Jakub Kochan
jakub.kochan@wat.edu.pl
Military University of Technology in Warsaw

Mariusz Ważny
mariusz.wazny@wat.edu.pl
Military University of Technology in Warsaw

Krzysztof Falkowski
krzysztof.falkowski@wat.edu.pl
Military University of Technology in Warsaw

THE CONCEPT OF DETERMINING GROUND SPEED WITH LIMITED ACCESS TO GNSS SIGNAL ON UAV

DOI: 10.55676/asi.v5i1.60

Abstract
This article presents an innovative approach to navigation using image recognition in situations characterized by limited access to GNSS (Global Navigation Satellite System) signals (signal interference). The presented system relies on image processing to define the characteristic edges of random objects. Subsequently, the actual ground speed of the moving object (UAV) is obtained based on changes in the object's position. The article aims to show the potential of image recognition in navigational systems. The actual ground speed obtained by the image recognition can be used to correct the inertial navigation system.
Keywords*: Ground speed, UAV, Image Recognition, Navigation, Object detection, GNSS, GPS*

KONCEPCJA UKŁADU WYZNACZANIA PRĘDKOŚCI WZGLĘDEM ZIEMI Z OGRANICZONYM DOSTĘPEM DO SYGNAŁU GNSS NA PLATFORMY BEZZAŁOGOWE

Streszczenie
Niniejszy artykuł prezentuje nowatorskie podejście do nawigacji z wykorzystaniem rozpoznania obrazu w sytuacjach charakteryzujących się ograniczonym dostępem do sygnału GNSS (zakłócenia sygnału). Zaprezentowany system opiera się na przetworzeniu obrazu w celu zdefiniowania charakterystycznych krawędzi obiektu. Następnie na podstawie zmian położenia obiektu wyznaczana jest prędkość rzeczywista poruszającego się BSP. Ponadto w artykule przedstawiono potencjał zastosowania rozpoznania obrazu w systemach nawigacyjnych. Wyznaczona prędkość z użyciem rozpoznania obrazu może być wykorzystana do korekcji inercjalnych systemów nawigacji.
Słowa kluczowe*: Prędkość względem ziemi, BSP, rozpoznanie obrazu, nawigacja, wykrywanie obiektów, GPS, GNSS*

1. Introduction

The task of navigation systems is to answer two questions. The first question is "Where am I?" and the second is "In which direction do I need to move to reach my destination?" These questions give rise to very specific tasks for navigation systems. Therefore, one of the fundamental tasks of navigation systems is to determine the current position.

The most popular type of navigation systems widely used are solutions that utilize satellite signals, known as GNSS (Global Navigation Satellite System). The miniaturization of receivers and their low cost have led to the widespread adoption of these systems. At the same time, affordable and compact satellite signal receivers have become the primary system supporting the navigation of small unmanned aerial vehicles (UAVs weighing less than 5 kg). Satellite navigation systems are integrated systems dedicated to Earth navigation. They consist of a space segment (satellites) and a ground segment (receivers).

The major drawback of such systems is the susceptibility to signal interference. Factors such as urban infrastructure, natural obstacles, or solar storms can lead to a loss of measurement accuracy. In military applications, these systems are vulnerable to artificial interference (jamming). Cruise missiles or unmanned aerial vehicles, due to interference with GNSS receivers, may not execute their planned missions correctly. Small unmanned aerial vehicles are particularly susceptible to these limitations because they cannot accommodate high-end navigation systems on board, as is done with manned aircraft or large unmanned aerial vehicles (which use high-end INS systems or employ astronavigation).

Due to the widespread use of small unmanned aerial vehicles in military operations, improving their functionality in situations with limited or lost satellite signal access becomes a critical issue. Therefore, alternative methods for navigation are being sought. Visual systems are becoming an increasingly popular solution for navigation systems, especially in situations with poor GPS signal access. Such systems solve the navigation problem in indoor spaces or during conflict. The systems can use different technologies, from AruCo markers[1] to the use of artificial neural networks and deep learning[2]. Such systems aim to determine the speed, drift[3], or location of the UAV within specified markers. The image source can come from a camera mounted on the UAV or other sources such as satellites[4]. This article will present a proposal for creating a counting navigation system based on image recognition by the camera mounted on a UAV.

