

Extraction of Flight Parameters from MFD Video Signal Frames of the C-95 Aircraft

Alexandre Tabosa Santiago^{1,2}, Déborah Fernandes da Silva¹, Marcos R. O. A. Maximo², Carlos H. Q. Forster², Nelson P. O. Leite¹, and Luis G. Moro^{1,2}

¹ Instituto de Pesquisas e Ensaios em Voo (IPEV), Pça Mal. Eduardo Gomes nº 50, Vila das Acácias, São José dos Campos, SP, 12.228-901 - Brazil,

² Instituto Tecnológico de Aeronáutica (ITA), Pça Mal. Eduardo Gomes nº 50, Vila das Acácias, São José dos Campos, SP, 12.228-900 - Brazil
tabosaats@fab.mil.br

Abstract

The airborne Multifunction Display (MFD) is an essential real-time information system used by the flight crew to monitor the aircraft's operational condition, supporting informed decision-making and flight safety. Accordingly, this paper describes a computer vision algorithm for extracting flight parameters from C-95 aircraft MFD image frames acquired directly from the video output signal. Each frame is segmented into areas where parameters are displayed, then divided into sub-areas containing individual digits. The algorithm computes the quantity and spatial distribution of zero-pixel values in each sub-area to determine the most probable numerical value. Using a real flight MFD video recording, test results show that the algorithm extracts up to 35 flight parameters per frame in (4.18 ± 0.37) ms (1σ), corresponding to a worst-case throughput of approximately 220 fps. Reference values were synthesized using the MFD time display and the known video frame rate of 29.97 fps, thereby enabling ground-truth generation without manual annotation. Over 750,000 digit-level test cases yielded a minimum F1-score of 0.9964. Although tailored to the C-95 MFD configuration, the proposed solution is computationally efficient and robust, making it a viable alternative for cost-effective Flight Test Instrumentation (FTI) where dedicated high-precision instrumentation is unavailable.

Key words: Flight Data Extraction, Digit Recognition, Computer Vision, Multifunction Display, Flight Test Instrumentation.

1. Introduction

The continued airworthiness of an aircraft relies on the gathered information during the development and certification flight test campaigns [1]. Therefore, the integration of a complete Flight Test Instrumentation (FTI) system is essential for acquiring accurate and reliable test data that characterizes the test article's operational conditions. In most applications, the FTI installation is a high-cost and time-consuming task due to the required data accuracy and reliability. However, in certain applications, a less intrusive, cost-efficient solution could provide the required information [2]. In this context, one cost-effective alternative is to record flight data directly from the MFD (Multifunction Display) video output signal. The MFD is an electronic cockpit display that consolidates multiple flight parameters, such as airspeed, altitude, and fuel pressure, on a single screen [3]. Unlike camera-based cockpit recording, tapping the display's video output signal provides a clean, stable image of the

rendered flight data, avoiding perspective distortion, reflections, and lighting variability inherent in optical capture, while requiring no additional hardware installation in the aircraft and avoiding the complexity of a full FTI installation.

Nevertheless, manually extracting the required information from video frames is operationally impractical. Therefore, computer vision-based approaches have been explored for MFD flight data extraction, ranging from classical techniques such as Optical Character Recognition (OCR) [4] and template matching [5] to more recent deep learning methods such as Convolutional Neural Networks, including YOLO-based (You Only Look Once) architectures [6], which have significantly advanced image recognition and object detection capabilities [7].

One of the disadvantages of Neural Networks is the high demand for training data and the substantial computational cost of both training

and inference, requiring supercomputing resources in some applications [8, 9]. On the other hand, OCR-based methods are sensitive to non-standard numerical fonts [10]. In addition, template-matching techniques generally require comparisons with multiple predefined templates for each digit, which may increase processing time as the number of monitored parameters grows and often necessitates an additional post-processing step to improve recognition reliability [11].

To address these limitations, this paper proposes a digit identification algorithm to extract up to 35 flight parameters from EMBRAER C-95 Bandeirante MFD video recordings while maintaining high accuracy and low computational cost. The algorithm analyzes the quantity and spatial distribution of zero-valued pixels within binary digit sub-regions, enabling discrimination among all ten classes and subsequent extraction of numerical parameter values from each image frame.

2.1 Methodology

In this section, we describe the algorithm used to locate and identify the flight parameters displayed on the MFD of the C-95 aircraft.

2.1 Preprocessing of collected data

Initially, a pixel-based coordinate system was adopted, in which the origin (0,0) corresponds to the upper-left corner of the acquired frame, and the positive axes extend from left to right and from the top to bottom. Based on this reference, it was possible to determine the coordinates of the rectangles that display the numerical values of the parameters of interest, as illustrated in Figure 1.



