

Software Module for Unmanned Autonomous Vehicle's On-board Camera Faults Detection and Correction

Egor Domnitsky^a, Vladimir Mikhailov^a, Evgeniy Zoloedov^a, Danila Alyukov^a, Sergey Chuprov^a, Egor Marinenkov^a and Ilia Viksnin^a

^aITMO University, Kronverksky Pr. 49, bldg. A, St. Petersburg, 197101, Russia

Abstract

Sensor devices proper operation is crucial for the localization and movement of unmanned autonomous vehicles. On-board cameras and computer vision technologies are used in many models of unmanned vehicles and robotic devices for recognizing surrounding objects. However, malfunctions in the procedures of receiving or processing a video stream can significantly affect vehicle's safety and endanger other road users. In this paper, we review existing methods for detecting and correcting faults occurring in video stream from on-board camera. A real-time fault detection and correction software based on existing solutions is proposed. Moreover, we perform demo-setup with a test video fragment to assess the software performance in different light conditions. The video of the software operation in the demo-setup process is provided. The proposed approach and software developed on its basis showed appropriate performance in daylight conditions.

Keywords

UAV, Fault detection, Fault correction, On-board camera

1. Introduction

Nowadays, with the intensive technology development, the urban population and the private transport amount are expected to continuously grow in the coming years. In perspective, congested urban traffic will require precise management in automation and optimization aspects. As stated in [1], the popular "Smart City" concept implementation poses a variety of challenges for transportation area, such as ensuring the safety of road users, traffic optimization, accident prevention and other significant issues.

One of the possible solutions to meet these challenges is the unmanned autonomous vehicles (UAVs) integration. However, such UAVs should be reliable and conform the functional and information security and safety requirements. UAVs on-board devices for collecting and trans-

Proceedings of the 12th Majorov International Conference on Software Engineering and Computer Systems, December 10–11, 2020, Online & Saint Petersburg, Russia

✉ egor.dom0923@gmail.com (E. Domnitsky); v.mihajlov2001@gmail.com (V. Mikhailov); evgenijzoloedov@gmail.com (E. Zoloedov); d@alyukov.net (D. Alyukov); chuprov@itmo.ru (S. Chuprov); egormarinenkov@gmail.com (E. Marinenkov); wixnin@mail.ru (I. Viksnin)

🆔 0000-0002-4369-6736 (E. Domnitsky); 0000-0002-0178-4631 (V. Mikhailov); 0000-0002-8039-1018 (E. Zoloedov); 0000-0002-9667-8259 (D. Alyukov); 0000-0001-7081-8797 (S. Chuprov); 0000-0001-9895-239 (E. Marinenkov); 0000-0002-3071-6937 (I. Viksnin)



© 2020 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

mitting data, and performing localization and movement (sensors, cameras, transmitters, etc) are needed to be supervised by a special subsystem that is capable of performing real-time fault detection procedures. By the detection of faulty, defective or maliciously attacked elements, such subsystem prevents negative effects on joint on-board systems and on other vehicles. For example, it is critical for an on-board camera to have a full view on the road. Especially, when it is responsible for providing other joint systems with environmental information used for orientation. In vehicular ad hoc networks (VANETs) disinformation can lead to critical consequences, such as traffic accidents, human casualties or deaths, and financial losses.

In the present paper, we analyzed algorithmic methods for detecting and correcting possible malfunctions in the on-board camera video stream, which is used by the UAV for perception and localization purposes. Moreover, we develop and assess our custom software that allows to process, detect and correct the malfunctions in on-board camera's video stream in real-time conditions.

The paper is organized as follows. Section 2 contains an overview of existing video stream analyzing and processing algorithms for identifying and correcting on-board camera faults. Section 3 describes the goals and objectives of the present study. Section 4 contains a description of chosen methods for detecting and correcting selected malfunctions. Section 5 contains a demonstration of the developed software, testing approach description, and the results overview for each implemented malfunction. Section 6 states the conclusions and plans for further research.

