Automatic Detection of Stork Nests on Very-High Voltage Towers

João Pedro Ribas Fernandes
Instituto Superior Técnico
Technical University of Lisbon
Lisbon, Portugal

Abstract—This paper proposes a method for the automatic detection of stork’s nests on the top of very-high voltage towers, using common image processing tools 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. 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.

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. This 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.

I. INTRODUCTION

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 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 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].

In this paper we propose a method, based on common image processing tools, able of automatically identify the presence of nests on the top of 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. Such report is intended to alert REN to towers that are in risk of malfunction, suggesting the removal of nests.

The tower model used on the development of the method is shown on figure 1, and the search for nests is reduced to the most critical part of the tower, represented by the regions surrounded by the blue and red lines.

Fig. 1. Tower model used and the area of interest to search for nests in red (horizontal structure) and in blue (vertical structures)

Five videos from a regular inspection were used, with a total duration of nearly one hour, and containing a total of 165 towers and 39 nests. From those 5 videos, 20 short duration sequences and a group of 165 still images were extracted, to train and evaluate the performance of the method. The performance evaluation of the method is based on the Hit-Rate (HR) – percentage of towers in risk (i.e., with nests) well detected – and on the False Positive Rate (FPR) – percentage of towers with no nests that were identified as in risk.

The algorithm was based on a previous work [2], that used only still images analysis, applied to frames extracted from videos. The main contribution of the presented work was to enhance the tower detection algorithm and to reformulate the nest detection algorithm. Also, the temporal correlation between subsequent frames of a video sequence was exploited to improve the results.

This paper is organized as follows: Section I introduces and formulates the problem that motivated the work. Section II and III describe the method developed to detect towers and nests, respectively. Those methods are firstly applied to still images and then to video sequences. At the end of each section, results from both approaches are presented. Section IV presents the results obtained by applying the final developed method to the 5 videos at their full length. On section V, main conclusions are presented, as well as some suggestions for future work.
II. TOWER DETECTION

Object detection in images is a problem deeply discussed and addressed due to its potential use in a wide variety of applications. Typically, objects detection is based on the search for features that best describe them. On the subject in study, towers have a very distinguishing characteristic from the rest of the scenario: their well-defined structure, formed by straight lines. That said, the algorithm developed to detect towers in still images will search for the most contrasting straight lines on an image to select the most probable tower regions. Then, a new and more specific search for lines is performed on each region, trying to find the set of lines that would best define the contour of a tower present on that region, selecting which of the regions, if any, is more likely to contain a tower. After tower detection on video sequences, analyzed as a group of independent still images, the temporal correlation between frames is exploited, in an attempt to find frames with undetected towers or detections without true correspondence.

A. Tower detection in still images

The method developed to detect towers in still images is composed by the following steps:

1) Edge detection.
2) Line detection.
3) Detection of probable tower regions in each image.
4) Find the configuration of a possible tower contained on the most probable region.

Edges in images are detected using the Canny Edge Detector [3], which uses a Gaussian filter to smooth the noise on the image, then computes the gradient in each image pixel. The gradient norm is compared to a given pair of thresholds, \( T_1 < T_2 \). Pixels with gradient norm above \( T_2 \) are considered as "strong edges"; pixels with gradient norm between \( T_1 \) and \( T_2 \) are considered as "edges" if they are 8-connected to a "strong edge"; otherwise, they are considered as "non-edges". The Canny Edge Detector was used with \( T_2 = T \) and \( T_1 = 0.4 \times T \), initially using \( T = 0.4 \), and adapting \( T \) to the image after applying an edge thinning operator [4].

The edge thinning was performed using a morphological "closing" operator [5], joining edge pixels distanced by 5 pixels or less. After that, connected pixels are grouped, and groups that have less pixels than 15% of the image horizontal length will be discarded. Since it is expected that the tower represents a significant part of the image, discarding small groups of pixels will allow to discard edges that do not belong to a tower.