Figure 1. MFD of the C-95 aircraft with the parameters to be monitored highlighted in red.

In this particular application, the image frames are acquired directly from the video output signal, hence the typical camera effects such as distortion, illumination variation, saturation, and

vibrations are not present. Thus, the location of all Regions of Interest (ROI) is invariant (red rectangles in Figure 1). Subsequently, to enhance image contrast and digit detection, the RGB pixel intensities are converted to grayscale and then binarized, so that the digits appear in white and the background in black, as shown in Figure 2.

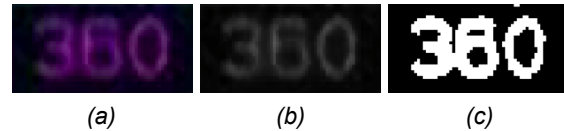


Figure 2. Command digital heading (a) unprocessed, (b) grayscale, and (c) binarized.

2.2 Digit Count Determination

Since the digit count is not uniform across parameters, it must be determined for each ROI individually. In this context, after the binarization stage, a horizontal black-pixel-counting search is performed at half-height within the ROI rectangle, starting from the rightmost black pixel until a white pixel is detected, as illustrated in Figure 3, and the value is stored in a variable “ m ”.



Figure 3. Digit Count determination process with h , the height of the ROI, and m , the variable that represented the black pixel counted.

Subsequently, m is compared against predetermined thresholds: the larger m , the more black pixels are counted and, consequently, the fewer digits occupy the ROI. Once determined, the position of each digit is fully defined, and the ROI is divided into sub-regions of 22×15 pixels (width $\times$ height), each containing a single digit.

2.3 Identification of a Single Digit

For fixed-position digit identification, the algorithm receives a binary matrix ($M_{22 \times 15}$) of pixel intensities (0 = black, 255 = white) and returns the corresponding digit class (0 through 9). Although each digit is not exactly identical in every occurrence, its spatial pixel intensity pattern remains sufficiently consistent. This

property enabled the development of a robust digit-identification algorithm. As an example, Figure 4 illustrates different instances of the digit 2 and their corresponding pixel intensity matrices.

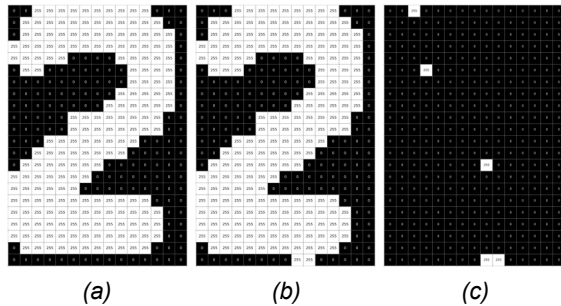


Figure 4. Digit 2 in two different occurrences in (a) and in (b), and the absolute difference in (c).

Given the low variability, it is possible to construct a consistent identification algorithm. Accordingly, four submatrices within matrix $M_{22 \times 15}$ are considered, and for each submatrix, the number of values equal to 0 is counted, as well as the majority value, as illustrated in Figure 5.

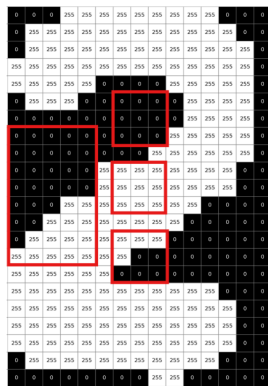


Figure 5: The four submatrices highlighted in red are used to identify the digit 2.

When the features extracted from the four submatrices are insufficient for unambiguous digit classification, additional parameters are defined using the submatrix formed by columns 7, 8, and 9 of $M_{22 \times 15}$. For each column, only zero-valued pixels vertically bounded by 255-valued pixels are considered. The count of such pixels is determined separately for the upper and lower halves of each column, and the arithmetic mean across the three columns is computed for each half, yielding parameters p and q , respectively. Figure 6 illustrates this procedure for the digit 8.

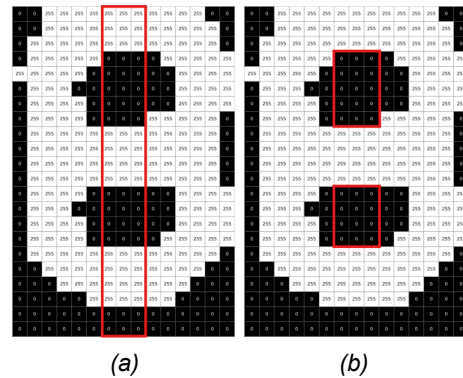


Figure 6. Digit 8: (a) represents the submatrix formed by columns 7, 8, and 9; and (b) represents the auxiliary vectors used to compute parameters p and q on the halves.