The concept of the system aims to provide support for traditional navigation systems in situations where there is limited access to GNSS signals. The main idea is to utilize image recognition to determine the speed of the moving object. Image recognition is an evolving computer technique that involves processing an image to extract information about objects within the image, which can be assigned to the appropriate class and their positions localized. To determine the speed of the moving object, it is necessary to process multiple frames of video and track the object in each frame captured by the camera. Detecting objects for tracking is the first task of the system. It must locate and mark the edges of the object's position. An essential part of the image recognition task is finding objects in successive frames, which should enable the calculation of displacement. Knowing the frame capture rate of the video makes it possible to determine how fast the multi-rotor flying platform is moving. This type of system can be classified as a map-less system. A system like this doesn't have information about the actual map. It performs navigation tasks without this information[5].

In summary, the proposed counting navigation system offers a viable alternative in situations where access to GNSS signals is limited. By integrating image recognition technology to measure the speed of the multi-rotor flying platform, this system aims to enhance the functionality of small UAVs, particularly in military operations where reliable navigation is paramount.

2. Edge approach to image recognition for speed determination

2.1. Canny Detector

The primary source of information for the system is the camera mounted on the unmanned aerial vehicle (UAV). Based on the information from this camera, the UAV's speed of movement will be determined. Image frames need to be processed to determine the speed. The camera on the UAV is vertically mounted downward, capturing an aerial view. This orientation allows for the determination of the Field of View (FOV), which is the area covered by the camera's image. It has specific length and width dimensions. Knowledge of the FOV size is necessary for scaling distances between pixels to actual distances.

The first task of the system is to identify characteristic elements within the image. The Canny edge detection method is an approach developed by John F. Canny in 1986[6] . Its primary objective is to detect edges in

[1] F. Wang, Y. Zou, C. Zhang, J. Buzzatto, M. Liarokapis, E. del Rey Castillo, J.B. Lim, *UAV navigation in large-scale GPS-denied bridge environments using fiducial marker-corrected stereo visual-inertial localization*, Automation in Construction vol. 156, December 2023.
[2] Y. Chang, Y. Cheng, U. Manzoor, J. Murray, *A review of UAV autonomous navigation in GPS-denied environments*, Robotics and Autonomous Systems 170, December 2023, pp. 4–5.
[3] P. Chmielewski, K. Sibilski, *Ground Speed Optical Estimator for Miniature UAV*, Sensors 2021, 21(8):2754, https://doi.org/10.3390/s21082754.
[4] P. Doherty, G. Conte, *An Integrated UAV Navigation System Based on Aerial Image Matching*, Big Sky, MT, USA, 2008, pp. 1–10, DOI: 10.1109/AERO.2008.4526556.
[5] Y. Lu, Z. Xue, G.S. Xia, L. Zhang, *A survey on vision-based UAV navigation*, Geo-Spat. Inf. Sci. 2018, pp. 21–32.
[6] J. Canny, *A Computational Approach To Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986, pp. 679–698.

images by searching for abrupt changes in pixel intensities. The edge detection algorithm consists of several steps.

The first step is noise reduction, which eliminates unnecessary details and smoothing the image. This is achieved using Gaussian filters.

The next step involves calculating the intensity gradient of the image, which determines the direction and strength of pixel intensity changes. It is a crucial element in detecting areas with abrupt changes in pixel intensity. This step uses the Sobel filter, which enables the approximation of directional derivatives of image intensity. The approximation is done both horizontally (G_x) and vertically (G_y). The resulting images are used to determine the gradient and edge direction using the following formulas[7].

$$G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = tan^{-1}\left(\frac{G_y}{G_x}\right)$$

The next step aims to determine edge continuity. For each pixel, the gradient value and edge direction are analyzed. This process searches for local maxima, while less significant pixels are suppressed. The result is a continuous line composed of individual pixels.