2. Related Work

Since autonomous technology started to actively develop, various solutions for camera malfunction detection have been introduced. In [2] the authors proposed a frame-by-frame video stream processing algorithm in a video surveillance system. Each frame is processed bodily and divided into blocks. From each frame (and frame blocks) an image of brightness, brightness gradient, borders, intensity and borders direction (for example, using the Sobel operator), RGB of frame, HSV of frame, as well as mean values of all named parameters are obtained. It detects objects movement in the frame and creates a motion picture for each frame. Then, the analyzed input frame is compared with the previous "saved" frame, which is also fully analyzed. A "saved" frame is a blocks' array on which no movement or deviations (malfunctions) were recorded. When comparing the input and "saved" frames, a malfunction candidate image is formed - the block of the saved frame and corresponding one of input - their parameters and mean values are compared, and if the difference exceeds a certain threshold, this block is considered as a malfunction candidate. Blocks that were not identified as faulty and did not participate in the motion picture renew the corresponding one's in "saved" frame. Further, the motion picture is applied to the formed picture of candidates, and the blocks participated in the motion are excluded from the picture of candidates. Thus, the malfunction picture is formed. In turn, based on the frames parameters comparison and the set of compared frames, several fault patterns are formed, one for each fault type. The sets of compared parameters responsible for certain malfunctions are also presented in the paper. The proposed algorithm is computing-power consuming and can be used effectively only on stationary cameras.

In [3] a morphological analysis for simple malfunctions, and machine learning approach to deal with detecting complex issues are proposed. The idea is to detect: lack/excess of brightness by counting number of gray-level pixels; saturation error by counting pixel number with high saturation; freezing by counting identical frames; frame loss by counting blue/black frames; broken frame by gradient mapping evaluation; excess of palette colors (color cast) by color space deviation evaluation. A convolutional neural network is used to detect frame banding malfunctions, overlaps, and image blur. In terms of mobile cameras, morphological analysis may operate rather effectively due to its simplicity. The use of convolutional neural network is a perspective approach, however, according to the hardware restrictions and deep learning requirements, it might not be an effective solution.

In [4] the authors looked into the problem of stereoscopic 3D (S3D) color correction in terms of visual inconsistency, which leads to faulty frame perception. In the paper, a color correction algorithm for S3D images and videos is proposed that simultaneously deal with global, local, and temporal color inconsistencies. Algorithm is split into three steps:

- coarse-grained color grading for global color matching;
- fine-grained color correction;
- local color correction.

These steps allow to ensure structural consistency before and after the color correction procedure. Moreover, the display functions for each color channel are changed gradually with the video stream to avoid abrupt temporal color deviations. Experimental results showed that the proposed algorithm is superior to many modern image and video color correction techniques.

3. Problem Statement

The aim of this work is to develop a multiple faults detection software that allows real-time processing and correcting of UAV on-board camera's video stream. The software should detect the malfunctions and apply correction measures (if possible), and notify an operator on it.

To reach the research aim, following tasks are introduced:

1. to examine, which video stream properties can be obtained for further processing;
2. to define the approach for video stream processing;
3. to determine most common camera malfunctions;
4. to analyze the proposed malfunction detection methods and define the most appropriate;
5. to analyze existing correction methods and define the most appropriate;
6. to implement selected methods in a software to perform video stream real-time detection and correction and test it;
7. to provide conclusions on the study performed.

Maintaining UAV's functionality is a complex challenge. An integral part of this issue is the initial problem detection. Before the safety responsible subsystem apply measures to control the functionality, it is necessary to determine possible damage, as this characteristic allows to determine possible measures for its mitigation or harmful effect minimization. Accordingly, the on-board systems' functionality control is divided into two stages:

1. problem detection and determining its nature. It is necessary to determine the on-board system's parameters for its further self-diagnosis and malfunctions detection;
2. applying corrective measures. If it is possible, return the system to normal operation without physical interaction via software correction algorithms application.

4. Correction Measures Overview

In the present study, we consider three common on-board camera faults: color cast, image blur, and lens overlap by another objects or substance, e.g. dirt. The approach for a video stream processing is a frame-by-frame analysis.