After applying the edge thinning, the percentage of image pixels that were set as "edges" is computed and, if that percentage is lower than 20%, the Canny Edge Detector is reapplied, reducing the threshold by 0.01. The process is repeated until that percentage is above 20%. Such threshold adaptation has the purpose of guaranteeing that a significant number of edges is found. Figure 2-b) shows an example of an edge map resulting from the algorithm.

Next step is to find straight lines on the edge map, which would represent straight edges, using Hough Transform [6]: it gives every possible line on an image a score proportional to the number of points of interest (in this case edge pixels) that each line superimposes. It was applied, using 15% of image horizontal length as the minimum line length, in pixels, and selecting a maximum of 20 high scored lines from the transform. Figure 2-c) shows an example of a set of detected lines on the edge map from figure 2-b).

Detected lines are divided in two groups, whether they are vertical (with an angle between \( \frac{\pi}{4} \) and \( \frac{3\pi}{4} \) rad with the horizontal direction), or horizontal (otherwise). Every horizontal line that is intercepted by, at least, one vertical line is considered to be a candidate to an horizontal edge of a tower. A probable tower region is selected on the image, according to the size of that horizontal line, \( L_H \), as shown in figure 3.

Fig. 2. Original image (a), edge map resulting from the Canny Edge Detector (b) and lines (in blue), detected by the Hough Transform (c)

This selection is an attempt to keep only the interesting region of a tower for further analysis. Figure 4 shows an example of the selected regions in an image. After the edge and the line detection, five horizontal lines, intercepted by vertical lines, were detected, therefore five regions were selected.

Fig. 3. Example of a selected probable tower region

(a) Original image  
(b) Edge map  
(c) Detected lines

(a) Original image  
(b) Edge map and lines

Fig. 4. Original image with selected probable tower regions (a) and the edge map with detected lines and selected regions (b)
To identify which of the regions, if any, contains a tower, the regions are re-analyzed, searching for more specific edges and straight lines, using the same method presented before but with some modifications applied to the edge and line detecting methods: the Canny Edge Detector is applied, adapting the threshold to the image until the percentage of pixels selected as edges are between 7% and 10% of the region size. In the edge thinning, groups that have less pixels than 80% of the image horizontal length (in pixels) are discarded. The Hough Transform is now applied three times: a first time searching for image horizontal length (in pixels) are discarded. The Hough edge thinning, groups that have less pixels than 80%, as edges are between horizontal lines (with an angle, with the horizontal direction, 

\[ \pi \] and \[ \frac{\pi}{2} \] rad; a second time searching for vertical left lines (with an angle, with the horizontal direction, between \( \frac{\pi}{2} \) and \( \frac{3\pi}{2} \) rad); and a third time, searching for vertical right lines (with an angle, with the horizontal direction, between \( \frac{\pi}{2} \) and \( \frac{\pi}{2} \) rad). This will allow to strictly search for horizontal and vertical left and right lines, specifically from each part of the tower structures contour. Figure 5 shows the lines detected on this second search, applying it to one of the regions selected in figure 4, where it is seen that more specific lines from the tower contour were detected.

(a) Selected region
(b) Edge map and lines

Fig. 5. One of the selected regions from figure 4 (a) and detected lines (b): horizontal lines in red, vertical left in blue and vertical right in green

Next step is to score each selected region with a value that will depend on how similar the detected lines are to the ones presented in figure 6.

The algorithm chooses one horizontal line at a time (assuming it as an H1 line) and scores every other line (assuming they are H2, L1 or L2 and R1 or R2) according to their slope, size and position on the image, relatively to the line H1. Each score given to a line feature is normalized, using the normalization function presented in figure 7 and values presented in table I. Algorithm 1 shows the pseudo code that represents the algorithm developed to score each region, where \( A, B, A_{center}, A_{leftpoint}, A_{rightpoint} \) are the distance, in pixels, between point \( A \) and point \( B \), the center, the leftmost and the rightmost points of line \( A \), respectively.