The final step in the Canny algorithm is thresholding with hysteresis. This stage identifies insignificant edges that have a gradient below a selected threshold. Two thresholds are determined: a minimum and a maximum. Pixels with a gradient above the maximum threshold are considered strong edges. Gradient values of pixels below the minimum threshold are suppressed. This way, an edge terminates if the gradient value of the last pixel falls below the minimum threshold. Strong edges are connected to weaker edges. The described method allows for obtaining the edges of objects in the image[8].

In Figure 1, an example of using the Canny detector is presented. The original image is shown on the left side of the figure. Then, the detector is applied, resulting in a black image with white contours marked. Finally, the contours are highlighted in blue, and the regions containing the contours are marked in green. The figure illustrates the difference between different threshold levels for the Canny detector. Manipulating threshold values can change the accuracy of detected edges. The images are black and white to increase the algorithm's processing speed.

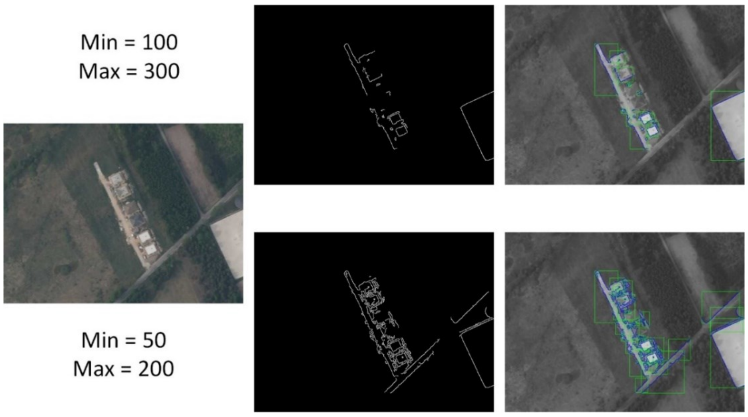

Fig. 1. Example of using the Canny Detector for Different Thresholds
Source: Author's work.

[7] S. Sofiane, *Canny Edge Detection Step by Step in Python – Computer Vision*, published in Towards data science, 25.01.2019.
[8] OpenCV documentation, *Canny Edge Detection*, https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html [accessed: 15.10.2023].

An important consideration is that higher edge detection accuracy is not always needed because it can lead to an excessive number of elements. Many of them may be too detailed and too small for further stages of the system's operation. This can also result in duplicating the same regions in the image.

2.2. Image Histogram

The determination of the displacement of detected edges in successive image frames is achieved using image histograms. A histogram is a graphical representation of the distribution of pixel intensities in an image. Histograms allow for organizing pixels into appropriate categories[9]. The Canny detector locates the edges of objects, which are then cropped from the image. For each cropped segment, its histogram is obtained. Histograms from the previous image frame are compared with histograms from the next image frame. This way, we identify the same objects in two different images. Figure 2 illustrates the process of comparing histograms. It compares individual image segments in search of the closest correlation between histograms from both images. The chart represents the histogram of an image segment. The horizontal axis shows pixel values, and the vertical axis represents the number of pixels for a given value. Once the same objects are found in two images, their positions become essential. The selected elements are referenced from the upper left corner in a Cartesian coordinate system with the Y-axis direction inverted. This allows for calculating the distance between two points.

$$\Delta_x = x_1 - x_2$$

$$\Delta_y = y_1 - y_2$$

This distance is calculated in pixels. Knowing the size of the Field of View (FOV) allows for scaling the difference into meters, representing the distance traveled by the object. To obtain the components of velocity, you also need to know the time, denoted as t_h, that has elapsed between the two images.

