reference point, the tower referential origin, normalized by the tower structure size, according to (see figure 4.16):

- For a candidate on the left structure:
  
  - point IT1L is the origin; the distance is normalized to the length of the line L1.
4. Nest Detection

- For a candidate on the horizontal structure:
  - left end of the line H2 is the origin; the distance is normalized to the length of the line H2.

- For a candidate on the right structure:
  - point IT1R is the origin; the distance is normalized to the length of the line R1.

![Diagram of tower referential with axes and distances labeled.

Figure 4.16: Representation of the tower referential, composed by three axis: horizontal detection’s referential (in red), vertical left detection’s referential (in blue) and vertical right detection’s referential (in green)

Although the tower changes its position from frame to frame, nests are always at the same position within the tower referential. This allows to learn which candidates in different frames are referring to the same nest. In order to relate nest candidates, the following algorithm is applied to each sub-list:

1. start at the first frame, with the first candidate;

2. search on each one of the following frames for candidates that are located on the same position:
   - candidates on the horizontal structure are assumed to be on the same position if the distance between them is less or equal to 0.1;
   - candidates on the vertical structures are assumed to be on the same position if the distance between them is less or equal to 0.2;

3. associate to the same nest those candidates located on the same position; score that nest with the number of associated candidates; if more than one candidate correspondence is found on the same frame, only the closest one will be assumed as correspondent;
4.2 Nest Detection in Video Sequences

4. remove associated candidates from the sub-list, and repeat from Step 1 with the next candidate and with the following frames, until the initial sub-list is empty;

5. nests with a score higher than one third of the total number of frames on the list are considered to be true nests.

Figure 4.17 shows an example of the algorithm application. Candidates, in different frames, that were distanced by 0.1 or less from each other, are associated to the same nest. Then, nests that had more than two associated candidates were assumed to be true nests. In this example, two final nests were detected.

<table>
<thead>
<tr>
<th>Frame 1</th>
<th>Frame 2</th>
<th>Frame 3</th>
<th>Frame 4</th>
<th>Frame 5</th>
<th>Frame 6</th>
<th>Score</th>
<th>Final nest?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.15</td>
<td>0.4</td>
<td>0.2</td>
<td>0.1</td>
<td>0.5</td>
<td>4</td>
<td>yes</td>
</tr>
<tr>
<td>0.7</td>
<td>0.7</td>
<td>0.8</td>
<td></td>
<td></td>
<td>0.99</td>
<td>2</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>no</td>
</tr>
</tbody>
</table>

Figure 4.17: Example of a list of detections on an horizontal structure

This method will allow to neglect false positives, as shown in figure 4.17: the second candidate on Frame 5, or the first candidate on Frame 3 and Frame 6 will be discarded; as they appeared on a few number of frames, they are most likely to be a false positive, e.g., a tree misinterpreted as a nest.

The method will also work for cases where the nest is very well detected on some frames but not so well on others, as exemplified on the third line of figure 4.17. These cases may happen, for instance, when the nest has a good contrast with the background but at some point of the video the tower crosses a group of trees, where the nest gets mixed up with the background.

As the presented method only relies on the temporal correlation between candidates, it will only serve for the purpose of reducing the number of false detections. Furthermore, the method also has some limitations: there are cases where the tower never occupies a significant part of the image, or cases where the tower is never detected with a good configuration. On those cases, the search for nests is not performed at all. On the other hand, if the second line of detections on figure 4.17 were true nests, not well detected on other frames, they would be discarded producing a false negative.

Summing up the number of nests on each part of the tower would give the total number of nests on that tower, producing the final report. This report is the final goal of this dissertation, suggesting or not the need of human intervention, to remove nests from that tower.
4. Nest Detection

4.2.2 Diagram of the developed system

Figure 4.18 summarizes, through a block diagram, the method developed to detect nests in a sequence of frames.

![Diagram](image)

Figure 4.18: Diagram of the developed algorithm for nest detection in a sequence of frames
4.3 Results and Comments

This section presents the results of the tests performed to evaluate the performance of the nest detection algorithm.

4.3.1 Still images analysis

A test sequence of 165 independent still images containing 97 nests was used to search for nests, where the correct tower configuration was hand-selected. The nest detection performance was assessed through the nests HR and nests FPR, defined as:

- **Nest HR (%)** = \( \frac{\# \text{ well detected nests}}{\# \text{ existing nests}} \times 100 \);
- **Nests FPR (%)** = \( \frac{\# \text{ detected nests without true correspondence}}{\# \text{ detected nests}} \times 100 \).

Results are shown on table 4.1.