\( \begin{align*}
\text{Score}(L) &= \text{mean}[\text{Score}(L_k)]; \quad k = 1; \\
\text{Score}(R) &= \text{mean}[\text{Score}(R_k)]; \\
\text{Score}(H1) &= \text{mean}[\text{Score}(H2_j)]; \quad i + +; \\
\text{Score}(H2) &= \text{mean}[\text{Score}(H2_j)]; \quad i + +;
\end{align*} \)

\( \text{Algorithm 1: Score a given image region} \)

After scoring each selected region from an image, if the highest score is above 65%, it is assumed that the image contains a tower on that region. If that is the case, the exact tower configuration is sought, trying to find the set of six lines that are more similar to the principal contours of the tower, exemplified on figure 6, among the lines found on that region.
A similar method is applied, scoring every combination of 6 lines, using the normalization values listed in a table presented in [7], and scoring functions presented next. The objective now is to find the exact set of lines that best define the tower structure and their score.

The set of lines chosen as the tower configuration will be the combination of lines that maximizes the following equations:

\[
\text{Score}(H) = \frac{\text{Size}(H) + \text{Position}(H) + \text{Slope}(H)}{3} \\
\text{Score}(L) = \frac{\text{Score}(L1) + \text{Score}(L2) + \text{Position}(L) + \text{Center}(L)}{2} \\
\text{Score}(R) = \frac{\text{Score}(R1) + \text{Score}(R2) + \text{Position}(R)}{2}
\]

where:

- \(\text{Position}(H) = \frac{\text{H1}_{\text{center}} + \text{H2}_{\text{center}}}{2}\)
- \(\text{Center}(L) = \frac{\text{L1}_{\text{center}} + \text{L2}_{\text{center}}}{2}\)
- \(\text{Center}(L) = \frac{\text{R1}_{\text{center}} + \text{R2}_{\text{center}}}{2}\)
- \(\text{Position}(L) = \frac{\text{Position}(L1) + \text{Position}(L2)}{2}\)
- \(\text{Position}(R) = \frac{\text{Position}(R1) + \text{Position}(R2)}{2}\)

Lines from the detected configuration are saved for later search for nests on those lines.

Figure 8 shows an example of three towers detected by applying the method to three different frames. On the first and second cases, towers are well detected although without the correct configuration on the first case. The third case is an example of a false positive, where a tower was detected but without true correspondence.

### TABLE I

VALUES USED IN THE NORMALIZATION FUNCTION, WHERE MAX VALUES ARE THE EXPECTED VALUES OF EACH FEATURE, GIVEN THE DIMENSIONS OF THE TOWER MODEL AND TAKING H1 AS REFERENCE

<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>(14 \times \text{Size}(H1))</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Position(H2)</td>
<td>(\text{Size}(H1))</td>
<td>0</td>
<td>(\text{Size}(H1))</td>
</tr>
<tr>
<td>Slope(H2)</td>
<td>(\text{Slope}(H1))</td>
<td>0</td>
<td>(\text{Slope}(H1))</td>
</tr>
<tr>
<td>Position(ITL)</td>
<td>(\text{Size}(H1))</td>
<td>(\frac{\text{Size}(H1)}{2})</td>
<td>(\frac{\text{Size}(H1)}{2})</td>
</tr>
<tr>
<td>Position(IBL)</td>
<td>(\text{Size}(H1))</td>
<td>(\frac{\text{Size}(H1)}{2})</td>
<td>(\frac{\text{Size}(H1)}{2})</td>
</tr>
<tr>
<td>Position(ITR)</td>
<td>(\text{Size}(H1))</td>
<td>(\frac{\text{Size}(H1)}{2})</td>
<td>(\frac{\text{Size}(H1)}{2})</td>
</tr>
<tr>
<td>Position(IBR)</td>
<td>(\text{Size}(H1))</td>
<td>(\frac{\text{Size}(H1)}{2})</td>
<td>(\frac{\text{Size}(H1)}{2})</td>
</tr>
</tbody>
</table>

B. Tower detection in video sequences

Extending the method to the case of video sequences, we expect to improve the results by increasing the number of detected towers and 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:

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

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 a second search on the following frames. As it will be shown, this procedure allows either to detect a misdetected tower in a new frame, either to lower the confidence of previously detected tower.