With the addition of these new parameters, it becomes possible to differentiate most digits. However, in order to resolve the remaining cases, a more detailed analysis of the matrices shown in Figure 5 is performed, focusing on identifying the rows and columns in which the values equal to 255 are distributed. Since this final step incurs a higher computational cost than the previous two, the algorithm is structured so that it is invoked only when the preceding steps are insufficient, thereby preserving overall computational efficiency. In this way, the computational cost required to identify a digit is limited to binarizing the regions of interest and determining the number of elements equal to zero within submatrices.

Finally, this algorithm is applied to each digit in the ROI, concatenating the identified values in order to determine the numerical value of the parameter.

2.4 Identification of Scrolling Digits

For most of the parameters to be monitored, the digits appear in well-defined positions. However, for the Air Speed and Altitude parameters, the digits can shift vertically within the ROI, as shown in Figure 7, rendering the fixed-position identification procedure insufficient. Therefore, these two parameters require a dedicated detection method.

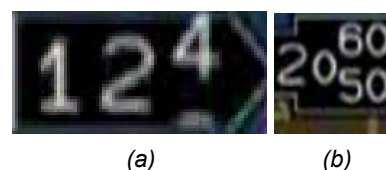


Figure 7. Vertical scrolling behavior of (a) the Air Speed parameter and (b) the Altitude parameter.

2.4.1 Identification of the Air Speed parameter

The number of digits of the Air Speed parameter is first determined. This is performed as follows: starting from the window corresponding to the hundreds digit, a horizontal search is performed along three predefined rows (V_1 , V_2 , and V_3) to verify whether any white pixel is present. If a white pixel is detected in any of the three rows, it indicates that the speed value has three digits (Figure 8).

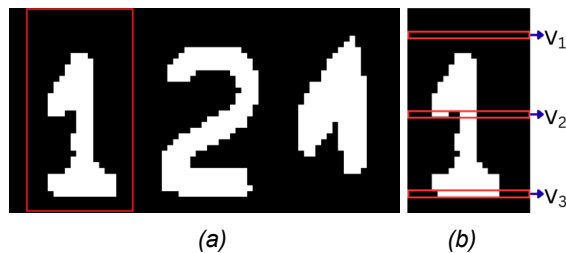


Figure 8. (a) represents the bounding box for the hundred digit in the Air Speed parameter, and (b) represents the three vectors in the bounding box in Figure 8a.

If the hundreds digit is not detected, the same horizontal scan is applied to the tens digit. If a residual white pixel is found within this window, the speed value contains two digits; otherwise, the parameter is considered absent, since the minimum possible speed value has two digits. Consequently, once the number of digits is determined, any detected tens and hundreds are downsampled to the standard sub-region size, since the digits in this parameter exceed the standard pixel sub-region dimensions (22 rows by 15 columns).

Notice that when the speed value ends in 9 or 0, the tens digit may transition. For this reason, only in this case, the last value obtained by the detection algorithm for the tens digit is stored, since the identification algorithm (Section 2.3) can not be reliably applied when the tens digit is not in its expected position, as illustrated by the transition from 130 to 129 in Figure 9. Similarly, the same concept is applied to the hundreds digit when the tens digit is 9 or 0.

Therefore, the detection algorithm is not applied to the tens digit in this situation, and the stored value is used instead. This value varies by one when the unit digit transitions from 9 to 0 or

from 0 to 9. In this context, the identification algorithm (Section 2.3) can be applied only to the tens digit when the first transition from 0 to 1 or from 9 to 8 occurs in the unit digit, since in these cases the tens digit is guaranteed to be fixed in its expected position.

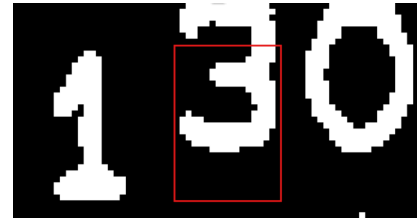


Figure 9. Air Speed transition from 129 to 130 shows that the tens digit is not in its expected position under scrolling conditions.

The limitation of this approach arises when, during the first measurement acquisition, a transition occurs simultaneously in both the unit digit and the tens or hundreds digit. In this situation, the detected number may be incorrect because the stored value for the tens digit has not yet been initialized.