$$V_x = \Delta_x * t_h$$

$$V_y = \Delta_y * t_h$$

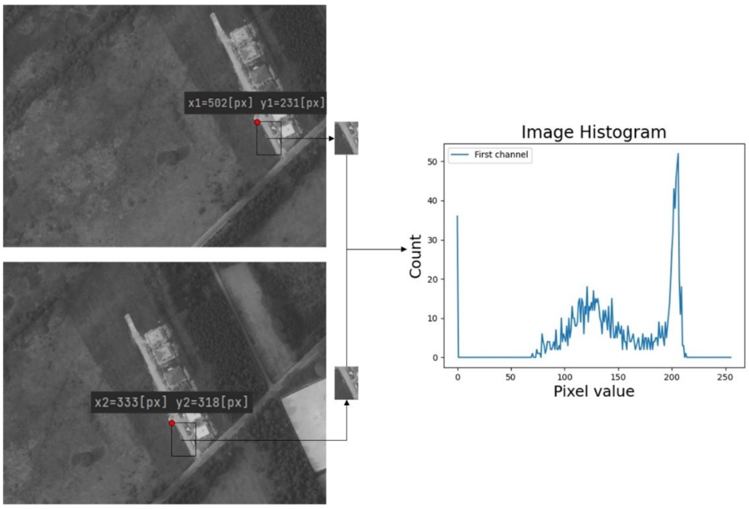

Fig. 2. The process of comparing histograms
Source: Author's work.

[9] S. Sasani, Perera, *OpenCV: Image Histogram Calculations*, https://medium.com/@sasasulakshi/opencv-image-histogram-calculations-4c5e736f85e, 14.7.2023 [accessed: 20.09.2023].

2.3. Detection algorithm

The navigation system begins its operation by orienting the camera perpendicular to the ground. This is a crucial step as it allows for the accurate determination of the Field of View (FOV). The moving aerial object changes its spatial position, so the camera should be equipped with an image stabilization system, whose task is to compensate for the camera's position relative to the UAV's position.

Next, the system initiates the phase using image recognition. This algorithm identifies the characteristic features of objects in the image. The detected areas are then extracted from the image for further processing. The next step involves calculating histograms. Histograms are used to determine similarities between areas. The result of the comparison is a percentage value indicating the similarity. Different areas are detected in two different images. A comparison result at a level of 95% confirms their identity. These areas are then tracked, and the difference in their positions in the *x* and *y* coordinates is determined.

The final element required by the system is the time elapsed between the camera's image frames.

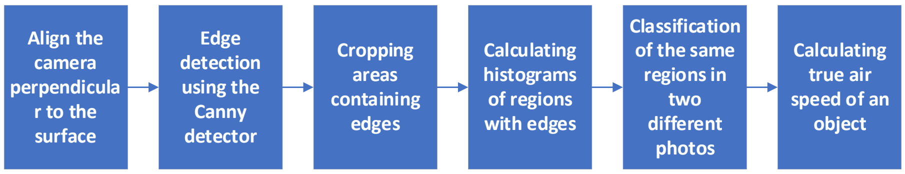
Align the camera perpendicula r to the surface
Edge detection using the Canny detector
Cropping areas containing edges
Calculating histograms of regions with edges
Classification of the same regions in two different photos
Calculating true air speed of an object

Fig. 3. System operation algorithm
Source: Author's work.

3. Corner approach to image recognition for speed determination

3.1. Shi-Tomasi algorithm

In the prior approach to the problem, edge detection was utilized for determining the image displacement. An alternative method could be the identification of distinctive points within the image. These distinctive points are locations where abrupt alterations in image parameters such as color, hue, or pixel intensity take place. A point is defined as a location where a minimum of two distinct edges intersect.