4.1. Color cast

4.1.1. Detection

To detect this malfunction, an approach proposed in [3, 5, 6] is used. In the RGB color space, it is difficult to determine the color deviation of a frame in color space, due to all three pixel "coordinates" in the color space are responsible for color. The researchers proposed a solution that allows translating the image into the Lab color space, where L is responsible for the pixel color brightness but not for the color component; two other channels are responsible for the color components: a - in the positive semi-axis (up to +127) for the magenta color, in the negative (-128) for green; and b component - in the positive semi-axis for yellow, and in negative semi-axis for blue. Due to the color space change we can place a point (pixel) on the color plane (a, b), and therefore determine the deviation of the points density relative to the axes a and b intersection ((0, 0) point). Density here means the pixels concentration in some point area in the color space (in the (0, 0) point area in normal case). The density can be characterized using two calculated parameters: D - the average chromaticity (distance from the (0, 0) point to the averaged density "center") defined by (1) and (2), and M - the average chromacity momentum defined by (3) and (4), i.e. average distance from the averaged density center to the points surrounding it, namely forming the density itself (average radius of the density). The factor (Cast Factor) $K = D/M$ indicates the color cast presence: the larger it is (i.e., the larger D and the less M), the more distinguishable the color deviation is.

$$mean_a = \frac{\sum_{i=1}^{i=H} \sum_{j=1}^{j=W} a(i, j)}{H \times W}, mean_b = \frac{\sum_{i=1}^{i=H} \sum_{j=1}^{j=W} b(i, j)}{H \times W} \quad (1)$$

$$D = \sqrt{mean_a^2 + mean_b^2} \quad (2)$$

$$M_a = \frac{\sum_{i=1}^{i=H} \sum_{j=1}^{j=W} |a(i, j) - mean_a|}{H \times W}, M_b = \frac{\sum_{i=1}^{i=H} \sum_{j=1}^{j=W} |b(i, j) - mean_b|}{H \times W} \quad (3)$$

$$M = \sqrt{M_a^2 + M_b^2} \quad (4)$$

In [7] the authors proposed to use the interval from 1 to ≈ 2 as normal values of the K factor. If $K > 2$, then a fault is detected and warning message in the operator interface is displayed. Cases when $K < 1$ are considered as normal depending on the on-board camera device characteristics and overall luminance. The precise K factor's thresholds are required to be set according to a specific camera model.

4.1.2. Correction

In [8] Gasparini and Schettini proposed several methods for color cast correction. The measured values of the RGB frame are different for various viewing conditions, however human's eyes are capable of compensating the light source chromacity and approximately retain the scene colors. This phenomenon is known as chromatic adaptation. Digital imaging systems cannot account for these shifts in its color balance. In order to restore the original frame chromacity under different lighting and viewing conditions, the measured RGB channels' values need to be converted. These conversions are called chromatic adaptation models. The chromatic adaptation model converts RGB channel viewing condition set values to such matching the required ones.

The gray world algorithm assumes that if there is an image with enough color variations, the average values of its RGB channels is equal to the gray value. Thus, in an image taken with a digital camera in a particular lighting environment, the color cast caused by this lighting via this algorithm. After the gray value is selected, each color channel is scaled by applying a Von Kries transformation adapted to RGB space, which is represented by (5). Von Kries transformation coefficients are defined by (6). The averages of the RGB channel are calculated according to (7). The gray value is defined according to (8).

$$\begin{aligned} R_{new} &= k_R \times R \\ G_{new} &= k_B \times G \\ B_{new} &= k_B \times B \end{aligned} \quad (5)$$

$$\begin{aligned} k_R &= Gray_R / R_{avg} \\ k_G &= Gray_G / G_{avg} \\ k_B &= Gray_B / B_{avg} \end{aligned} \quad (6)$$

$$\begin{aligned} R_{avg} &= \sum R_i / n \\ G_{avg} &= \sum G_i / n \\ B_{avg} &= \sum B_i / n \end{aligned} \quad (7)$$

$$Gray_R = Gray_G = Gray_B = \frac{R_{avg} + G_{avg} + B_{avg}}{3} \quad (8)$$

In fact, most color balancing/restoring algorithms work well only in the certain accepted assumptions conditions. For gray world algorithm correct operation, the frame/image need to be sufficiently colorful, otherwise the results can be distorted or gray-prevailing.