For the determination of the unit digit, the region near the Air Speed value that contains some markings of the measurement scale is considered, as illustrated in Figure 10. On the binarized measuring ruler, as the test progresses, the white markings in the right margin may slide along the ROI rows. Therefore, these marks are at specific positions for each unit digit. From this region, the penultimate column is extracted and analyzed as a vector. First, starting at the 36th position and moving upward along the vector (toward decreasing indices), the algorithm searches for the first white pixel and stores the number of black pixels encountered (d_1). Next, starting from the 38th position and moving downward (toward increasing indices), the same procedure is repeated, storing the number of black pixels encountered (d_2) until the first white pixel is encountered. Finally, the two stored values are subtracted, and the result is compared with predefined intervals to determine the unit digit.

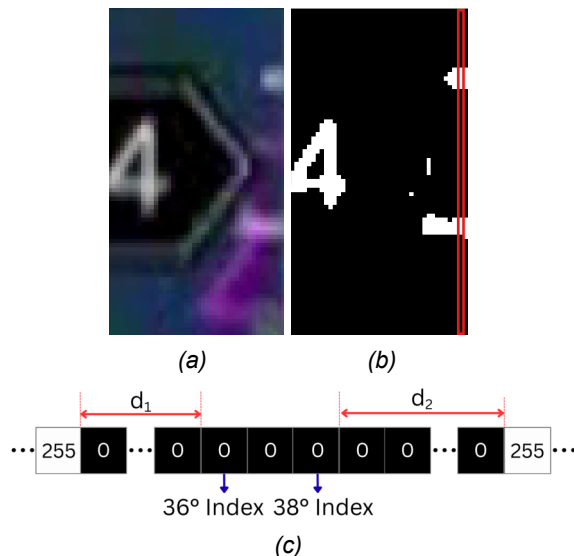


Figure 10. (a) The region of interest used to determine the unit digit of the Air Speed, (b) The binarized image corresponding to (a) with the penultimate column highlighted in red, and (c) Part of the column vector used to determine the unit digit.

2.4.2 Identification of the Altitude parameter

The Altitude parameter has exactly four digits, and the unit digit is always 0. Moreover, the identification of the hundreds and thousands digits is performed analogously to the identification of the tens and hundreds digits in the Air Speed parameter, respectively.

Furthermore, two distinct approaches were developed to determine the tens digit of the Altitude parameter, since this digit exhibits characteristics not observed in the other three digits.

2.4.2.1 First approach to detect tens digit in Altitude parameter

This approach is fully analogous to identifying the unit digit of the Air Speed parameter, since the altitude parameter also has a measurement scale, as illustrated in Figure 11. In this case, the vector analyzed is the second one of the binarized corresponding ROI.

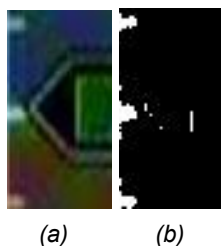


Figure 11. (a) The region of interest used to determine the unit digit of the Altitude, (b) The binarized image corresponding to (a)

2.4.2.2 Second approach to detect tens digit in Altitude parameter

This approach was carried out as follows: initially, a routine was implemented to determine the exact position at which the tens digit appears. Starting from the expected position corresponding to the base of the digit, assuming alignment with the complete number, a horizontal search is performed along the last row of the tens digit's bounding box to check for the presence of white pixels.

If white pixels are found, the bounding box is shifted down by 1 pixel, and the scan is repeated. This process continues until the scanned row contains only black pixels, which indicates the correct position of the tens digit, as illustrated in Figure 12.

If no such row is found during the downward search, the bounding box is reset to its initial position and shifted up by 1 pixel instead, after which the same scanning logic is applied in the upward direction.

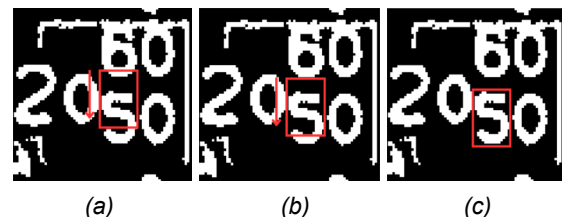


Figure 12. Routine for identifying the tens digit in the altitude parameter, even under scrolling conditions. (a) Initial position, (b) Descending, and (c) Stop condition.

3. Results

In this section, we evaluate the algorithm's performance using data collected from a real flight test.