The Shi-Tomasi algorithm is a method for detecting distinctive points. It is an extended concept of the Harris algorithm, developed by Chris Harris and Mike Stephens. Its task is to search for object corners during image processing. The algorithm is primarily used for object recognition. In the project, it is used to locate distinctive points in the image, for example, object corners. The algorithm itself calculates the eigenvalue of points, considering the local change in intensity around each point. Object corners are those points whose eigenvalue exceeds a user-defined threshold[10].

The algorithm, similar to edge detection, begins its operation by calculating the image gradient. In this way, it determines the direction of pixel intensity changes. A small frame (u, v) moves across the image I at coordinates (x, y). The weighted sum between two adjacent frames is determined by S (Harris Corner Detection and Shi-Tomasi Corner Detection | by Nisha Gandhi | Pixel-wise | Medium).

$$S(x,y) = \sum_u \sum_v w(u,v)\,(I_x(u,v)x + I_y(u,v)y)^2$$

$$S(x,y) = [x \quad y] A \begin{bmatrix} x \\ y \end{bmatrix}$$

$$M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

[10] Haydar Abdulameer Kadhim, *Waleed Abdullah Araheemah, A method to improve corner detectors (Harris, Shi-Tomasi & FAST) using adaptive contrast enhancement filter*, March 2020, pp. 508–511.

Where:
$w(u, v)$ – type of window that slides over the image,
I_x, I_y – image derivatives in the X and Y directions,
M – structure tensor.

After obtaining frames with a large change in pixel intensity, a selection is made. For this purpose, the eigenvalue of the structural tensor M is calculated. The M matrix should have two large eigenvalues for the point to be classified as a corner. Harris proposed the following solution.

$$R = \det(M) - k\big(trace(M)\big)^2$$

$$\det(M) = \lambda_1 \lambda_2$$

$$trace(M) = \lambda_1 + \lambda_2$$

Where:
λ_1, λ_2 – eigenvalue of structural tensor M,
k – empirical value between 0,04–0,15.

The Shi-Tomasi algorithm modified the score formula. It applied the following function[11].

$$R = \min(\lambda_1, \lambda_2)$$

Figure 4 illustrates the relationship between the eigenvalues λ_1-λ_2. The $\lambda_{\min}$ values determine the boundary values for the Shi-Tomasi algorithm. The green region of the graph specifies when the frame will be classified as a corner. This happens when the eigenvalues exceed the threshold. The orange region specifies when the frame will be counted as a line. The grey region defines the frame as a flat region.

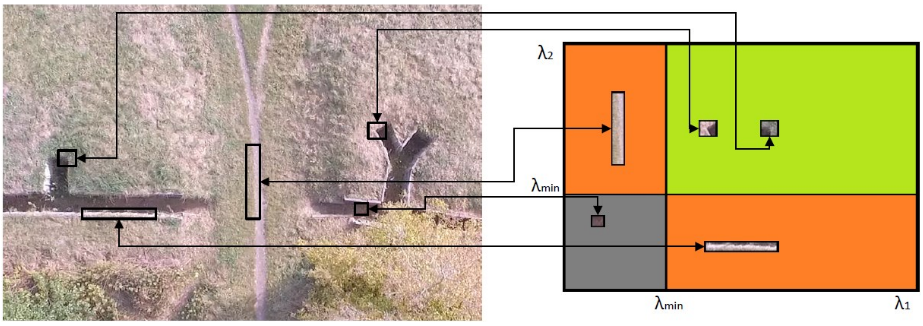

Fig. 4. Eigenvalue dependency graph of the Shi-Tomasi algorithm
Source: Author's work.

OpenCV is a library of functions designed for image processing. It includes a function that utilizes the Shi-Tomasi algorithm. This function is called goodFeaturesToTrack. To use it, the following parameters need to be defined:

- InputArray – Input image (in grayscale),
- maxCorners – The maximum number of corners to return (in the case of a larger number of detected corners, the algorithm returns the strongest corners),
- qualityLevel – Specifies the minimum quality of corners (value between 0–1),
- minDistance – The minimum Euclidean distance between detected corners[12].