<table>
<thead>
<tr>
<th>Nest HR</th>
<th>53%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nest FPR</td>
<td>29%</td>
</tr>
</tbody>
</table>

Table 4.1: Results obtained testing the sequence (presented in appendix D)

As mentioned, the system fails when the nest gets mixed with the background, which happens very often. On the other hand, trees represent a major problem as their color and shape resembles a possible nest.

Even though the results are not satisfactory, both the nest detection rate and number of false positives will be improved when analyzing frames as video sequence, as will be seen on the next subsection.

4.3.2 Video sequences analysis

Twenty short duration test video sequences were analyzed, containing 20 towers, 13 of which having nests, between 1 and 4 per tower, summing up a total of 23 nests. Table 4.2 shows the results for each sequence, analyzed as a set of independent frames, or as a true video sequence, where nest detections were cross-referenced between frames. In the still images analysis, performance was evaluated using the same parameters as in the previous subsection. On the video sequences analysis, results are presented in [L H R] detections, which refers to detections on the left, horizontal and right structure.
4. Nest Detection

Video sequences analysis may be used to find towers that are in risk of malfunction, if there are nests on that tower. From this perspective, the method was evaluated through the following parameters:

- **Tower "in risk" HR (%)** = \( \frac{\# \text{towers well detected as in risk}}{\# \text{towers in risk}} \times 100 \), where a tower in risk may have been detected with:
  - the exact number of existing nests; or
  - more or less than the exact number of nests.

- **Tower "in risk" FPR (%)** = \( \frac{\# \text{towers wrongly detected as in risk}}{\# \text{towers detected as in risk}} \times 100 \)

The hit-rate shows the percentage of towers for which an alert should be generated. The number of false positives is the number of generated alerts that did not correspond to a tower in risk. This analysis was performed and results are shown on table 4.3.

\[
\begin{array}{|c|c|c|c|c|c|c|}
\hline
\text{Seq. #} & \text{Frames} & \text{Nests/frame} & \text{Nest HR} & \text{Nest FPR} & \text{Existing nests} & \text{Detected nests} \\
\hline
1 & 41 & 1 & 5\% & 60\% & [1 0 0] & [0 0 0] \\
2 & 7 & 0 & - & 7 & [0 0 0] & [0 0 0] \\
3 & 15 & 0 & - & 29 & [0 0 0] & [0 0 0] \\
4 & 34 & 0 & - & 94 & [0 0 0] & [0 1 0] \\
5 & 34 & 0 & - & 19 & [0 0 0] & [0 0 0] \\
6 & 15 & 0 & - & 11 & [0 0 0] & [0 0 0] \\
7* & 11 & 2 & 28\% & 0\% & [1 0 1] & [1 0 1] \\
8 & 41 & 0 & - & 118 & [0 0 0] & [0 0 0] \\
9* & 41 & 2 & 68\% & 57\% & [1 0 1] & [1 0 1] \\
11 & 18 & 2 & 72\% & 37\% & [1 0 1] & [1 0 1] \\
12 & 12 & 2 & 88\% & 16\% & [1 0 1] & [1 1 1] \\
13 & 55 & 1 & 18\% & 57\% & [1 0 0] & [0 0 0] \\
14* & 11 & 3 & 76\% & 14\% & [1 0 2] & [1 1 0] \\
15 & 63 & 2 & 19\% & 40\% & [1 0 1] & [1 0 1] \\
16 & 41 & 0 & - & 5 & [0 0 0] & [0 0 0] \\
17* & 47 & 1 & 17\% & 60\% & [1 0 0] & [1 0 0] \\
18 & 51 & 1 & 6\% & 80\% & [0 0 1] & [0 0 0] \\
19 & 71 & 1 & 24\% & 0\% & [0 0 1] & [0 0 1] \\
20 & 7 & 1 & 29\% & 60\% & [1 0 1] & [1 0 1] \\
\hline
\text{Overall} & 699 & 743 & 40\% & 37\% & \multicolumn{2}{c|}{\text{Nest HR = 78\% \text{ Nest FPR = 22\%}}} \\
\hline
\end{array}
\]

Table 4.2: Results obtained on the video sequences analysis (*excerpt presented in appendix E)

Table 4.3: Results obtained on the test video sequences
The tower configuration was automatically computed by the developed tower detection method, which means that some errors were also due to an incorrect tower configuration. More tests were performed using the available video sequences in their full length; they will be presented and commented on the next (and final) chapter.
4. Nest Detection
5

Final Experiments and Conclusions

Contents

5.1 Final Experiments ........................................ 81
5.2 Main conclusions ........................................... 83
5.3 Future work .................................................. 83
5. Final Experiments and Conclusions
This chapter presents the tower in risk detection results obtained after applying the developed algorithm to the video sequences listed in section 1.4, at their full length. Main conclusions are drawn, and some suggestions for further work are put forward.