4.2. Blur

4.2.1. Detection

To detect frame blur, we adopt and apply the algorithm described in [9]. The main idea of this approach is to calculate the frame edge dispersion and compare it with a threshold value. It can be done via the 2nd derivative utilization: if the derivative changes its sign in some point, this point is a function graph inflection point. The number of black to white transitions (dispersion) is counted in the algorithm.

The algorithm steps for image blur detection are described below.

1. Get the input frame.
2. Convert the input frame from the RGB to the GRAY color space to avoid possible interference in the estimation.
3. Apply the Laplace Operator. At this stage, all object's edges are outlined in the frame.
4. Count the transitions number (dispersion).
5. Compare the obtained value with the predefined threshold. The threshold is calculated experimentally as it depends on many factors, such as illumination and objects number in the frame. If the value is greater than the threshold, the image is not blurry, otherwise blur is detected.

4.2.2. Correction

The blur correction algorithm steps are provided below.

1. Calculate the absolute difference between the current and the next GRAY frames.
2. Count the number of pixels above and below normal.
3. If the threshold is exceeded, apply the Sobel Operator, where A is the input frame matrix, G_x is the x 's derivative (9), G_y is the y 's derivative (10) and G is the xy 's derivative (11).

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \times A, \quad (9)$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \times A, \quad (10)$$

$$G = \sqrt{G_x^2 + G_y^2}, \quad (11)$$

4. Display borders in the frame.

4.3. Dirt detection

To detect interfering objects and substances overlapping the lens, the following algorithm based on calculating the difference between adjacent frames is applied. It was proposed by one of authors of the present paper. The algorithm is organized as follows:

1. Get the input frame.
2. Convert the input frame from the RGB to the GRAY color space to avoid possible interference in the estimation.
3. Calculate the absolute difference between the adjacent frames.
4. If the difference exceeds certain threshold, the percentage of different pixels between the adjacent frames is calculated.
5. If this percentage exceeds certain threshold, the Sobel Operator is applied to outline the overlap edges.

5. Results

In this section we provide a demonstration of on-board camera's faults detection and correction by the developed software. The program contains several modules, each of which includes a class with variables and methods necessary for the corresponding detection/correction algorithm. For developing purposes C++ programming language was used, along with the OpenCV library for frame-by-frame video stream processing, and the Qt library for interface part (handy track bars to manually set and apply artificial malfunctions on a video-stream). Figure 1 represents the graphical user interface (UI) of the developed software. On the left top side of the interface one can observe the input frame with artificial malfunctions applied, on the right top side - an output frame with color cast corrected or lens overlapping marked. On the left side of the UI buttons for basic file opening and image rotating are placed. In the middle of UI the control track bars for artificial malfunctions: color channel balance (color cast), dirt (for overlapping), and blur respectively are placed. On the right side one can observe a fault indication panel.

For testing purposes, artificial faults were manually applied to the original frame: image blur, artificial spots (overlapping), and a change in the frame color balance. The conducted demo-setup of the developed software was recorded and can be accessed publicly¹, further we will reference to the video time-codes. The testing was performed on the video stream fragment, which also can be found in a public access². Module was tested on a computer equipped with eight-core processor, able of 150 GFlop/s computing performance, and ran effectively with FPS of 80-100. For example, processors with 200 TFlops/s are already available on the market, and even designed specially as platforms for UAV's systems development³. This allows us to consider our software module not demanding in terms of computing resources.

5.1. Color cast

The conducted testing showed that fault detection algorithm performed well in daylight conditions and was able to detect even slight color deviations. The correction algorithms also perform well in daylight conditions; even if a slight deviation remains, it is almost indistinguishable to the human eye and incapable of disrupting the correct perception of color by machine. However, the algorithm loses its effectiveness in low light conditions, as can be seen from the testing video¹ on 2:23.