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 previously.
2) Neighborhood confirmation.
3) Improving tower configuration.

If a tower was detected on frame \(n\), it will be confirmed using \(n \pm 1\). The tower region from frame \(n\) will be used on neighbor frames as the confirmation area, centered on the same image coordinates but 10% longer, in both directions, to cope with the "tower movement" (e.g. in figure 9).

![Frame n and Frame n ± 1 with confirmation area in dashed yellow lines](image)
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 developed to evaluate detected probable tower regions, presented previously, using the confirmation area, leads to:
   a) **No tower detected** (Frame \( n \) has a FP?).
   b) **A tower is detected** (new tower detected!).

2) Frame \( n \pm 1 \) already have a detected tower, that is:
   a) **Outside the confirmation area** (Frame \( n \) or Frame \( n \pm 1 \) must be a FP).
   b) **Inside the confirmation area** (confirms tower detected on Frame \( n \)).

The following pseudo code represents the algorithm applied to a video sequence, pre-analyzed as a a group of independent still images, to confirm detected towers or to detected new ones. The confirmation procedure is applied two times, forward and backward, along the video sequence.

**Data:** Video sequence with \( N \) frames, analyzed as a group of independent still images

**Result:** Video sequence with confirmed towers and new towers detected

\[ n = 1; \]
% Forward confirmation;

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;
        else
          discard tower detected on frame \( n + 1 \);
        end
      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
  \( n = n + 1; \)
end
% (Continues next column)

\( n = N; \)
% Backward confirmation;

while \( n > 1 \) 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;
        else
          discard tower detected on frame \( n - 1 \);
        end
      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
  \( n = n - 1; \)
end

group consecutive frames with detected towers;

**Algorithm 2:** Confirm 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.

Figure 10 shows an example of a tower detected in two subsequent frames. It is clear that the tower configuration was better detected on frame 1 than on frame 2, and the objective is to project the configuration from frame 1 into frame 2 coordinates.

(a) Frame 1 - Conf. score = 80% (b) Frame 2 - Conf. score = 59%

Fig. 10. Configuration of the same tower found in two following frames
To project the lines that form the configuration of a detected tower from frame n to frame n+1, a block match algorithm [8] is applied to several points of each line and then the median motion vector is applied to each line. The method developed to improve the configuration of a group of frames is summarized in the following steps:

1) find which of the frames from the group has 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:
   a) for the vertical structures:
      i. 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;
      ii. 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;
      iii. save the new lines temporarily.
   b) for the horizontal structures:
      i. select left end and four more points, spaced by 16 pixels (half block size), from each of the horizontal lines (H1 and H2);
      ii. compute their motion vector and the resulting median vector, MV1, from frame n to frame n+1;
      iii. select right end and four more points, spaced by 16 pixels, from each of the horizontal lines (H1 and H2);
      iv. compute their motion vector and the resulting median vector, MV2, from frame n to frame n+1;
      v. 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’;
      vi. save the new lines temporarily.
5) compute the score of the new configuration, formed by the new lines, using the method presented before. Replace the old configuration only if the new configuration has a higher score;
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 a tower found in each frame, by comparing its configuration with the configuration of the tower found in the neighbor frame. It starts from the frame where the tower configuration was best detected but using the previous detection if the new one has a lower score, to avoid error propagation.

Figure 11 shows the configuration detected in frame 2 before (a) and after (b) applying the method to project the left part of the structure detected in frame 1.

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

C. Experimental results

The performance of the method developed to detect tower in still images was evaluated using 165 images, 115 of which having a tower. The following parameters were used:

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

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

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

Table II presents the obtained results:

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

**TABLE II**

RESULTS OBTAINED IN THE TEST SEQUENCE
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 an incorrect configuration, which would lead to wrong 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 next. Moreover, the number of false positives will also be reduced.