[11] Nisha Gandhi, *Harris Corner Detection and Shi-Tomasi Corner Detection*, https://medium.com/pixel-wise/detect-those-corners-aba0f034078b, 24.7.2018 [accessed: 15.10.2023].
[12] OpenCV documentation, *Shi-Tomasi Corner Detector & Good Features to Track*, https://docs.opencv.org/3.4/d4/d8c/tutorial_py_shi_tomasi.html [accessed: 18.10.2023].

The function itself returns the strongest corners defined by the maxCorners parameter, which exceed the quality specified by the qualityLevel parameter and are separated from each other by the value of the minDistance parameter. Figure 5 shows a basic photo prepared for image processing. The photo was taken from an unmanned aerial vehicle DJI Phantom 3 Professional. This UAV has a Sony Exmor camera mounted on it. The camera has 94° FOV. The video was recorded in FullHD resolution with 30 fps. The camera was pointed perpendicular to the ground. Figures 6 and 7 illustrate the use of the Shi-Tomasi algorithm. The characteristic points defined by the algorithm are marked with red dots. Figure 6 shows an example of using the Shi-Tomasi algorithm. Each case in Figure 7 differed only by the change in the maxCorners parameter. The qualityLevel and maxDistance parameters were constant and were 0.1 and 5, respectively. MaxCorners were 20, 50, and 100 from the top of the image. The photo placed at the very bottom has 100 detected points, which means it has reached its limit. It can be observed that with a higher value of the maxCorners parameter, clusters of points are formed. These can be excluded by increasing the maxDistance parameter as it is shown in Figure 6. Utilizing this will increase the distance between the detected corners.

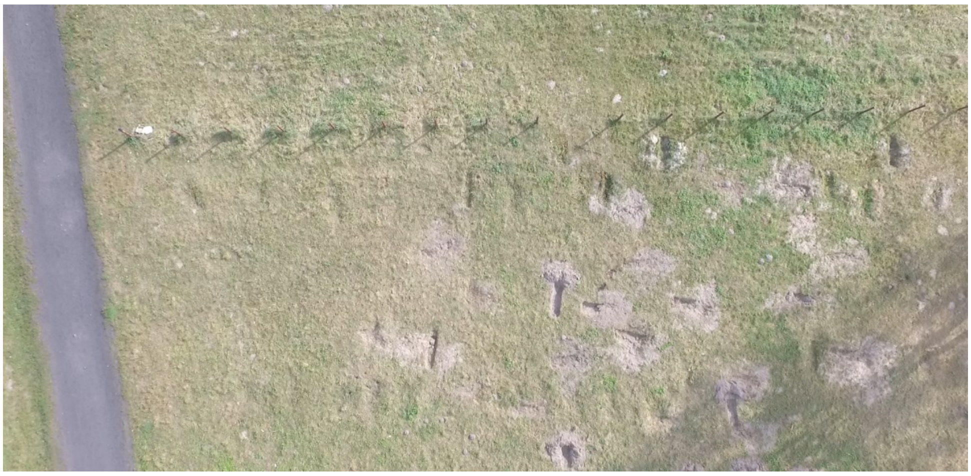
Fig. 5. Basic photo prepared for image processing
Source: Author's work.