¹<https://youtu.be/PdSda2QE1yg>

²<https://bdd-data.berkeley.edu/>

³<https://www.nvidia.com/ru-ru/self-driving-cars/drive-platform/>

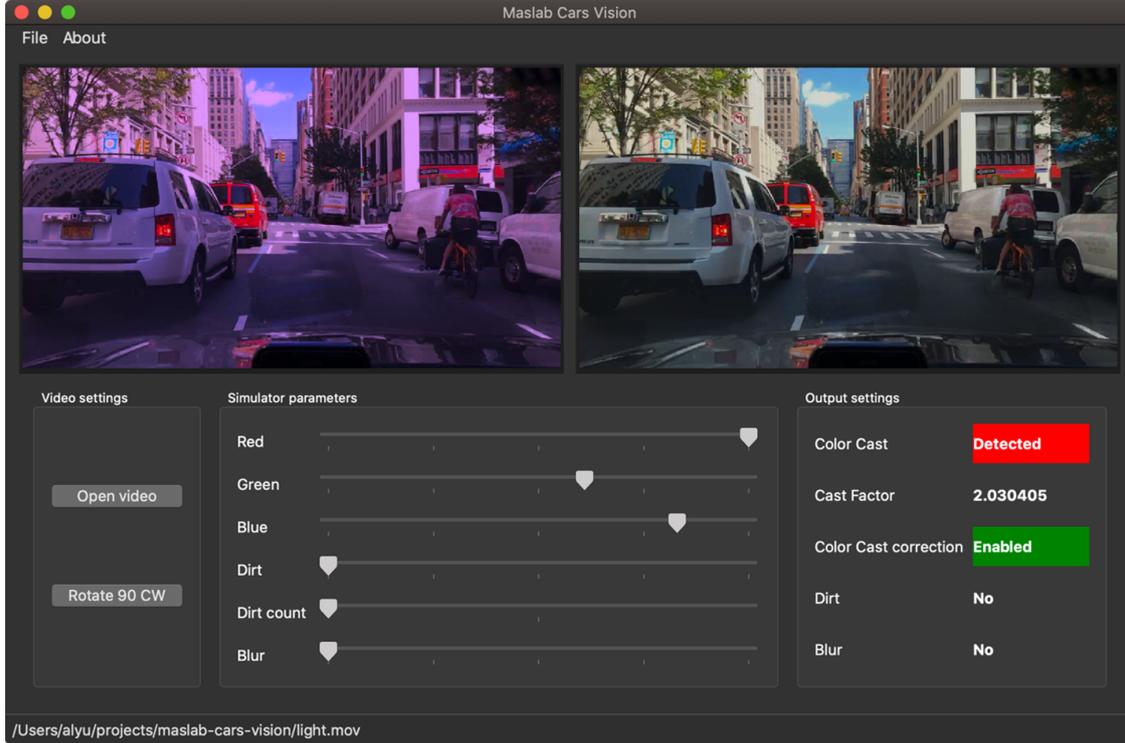


Figure 1: Demonstration of the developed software’s graphical user interface

To increase the algorithm’s efficiency in low light conditions, it was decided to increase the input frame’s brightness and contrast so that the algorithm would work correctly and the image would not be overly lightened. According to the OpenCV library documentation⁴, the `cv::Mat::convertTo` processes the values of each pixel according to (12).

$$g(i, j) = \alpha \cdot f(i, j) + \beta, \quad (12)$$

where $g(i, j)$ is the output pixel value, $f(i, j)$ is the input value, i and j are the pixel row and column numbers, α is the contrast ratio (from 1 to 3), and β brightness coefficient (from 0 to 100). It is necessary to calculate the α and β coefficients depending on the average L channel value (responsible for luminance) of the frame converted to the Lab color space. In daylight the L average channel value is approximately 130 (L value is in the interval from 0 to 255). Thus, let us define this value $L_i = 130 - \text{average daylight luminance}$. The frame highlighting need to be occurred with an average of $L < 100$. To slightly increase the color cast detection efficiency, we introduced a trial experimental dependence, which is calculated according to (13).