To show the improvements brought to the results when exploiting the temporal correlation between frames, instead of analyzing them as a group of still images, 20 short duration video sequences containing towers were considered. Table III shows the tower HR resulting from analyzing the sequences as groups of still images and as a video sequence, and the percentage of towers detected with an acceptable configuration before and after the motion analysis.

<table>
<thead>
<tr>
<th>Tower HR</th>
<th>Still images</th>
<th>Video seq.</th>
<th>% detect towers w/ accept. config. Before mot. an.</th>
<th>After mot. an.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>33%</td>
<td>78%</td>
<td>48%</td>
<td>80%</td>
</tr>
</tbody>
</table>

TABLE III
RESULTS OBTAINED TESTING THE VIDEO SEQUENCES

The method was also tested using 10 video sequences containing no towers, to test the resilience against false positives. Again, the sequences were firstly analyzed as groups of independent still images and then as video sequences.

<table>
<thead>
<tr>
<th>Number of false positives using:</th>
<th>Still images</th>
<th>Video sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>41</td>
<td>19</td>
</tr>
</tbody>
</table>

TABLE IV
RESULTS OBTAINED USING 10 VIDEO SEQUENCES WITHOUT TOWERS

These results prove that the method is working as expected, detecting the major part of a tower with a low false positive rate. Moreover, results also proved that, even though it adds complexity to the method, exploiting the temporal correlation between frames greatly improves the method.

III. Nest Detection

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.

After searching for nests on the detected towers, treating the video as a sequence of independent frames, the temporal correlation is exploited in an attempt to improve the results.

A. Nest detection in still images

In order to learn the nests color gamut, some image pixels from nests were manually selected and compared to pixels from the background. It was seen that the color histograms (both on RGB and HSV color space) were centered at different values, which suggested that some colors tend to appear more often on a nest pixel than on the rest of the image. It was also verified that color histograms from the background and from nests can be well described by a Gaussian distribution and by a Rayleigh distribution [9], respectively. Figure 12 shows the PDF of each color, using the values of $\sigma$ and $\mu$ from the data observed.

![PDF of each color](image)

Fig. 12. Probability density functions for background colors (in blue) and nest colors (in red)

A classifier was applied to every image pixel, labeling it as a nest pixel if it respects the following restrictions (thresholds presented in figure 12):
- R, G, B and V pixel values are below the respective threshold;
- H pixel value is above the respective threshold.

After this classification, a morphological closing operator was applied to nest pixels, connecting pixels that are closer than 5 pixels to each other. Then, connected pixels were joined in groups, where groups of pixels are considered as a nest candidate if the number of pixels in that group is:
- greater than the length, in pixels, of the tower’s H2 line on that image; and
- lower than 10 times the length, in pixels, of the same H2 line.

Figure 13 shows an example of an image to which the color classifier was applied, where each group of isolated white pixels represent a nest candidate.
As nests typically have an oval shape, an ellipse was associated to each nest candidate, in order to learn the values of eccentricity of candidates that represent true nests. It was found that nests have eccentricity between 0.5 and 0.9. Discarding candidates with eccentricity outside this interval, results in discarding a great number of false positives. However, trees on the background were also detected that respect these values, as shown in figure 14.

If the detected trees were in line with the tower region of interest, filtering for candidates inside the tower configuration lines would not discard them. However, in the example from figure 14 (and in most of the cases), discarding candidates outside this region (exemplified in figure 15) will eliminate all the false positives, keeping the detections that have a true nest correspondence.

As the tower is "moving" from frame to frame, the number and the position of false positives will vary from frame to frame, but the number of true positives will tend to be the same. Therefore, the temporal correlation between detections in consecutive frames will be exploited in order to reduce the number of false positives.

B. Nest detection in video sequences

The position of the detections will vary from frame to frame but the position, relative to the tower, of true nests will not vary. Hence, the position of each candidate, detected on each frame of a group of frames containing the same tower, is computed, using the referential presented in figure 16.