3.1 Processing Time

The algorithm was implemented in Python 3 using PyCharm as the development environment, with OpenCV, NumPy, and Pandas, and processing performed on a CPU (12th Gen Intel(R) Core(TM) i5-12500H, 2.50 GHz) with 16.0 GB of RAM. The code was provided with an input consisting of an MFD recording from an 80-minute and 42-second test flight of the C-95 aircraft and produced a CSV-format table with 145,136 rows and 35 columns representing the monitored

parameters. The total processing time was 10 minutes and 8 seconds. Additionally, the processing time was computed for each frame, from which the mean and standard deviation were obtained, resulting in a processing time of (4.18 ± 0.37) ms for the extraction of 35 flight parameters from a single frame.

3.2 Algorithm Evaluation

The evaluation is performed by comparing the true value of a digit with the value predicted by the algorithm; the larger the number of test cases, the greater the reliability of the predictor's precision and accuracy. However, in this case study, comparison with ground truth is not immediate, since frame-by-frame true values of the monitored parameters are unavailable. Thus, in order to mitigate this limitation, a substantial number of test cases were synthetically generated based on the "time" parameter, which represents the current timestamp displayed on the real flight MFD in hours, minutes, and seconds. Since this parameter increments deterministically with time, its frame-level ground truth can be reconstructed from the known initial and final timestamps combined with the video frame rate, without requiring manual annotation.

In this test scenario, the parameter varies as follows: starting from the initial timestamp of 12:20:52, the video proceeds normally until 13:03:59, at which point a discontinuity occurs, and the next timestamp is 13:42:28, which then proceeds normally until the final frame timestamp of 14:20:03. This discontinuity is due to the fact that the recording corresponds to a clipped segment extracted from a real flight test. Accordingly, given the number of frames in which these timestamps appear and the recording frame rate of 29.97 fps, frame-level timestamps were synthesized in a manner closely approximating real conditions.

Furthermore, in order to avoid errors associated with digit transitions and time synchronization, the first, second, penultimate, and last frames of each hour–minute–second set were discarded. Following this approach, a dataset comprising 125,768 timestamps was generated; since each timestamp consists of 6 digits separated by ":", the effective total number of test cases is 754,608 $(125,768 \times 6)$. Consequently, after the

predictor evaluated all test cases, a confusion matrix was constructed for the ten possible digit classes, as shown in Figure 13.

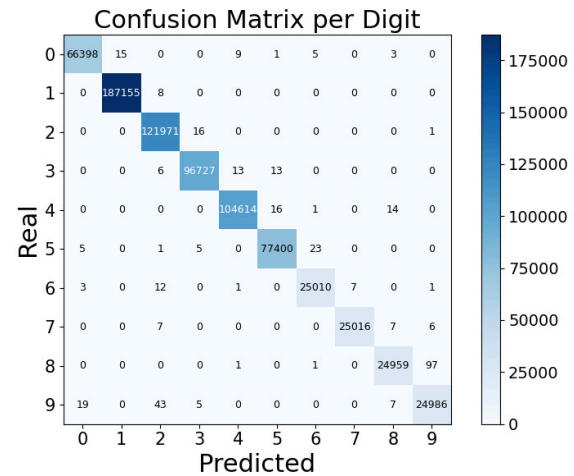


Figure 13. Confusion matrix for all ten possible digits.

From the confusion matrix, it was possible to derive Table 1, which presents the Precision, Recall, and F1-score metrics for all digits.

Table 1. Precision, Recall, and F1-score metrics for the ten possible digits.

Digit	Precision	Recall	F1-score
0	0.9996	0.9995	0.9995
1	0.9999	1.0000	0.9999
2	0.9994	0.9999	0.9996
3	0.9997	0.9997	0.9997
4	0.9998	0.9997	0.9997
5	0.9996	0.9996	0.9996
6	0.9988	0.9990	0.9989
7	0.9997	0.9992	0.9995
8	0.9988	0.9960	0.9974
9	0.9958	0.9970	0.9964

Table 1 shows that the metrics are close to their maximum values (1.0); consequently, the predictor can be considered near-ideal, exhibiting no significant bias between false positives and false negatives, which indicates a robust model for digit classification. Furthermore, the comparatively lower

performances are associated with digits 8 and 9, with the lowest performance observed for digit 9, which presents an F1-score of 0.9964.

3.3 Measurement errors

In some cases, an overlap may occur between the digit from the immediately preceding frame and the digit corresponding to the subsequent frame, since the transition is not instantaneous. In such situations, the algorithm receives as input a digit different from the previously expected ones, which may lead to detection errors, as illustrated in Figure 14.