5.1 Final Experiments

The video files listed in section 1.4 were tested at their full length. The objective was to present a report for each video sequence, with the number of towers and their position in time, marking a tower as "in risk" when it has, at least, one nest. Nests positions were classified as being in the horizontal, vertical left or vertical right tower structure, and the detection is assumed as valid only if detected in the correct tower structure. This result is intended to be used in real life. If the video sequences are georeferenced, it will be possible to locate the detected tower in a map, learning which of them require intervention related to the existence of storks nests.

In the first part of the test, the tower detection method was evaluated through: the tower detection HR and the tower detection FPR. These parameters give an idea of the reliability of the developed method, relatively to the percentage of towers that were actually detected and the number of times a false detection occurs. As previously, these parameters are defined as:

- **Tower HR (%)** = \( \frac{\text{# well detected towers}}{\text{# existing towers}} \times 100 \);
- **Tower FPR (%)** = \( \frac{\text{# wrongly detected towers}}{\text{# detected towers}} \times 100 \).

Results are presented on table 5.1:

<table>
<thead>
<tr>
<th>Video file</th>
<th>Existing towers</th>
<th>Tower HR</th>
<th>Tower FPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC_0138.mov</td>
<td>64</td>
<td>81%</td>
<td>10%</td>
</tr>
<tr>
<td>PIC_0140.mov</td>
<td>5</td>
<td>40%</td>
<td>33%</td>
</tr>
<tr>
<td>00006.mts</td>
<td>42</td>
<td>88%</td>
<td>14%</td>
</tr>
<tr>
<td>00007.mts</td>
<td>48</td>
<td>90%</td>
<td>16%</td>
</tr>
<tr>
<td>00014.mts</td>
<td>6</td>
<td>50%</td>
<td>0%</td>
</tr>
<tr>
<td><strong>Global</strong></td>
<td><strong>165</strong></td>
<td><strong>83%</strong></td>
<td><strong>14%</strong></td>
</tr>
</tbody>
</table>

Table 5.1: Tower detection results

The method commonly fails to detect towers that do not have enough contrast with the background. Cases like this happened in most of the towers from the second and the fifth video file. Although the number of towers is not so much representative on those video files, the method fails to detect half of the towers. For the remaining video files, most of the towers were correctly detected, with a low percentage of FP’s. The number of FP may be decreased if the detected towers positions are compared with the true positions, if known.
5. Final Experiments and Conclusions

The second part of the tests evaluated the performance of the nest detection algorithm. The objective was to identify which of the towers are "in risk" due to the presence of nests, suggesting the need of human intervention. The performance evaluation is based on tower "in risk" HR - the percentage of towers with nests which were well detected - and the percentage of towers where nests were wrongly detected - tower "in risk" FPR.

- **Tower "in risk" HR (%)** = \( \frac{\# \text{towers well detected as in risk}}{\# \text{towers in risk}} \times 100 \);
  - a *tower well detected as in risk* is a tower with, at least, one well detected nest;
  - a *tower in risk* is a tower that have, at least, one nest;

- **Tower "in risk" FPR (%)** = \( \frac{\# \text{towers wrongly detected as in risk}}{\# \text{towers detected as in risk}} \times 100 \).
  - a *tower wrongly detected as in risk* is a tower with no nests, but where nests were detected;
  - a *tower detected as in risk* is a tower with at least one detected nest.

Results are presented in table 5.2.

<table>
<thead>
<tr>
<th>Video file</th>
<th>Towers with nests</th>
<th>Tower &quot;in risk&quot; HR</th>
<th>Tower &quot;in risk&quot; FPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC_0138.mov</td>
<td>14</td>
<td>67%</td>
<td>68%</td>
</tr>
<tr>
<td>PIC_0140.mov</td>
<td>0</td>
<td>-</td>
<td>0%</td>
</tr>
<tr>
<td>00006.mts</td>
<td>21</td>
<td>83%</td>
<td>33%</td>
</tr>
<tr>
<td>00007.mts</td>
<td>4</td>
<td>100%</td>
<td>18%</td>
</tr>
<tr>
<td>00014.mts</td>
<td>0</td>
<td>-</td>
<td>0%</td>
</tr>
<tr>
<td>Global</td>
<td>39</td>
<td>79%</td>
<td>43%</td>
</tr>
</tbody>
</table>

Table 5.2: Nest detection results

The method shows a good performance on the last four video files, correctly marking most of the towers that contain nests. The nest detection relies on having the correct tower configuration detected. This means that, even if a tower’s position in a frame is correctly detected, if its configuration is not correct, a nest detection may be meaningless. This leads to undetected nests, or detections that do not have true correspondence.