Then, candidates in different frames, located on the same position, will be associated to the same nest. Candidates are assumed to be in the same position (in the tower referential), if they are distanced by 0.2 or less (or 0.1 or less for candidates on the horizontal structure). The final list of detections will be composed by nests which were detected on more than one third of the frames from the group. This will allow that a nest, which was well detected in part of the video but then got mixed up with the background, to be assumed as a true nest. On the other hand, if a tree is in line with a tower at some point of the video, that tree is not assumed to be a nest, as long as it does not stay aligned with the tower for more than one third of the time that the video is showing that tower.

C. Experimental Results

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:

- \[ \text{Nest HR (\%)} = \frac{\# \text{Well detected nests}}{\# \text{Existing nests}} \times 100; \]
- \[ \text{Nests FPR (\%)} = \frac{\# \text{Detected nests without true correspondence}}{\# \text{Detected nests}} \times 100. \]

Results are shown on table V.

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

TABLE V
RESULTS OBTAINED ON THE TRAINING SEQUENCE ANALYSIS

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 the video as a sequence of correlated frames, as will be seen next.

To test the video analysis, twenty short duration test video sequences were used, containing 20 towers, 13 of which having nests, between 1 and 4 per tower, summing up a total of 23 nests. Table VI shows the global results for the video sequences, analyzed as sets of independent frames, or as a true video sequences. In the still images analysis, performance was evaluated using the same parameters as in the previous subsection.

<table>
<thead>
<tr>
<th>Still images analysis: Nest HR</th>
<th>Nest FPR</th>
<th>Video sequences analysis: Nest HR</th>
<th>Nest FPR</th>
</tr>
</thead>
<tbody>
<tr>
<td>40%</td>
<td>37%</td>
<td>78%</td>
<td>22%</td>
</tr>
</tbody>
</table>

**TABLE VI**
RESULTS OBTAINED ON THE VIDEO SEQUENCES ANALYSIS

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 VII.

<table>
<thead>
<tr>
<th>Tower &quot;in risk&quot; HR</th>
<th>77% 50% with the exact number of nests</th>
<th>30% with more or less detected nests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tower &quot;in risk&quot; FPR</td>
<td>17%</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE VII**
RESULTS OBTAINED ON THE TEST VIDEO SEQUENCES

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.

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 were nests were wrongly detected - tower "in risk" FPR.

- **Tower "in risk" 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 VIII:

<table>
<thead>
<tr>
<th>Video file</th>
<th>Existing towers</th>
<th>Tower HR</th>
<th>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>Global</td>
<td>165</td>
<td>83%</td>
<td>14%</td>
</tr>
</tbody>
</table>

**TABLE VIII**
TOWER DETECTION RESULTS

The final method was tested using 5 videos, with a total duration of nearly one hour. 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.

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 (%)** = \(\frac{\# \text{ wrongly detected towers}}{\# \text{ detected towers}} \times 100\).

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) section.
• Tower "in risk" FPR (%): \[ \text{FPR} = \frac{\# \text{Towers wrongly detected as in risk}}{\# \text{Towers detected as in risk}} \times 100. \]

- 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 IX.

<table>
<thead>
<tr>
<th>Video file</th>
<th>Towers with nests</th>
<th>Tower &quot;in risk&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC_0138.mov</td>
<td>14</td>
<td>67% 68%</td>
</tr>
<tr>
<td>PIC_0140.mov</td>
<td>0</td>
<td>- 0%</td>
</tr>
<tr>
<td>000056.mts</td>
<td>21</td>
<td>83% 33%</td>
</tr>
<tr>
<td>000077.mts</td>
<td>4</td>
<td>100% 18%</td>
</tr>
<tr>
<td>00014.mts</td>
<td>0</td>
<td>- 0%</td>
</tr>
<tr>
<td><strong>Global</strong></td>
<td><strong>39</strong></td>
<td><strong>79% 43%</strong></td>
</tr>
</tbody>
</table>

| TABLE IX |
| Tower "in risk" 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, more particularly, the nest detection, needs to be enhanced before the automatic nest inspection could be considered as a valid substitution of the human inspection.

V. Conclusion

In this work 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 developed work developed 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 work 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.

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 work 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 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.

REFERENCES