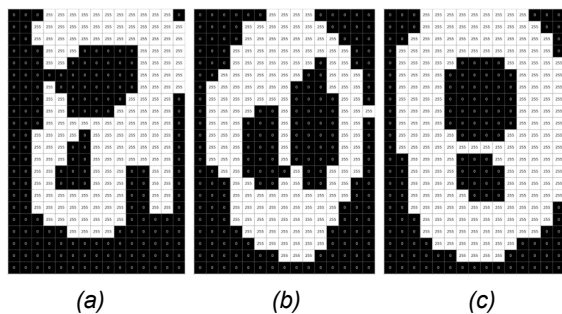


Figure 14. (a) Transition between 7 to 8, (b) Transition between 0 to 1, and (c) Transition between 0 to 9.

Furthermore, the transition from digit 8 to 9 results in more misclassifications due to the high visual similarity between these digits, as illustrated in Figure 15.

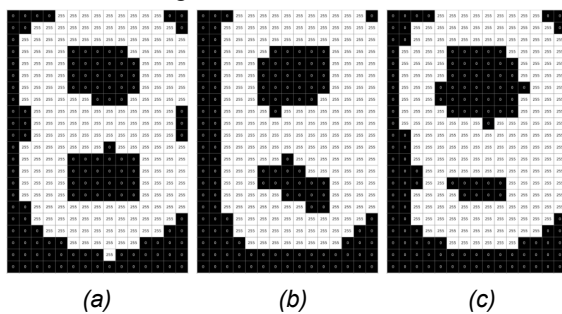


Figure 15. Digit 8 on (a), transition between 8 to 9 in (b), and digit 9 on (c).

Degenerate cases occur when the digit does not match the expected pattern. This behavior is associated with the binarization stage, in which pixels with intensity values close to the threshold may be incorrectly classified. As a result, pixels that should be identified as foreground (white) may be assigned to the background (black), leading to incomplete or distorted digit representations, as shown in Figure 16.

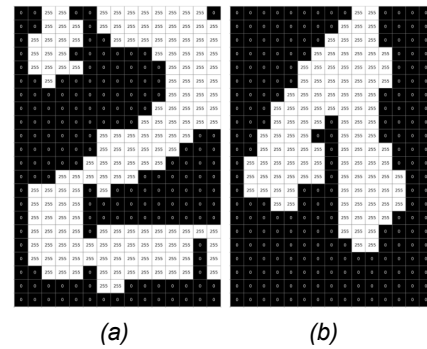


Figure 16. Degenerate cases for digit 2 in (a), and for digit 4 in (b).

When a sudden change in altitude occurs, the indicator may display digits scrolling in such a way that the number becomes vertically distorted. Consequently, the displayed digits differ from the patterns considered in the construction of the identifier, as illustrated in Figure 17.



Figure 17. Numeric altitude values of 2780 (right) and 2760 (left) under abrupt scrolling conditions.

From this perspective, in such cases, the second approach (Section 2.4.2) yields more outlier identifications, as shown in the graphical comparison between the approaches under abrupt motion conditions, as seen in Figure 18.

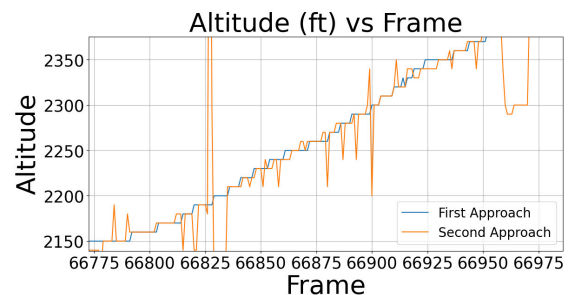


Figure 18. Altitude versus frames for both approaches under abrupt motion conditions, highlighting the increased occurrence of outliers in the second approach (Section 2.4.2)

Furthermore, during periods of relative altitude stability, the first approach (Section 2.4.1) exhibits larger fluctuations in value due to the sensitivity of the binarization process applied to the vector used to determine the corresponding value, as shown in Figure 19.

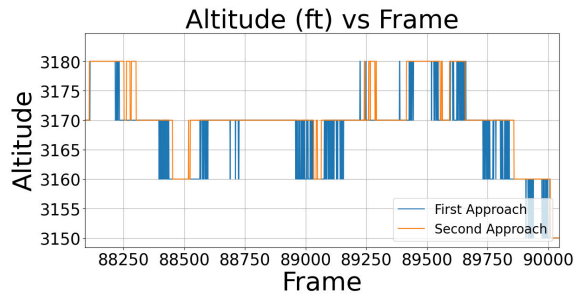


Figure 19. Altitude versus frames under stable conditions, showing increased fluctuations in the first approach (Section 2.4.1).