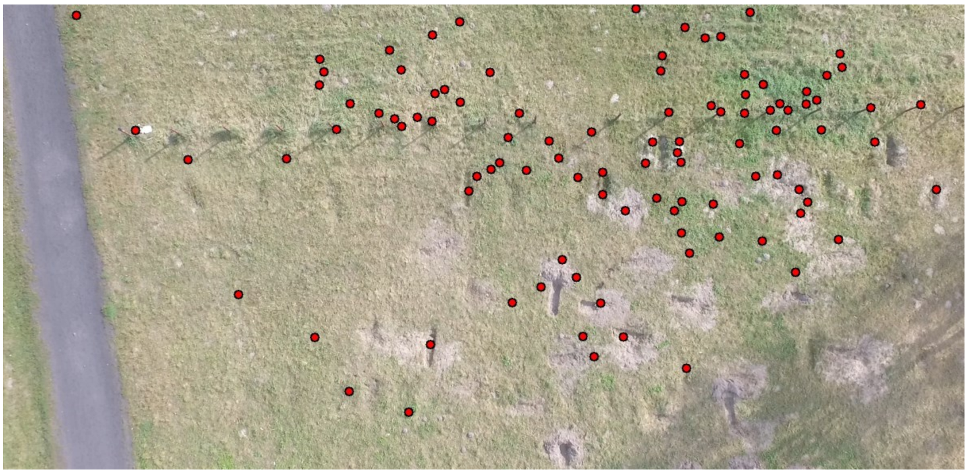
Fig. 6. Example of using Shi-Tomasi algorithm. Parameters maxCorners = 100, qualityLevel = 0,2, maxDistance = 20
Source: Author's work.