In many of the cases, more frequent on the first video, towers were not detected in every frame where it appears. This also lead to false detections, since the nest false positive reduction is based on the comparison between different frames containing the same tower.

These results, although encouraging, allow to conclude that the tower detection and, not particularly, the nest detection, needs to be enhanced before the automatic nest inspection could be considered as a valid substitution of the human inspection. Nevertheless, and comparatively to the results from the previous work [2], it was shown that the use of the temporal correlation between frames is an important step towards an accurate tower and nest detection.
5.2 Main conclusions

In this dissertation we developed a method for the automatic detection of very high voltage towers and the presence on it of storks nests, from video sequences. The method is intended to be used in the electrical power transmission maintenance, from which people’s daily life depend on. This means that a fully automatically approach needs to be extremely reliable to put aside human confirmation. Accordingly, results obtained by the work developed in this dissertation are not yet satisfactory. However, the developed algorithm may be considered a good approach that, with some improvements, may lead to a method accurate enough to dispense human intervention.

Another main objective of this dissertation was to prove that a method based on the analysis of a video sequence, exploiting the temporal correlation between frames, led to more accurate results relatively to a method that takes only still images in consideration. This was proved, either by the increased number of tower and nest detection, either by the decreased number of detections without true correspondence (false positives), when videos are used.

5.3 Future work

A major improvement could be brought to the results if videos were recorded by a camera fixed on the helicopter. Videos used on the development of this dissertation were recorded by an hand-operated camera. Operators often followed the tower, moving the camera or zooming it in or out, making the movement on the image quite unpredictable. Moreover, when moving the camera, a tower can go from one end of a frame to the opposite end in the following frame, which would not happen if the camera was fixed on the helicopter. Although the movement of the helicopter cannot be exactly predicted, it can be predicted with enough approximation, as it moves in a very constant speed and roughly at the same distance from the towers. This prediction could be extended to the video, and hence to a detected tower, predicting its movement from frame to frame as soon as it is detected.

Regarding the nest detection, the algorithm relies completely on the correct detection of the tower, accordingly, improving the tower detection will consequently improve the nest detection. The method itself may also be improved if the search for nests in a tower was not based on its color, as this is a feature that varies from nest to nest. A texture or a shape analysis, together with the color, could lead to better results.
Bibliography


A

Tower Model
A. Tower Model
Figure A.1: Tower model used and their most important measures

Measures presented here are normalized to the length of line H1. These values were estimated by measuring the proportions of the given model and are not exact values.
A. Tower Model
Tower Detection In Still Images - Results
B. Tower Detection In Still Images - Results
Figure B.1: Results achieved by analyzing the training sequence of still images (part 1)
B. Tower Detection In Still Images - Results

Figure B.2: Results achieved by analyzing the training sequence of still images (part 2)
Figure B.3: Results achieved by analyzing the training sequence of still images (part 3)
B. Tower Detection In Still Images - Results

Figure B.4: Results achieved by analyzing the training sequence of still images (part 4)
Figure B.5: Results achieved by analyzing the training sequence of still images (part 5)
Figure B.6: Results achieved by analyzing the training sequence of still images (part 6)
Tower Detection In Video Sequences
- Results
C. Tower Detection In Video Sequences - Results
Figure C.1: Sequence 19 analyzed as a group of still images (col. 1), as video sequence before (col. 2) and after (col. 3) the motion prediction
Figure C.2: Sequence 16 analyzed as a group of still images (col. 1), as video sequence before (col. 2) and after (col. 3) the motion prediction
Figure C.3: Sequence 9 analyzed as a group of still images (col. 1), as video sequence before (col. 2) and after (col. 3) the motion prediction
Figure C.4: Sequence 10 analyzed as a group of still images (col. 1), as video sequence before (col. 2) and after (col. 3) the motion prediction
Nest Detection In Still Images - Results
Figure D.1: Results achieved by analyzing the training sequence of still images (part 1)
Figure D.2: Results achieved by analyzing the training sequence of still images (part 2)
Nest Detection In Video Sequences - Results
E. Nest Detection In Video Sequences - Results
Figure E.1: Excerpt from sequence 7, both nests detected
Figure E.2: Excerpt from sequence 9, both nests detected
Figure E.3: Excerpt from sequence 14, three detected nests and a false positive
Figure E.4: Excerpt from sequence 17 with detected nest