Figure 20 presents the altitude versus frame count for the first approach (Section 2.4.2), excluding 28 data points identified as outliers due to altitudes exceeding 5000 ft. This approach was selected because, under abrupt motion conditions, it does not exhibit a significant number of outliers. Furthermore, although the first approach presents fluctuations under stable conditions, these fluctuations tend to remain within the full-scale range of the measurement.

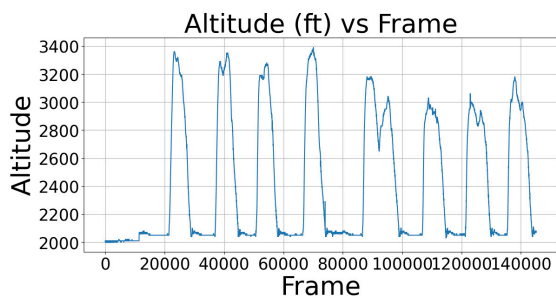


Figure 20. Altitude (ft) versus frame count, with 28 outliers omitted.

Figure 21 illustrates the Air speed (kt) versus frame count, excluding two outliers, one above 300 knots per second and another below 40 knots per second.

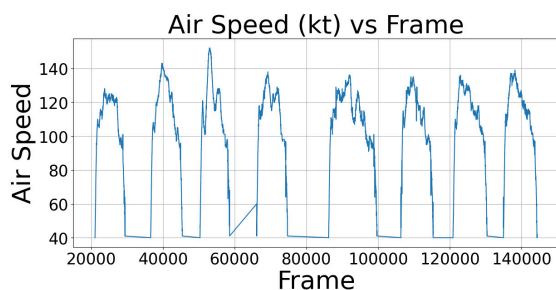


Figure 21. Air speed (kt) versus frame count, with 2 outliers omitted.

Figure 22 presents the TOT (Totalizer in lb) versus frame count, where six outliers can be

observed, five of which are associated with digit transition errors and one related to digit degeneration. Although these outliers can be filtered, for instance, by discarding any measurement whose relative change with respect to the previous one exceeds a given threshold, they were intentionally retained in this graph to present the data without post-processing.

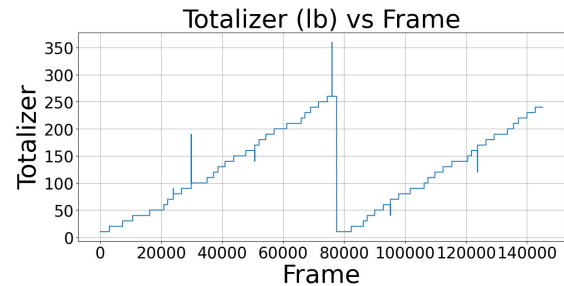


Figure 22. TOT (lb) versus frame count, including all data points.

Figure 23 illustrates the FOB (Fuel on Board in lb) versus frame count, in which 20 outliers are observed, resulting from digit degeneration and transition errors.

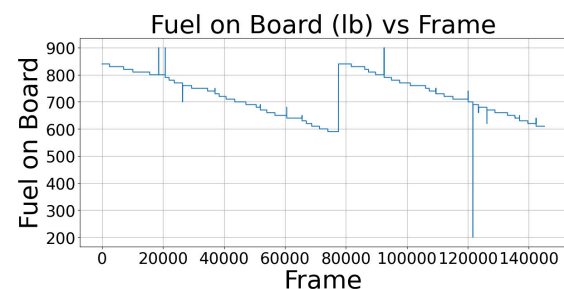


Figure 23. FOB (lb) versus frame count, including all data points.

From this perspective, the graphs in Figures 18-21 show consistent results, although outliers and, in some cases, discretization errors are observed. The TOT measurement, for instance, varies only in increments of 10 units; therefore, variations with a magnitude greater than or equal to 20 units likely indicate that the digit is in a state of degeneration or transition, characterizing a discretization error.

Thus, in order to mitigate inconsistent values and outliers, one may either discard such points or infer their most probable values based on the immediately preceding and subsequent samples, while taking into account the discretization of the measurement.

Although the identifier yields results close to the ideal, it is specialized for digits in a 22×15 format under a specific operating mode. Therefore, Figure 23 presents examples of formats for which the identifier fails, as well as the expected format for the algorithm.