⁴https://docs.opencv.org/3.4/d3/d63/classcv_1_1Mat.html

$$\begin{aligned}
a_0 &= 33 \\
b_0 &= 0 \\
b &= b_0 + \frac{Li - L}{Li} \cdot 20 \\
a &= a_0 \cdot \left(1.0 + 1.3 \cdot \frac{Li - L}{Li}\right) \\
\alpha &= 3.0 \cdot a \cdot 0.01 \\
\beta &= b \\
K &= 2.0 - \frac{Li - L}{2Li}
\end{aligned} \tag{13}$$

Under the conditions of 30-35% illumination, the color cast is detected by overly illuminating the frame with graininess side increase. However, in the conditions of a very low illumination level (about 20% of Li), even a boost in brightness and contrast does not help to significantly increase the sensitivity of the detection algorithm. K factor threshold value needs to be decreased in a leap, what is a doubtful measure, as we have no information on this algorithm's applicability limits. Such measure might result in more false positive errors in various conditions.

5.2. Blur

Blur detection showed satisfying performance in daylight. However, in low light conditions most of objects' edges fade, what leads to false detections, as can be seen from the testing video¹ on 2:08. In addition, false detections occur when applying the algorithm to a lens overlapped and uncorrected color cast frames (from 2:25 on the video¹).

5.3. Dirt detection

It should be noticed that this fault cannot be corrected without cleaning the lens or disassembling the camera, so the algorithm is focused only on unwanted objects and spots detection. For UAV's correct operation, it is vital to know if camera has an incomplete view in order to prevent accidents. Our demo-setup showed that dirt detection algorithm performs well even in dim luminance. In low light conditions, a false detection can occur - algorithm marks all dark parts in the frame as unwanted objects (dirt), as one can see from 3:00 on the video¹.

6. Conclusion

In this paper, we proposed and developed a custom software to process and correct video stream's quality from the UAV's on-board camera in real-time conditions. Initially, we analyzed and briefly described the existing approaches to detecting and correcting faults. Then, we implemented these approaches in the developed custom software as a frame-by-frame video stream processing algorithm, and conducted several experimental demo-setups to assess its effectiveness. As the results showed, the algorithm performs well in daylight conditions, manually introduced video stream faults were detected, processed, and corrected. However, in

low light conditions some faults were detected improperly due to the lack of accuracy. At this stage, the software and algorithms require improvement and revision for low light conditions depending on relations between faults, which are a future prospects for this research, as well as the implementation and testing of the proposed approach on real UAV's physical model.

References

- [1] O. Ganin, I. Ganin, "Smart City": development perspectives and tendencies, *Ars Administrandi* (2014) 124–135.
- [2] M. Itoh, Y. Li, T. Saeki, Y. Suda, Surveillance camera system having camera malfunction detection function to detect types of failure via block and entire image processing, 2015. US Patent 8,964,030.
- [3] L. Dong, Y. Zhang, C. Wen, H. Wu, Camera anomaly detection based on morphological analysis and deep learning, in: 2016 IEEE International Conference on Digital Signal Processing (DSP), IEEE, 2016, pp. 266–270.
- [4] Y. Niu, X. Zheng, T. Zhao, J. Chen, Visually consistent color correction for stereoscopic images and videos, *IEEE Transactions on Circuits and Systems for Video Technology* 30 (2019) 697–710.
- [5] F. Li, H. Jin, An approach of detecting image color cast based on image semantic, in: *Proceedings of 2004 International Conference on Machine Learning and Cybernetics* (IEEE Cat. No. 04EX826), volume 6, IEEE, 2004, pp. 3932–3936.
- [6] F. Li, J. Wu, Y. Wang, Y. Zhao, X. Zhang, A color cast detection algorithm of robust performance, in: 2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI), IEEE, 2012, pp. 662–664.
- [7] G. Han, X. Li, Z. Lin, S. You, X. Cheng, Lab-space-based detection method based on image color cast, 2016.
- [8] F. Gasparini, R. Schettini, Color correction for digital photographs, in: 12th International Conference on Image Analysis and Processing, 2003. *Proceedings.*, IEEE, 2003, pp. 646–651.
- [9] R. Bansal, G. Raj, T. Choudhury, Blur image detection using laplacian operator and open-cv, in: 2016 International Conference System Modeling & Advancement in Research Trends (SMART), IEEE, 2016, pp. 63–67.