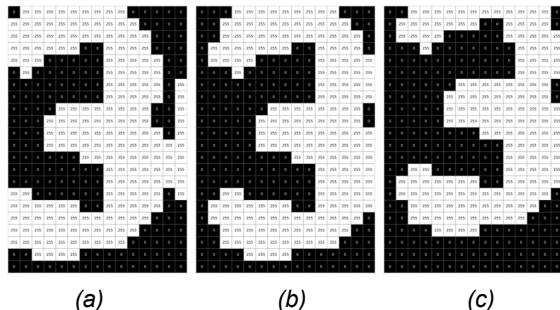


Figure 23. For the digit 3, (a) Expected format, (b) Unexpected format shift to the left, and (c) Unexpected format shifted upward

Thus, to fully utilize the algorithm, it is essential to define image cropping regions for each digit of each parameter, which may require considerable time during data preparation. Nevertheless, once this process is completed, the algorithm achieves performance that meets the requirements for real-time flight parameter extraction while maintaining low processing time.

Furthermore, although the identifier yields results that meet the requirements for the C-95 aircraft, it does not generalize readily to other aircraft, since they may employ distinct MFD layouts and numerical fonts not implemented in the algorithm.

4. Conclusion

This paper proposes a method to identify and register numerical values associated with 35 flight parameters of the C-95 aircraft, including both static and scrolling parameters. Using 754,608 test cases, the identifier demonstrated near-ideal performance, with the lowest F1-score being 0.9964, observed for digit 9. Furthermore, in an 80-minute and 42-second flight test, the algorithm extracted 35 parameters from a single frame in (4.18 ± 0.37) ms, enabling real-time applications at rates of up to approximately 220 fps. In addition, although errors related to digit transitions and digit degeneration are present, it is still possible to determine the most probable values of incorrectly estimated points through removal or

inference techniques. Moreover, the proposed method was developed specifically for the C-95 case and does not generalize immediately to other aircraft. Finally, the results obtained for the identifier, particularly regarding processing time, are significant, demonstrating potential for implementation in the routine operation of C-95 flight test activities.

Future work includes extending the method to support other aircraft types. Moreover, other possible works include deploying the system with a physical camera directed at the aircraft panel, which would introduce additional challenges such as perspective distortion, variable lighting conditions, and motion blur that are absent in direct screen recordings.

References

- [1] C. B. de Mendonça, E. T. da Silva, M. Curvo, and L. G. Trabasso, "Model-Based Flight Testing," *Journal of Aircraft*, vol. 50, no. 1, pp. 176–186, 2013, doi: 10.2514/1.C031778.
- [2] J. M. Ricciardi and C. Minwalla, "Non-intrusive Flight Test Instrumentation using Video Recognition," in *AIAA Modeling and Simulation Technologies Conference*, 2016, doi: 10.2514/6.2016-1435.
- [3] J. Knight, "The Glass Cockpit," in *Computer*, vol. 40, no. 10, pp. 92-95, Oct. 2007, doi: 10.1109/MC.2007.365.
- [4] RIBEIRO SILVA, Samara. Using Computer Vision to Gather Flight Testing Data. 2022. 64f. Final paper (Undergraduation study) – Instituto Tecnológico de Aeronáutica, São José dos Campos.
- [5] Zhang, Maodi; Zhou, Yuanhui; Gu, Qingfan: "Design of Image Recognition Algorithm for Critical Flight Parameters in Civil Aircraft Display System", IOP Conference Series: Materials Science and Engineering, Volume 751, Issue 1, IOP Publishing, 2020.
- [6] Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A.: "You Only Look Once: Unified, Real-Time Object Detection", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016.
- [7] Vasileiadis, A.; Alexandrou, E.; Paschalidou, L.; Chrysanthou, M.; Hadjichristoforou, M.: "Artificial Neural Network and Its Applications", Lecture Notes in Networks and Systems, Springer, 2024.

- [8] G. A. Melo, M. Máximo and P. A. Castro, "High Speed Marker Tracking for Flight Tests," in IEEE Latin America Transactions, vol. 20, no. 10, pp. 2237-2243, Oct. 2022, doi: 10.1109/TLA.2022.9885171.
- [9] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The Computational Limits of Deep Learning," arXiv:2007.05558 [cs.LG], 2022. [Online]. Available: <https://arxiv.org/abs/2007.05558>.
- [10] E. Zacharias, M. Teuchler, and B. Bernier, "Image Processing Based Scene-Text Detection and Recognition with Tesseract," arXiv:2004.08079 [cs.CV], 2020. [Online]. Available: <https://arxiv.org/abs/2004.08079>.
- [11] N. S. Hashemi, R. B. Aghdam, A. S. B. Ghiasi, and P. Fatemi, "Template Matching Advances and Applications in Image Analysis," arXiv:1610.07231 [cs.CV], 2016. [Online]. Available: <https://arxiv.org/abs/1610.07231>.