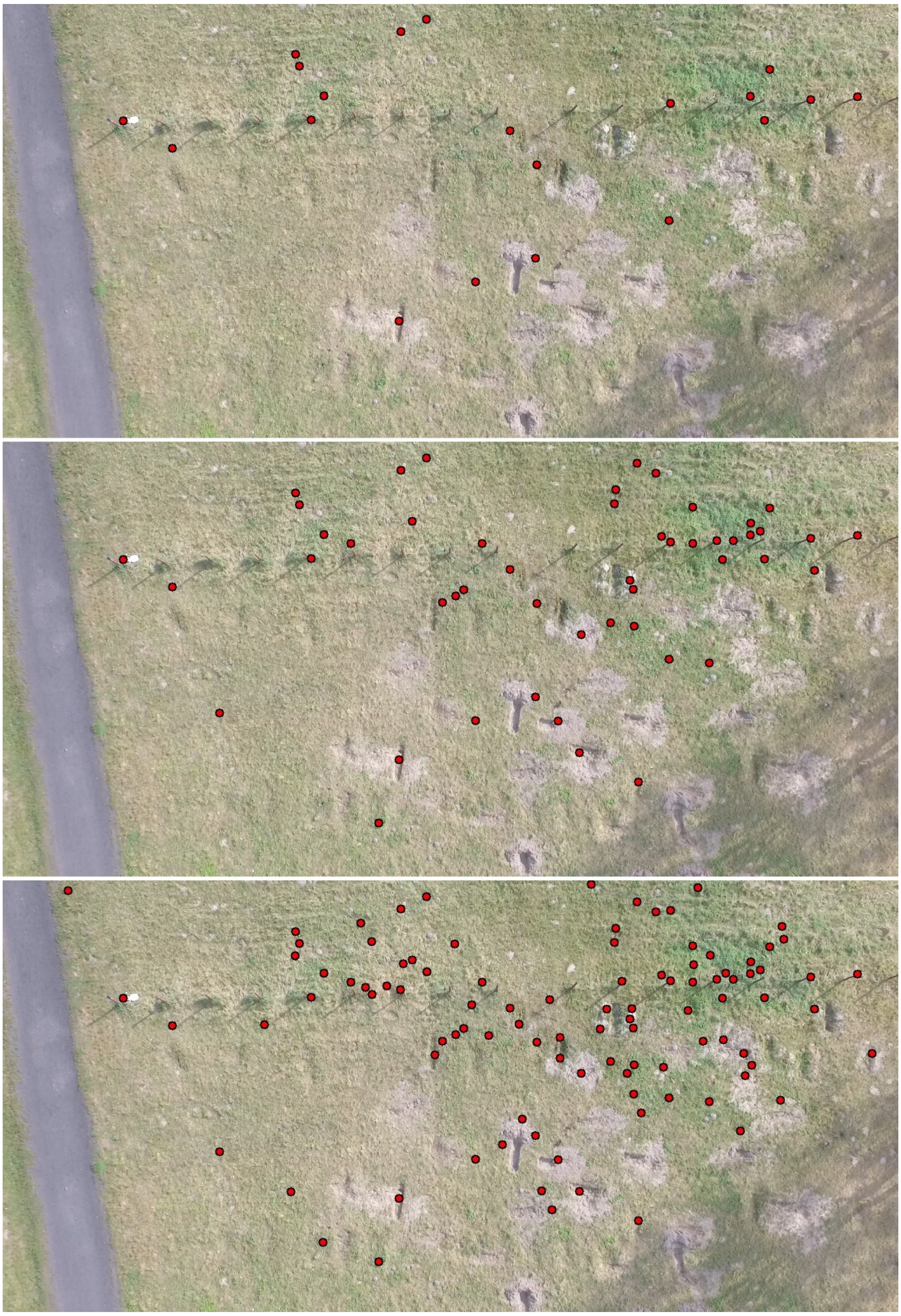

Fig. 7. Results of using the Shi-Tomasi algorithm for different values of the maxCorners parameter – 20, 50, 100
Source: Author's own work.

3.2. LK algorithm

The KLT (Kanade-Lucas-Tomasi) algorithm is used for visual motion analysis. Its main task is to track distinctive points in the image. In this way, it is possible to track the motion of objects in successive frames of the image. The determination of distinctive points (image features) is done using the Shi-Tomasi algorithm. It finds satisfying corners. After determining the features, their tracking in subsequent image frames begins. For each subsequent corner, the displacement is calculated using the Lucas-Kanade method. The algorithm uses the optical flow equation to determine the velocity vector (V_x, V_y).

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i)I_y(q_i) \\ \sum_i I_y(q_i)I_x(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i)I_t(q_i) \\ -\sum_i I_y(q_i)I_t(q_i) \end{bmatrix}$$

Where:
V_x, V_y – local image flow velocity vector in X and Y direction,
I_x, I_y – partial derivatives of the image I in X and Y direction,
q_i – pixels inside the window.

The position of individual points is updated using the results obtained through the Lucas-Kanade method[13].

$$(x + y) \leftarrow (x + y) + (\Delta_x + \Delta_y)$$

After updating all the corners, an update of the points can be applied. In the case where a point disappears from the image, it may be replaced with another point. For this purpose, the Shi-Tomasi algorithm is reused to search for characteristic points. Finally, the entire process is repeated for each new image frame. The OpenCV library has a function that uses the described algorithm. It is calcOpticalFlowPyrLK. It is used to track feature points in video. This function takes the following parameters:

- prevImg – previous video frame,
- nextImg – next video frame,
- prevPts – vector of 2D points for which the flow needs to be found,
- winSize – the size of the window to searching.

The important thing is that prevImg and nextImg need to be the same size and type. Points in prevPts must be single-precision floating-point numbers[14]. Figure 8 shows an example of using the LK algorithm. The script was written in Python language. The OpenCV, numpy, and time libraries were used in this script. The figure is a similar photo to Figure 5. It shows the terrain. There are red dots that show selected points to track. Green lines show move of the point. It shows a trace of that point made. On some begins of lines are red dots. If there is a red dot at the beginning of the line, it means that this point is being tracked. If there is only a green line, it means that the selected point was not found and this point is rejected. Additionally, a blue arrow has been added to the program. It is visible on the left side of the image. It symbolizes the direction of movement of the unmanned aerial vehicle. The speed value is scalable to the length of the arrow. The speed value is calculated only with the change of moving tracked points. When using the program, it is important to determine how often points are to be added. This time should be short enough that at least half of the detected points are always visible.

[13] D.L. Bruce, Takeo Kanade, *An Iterative Image Registration Technique with an Application to Stereo Vision*, 1981, pp. 121–130.
[14] OpenCV documentation, *Optical flow*, https://docs.opencv.org/4.x/d4/dee/tutorial_optical_flow.html [accessed: 12.10.2023].

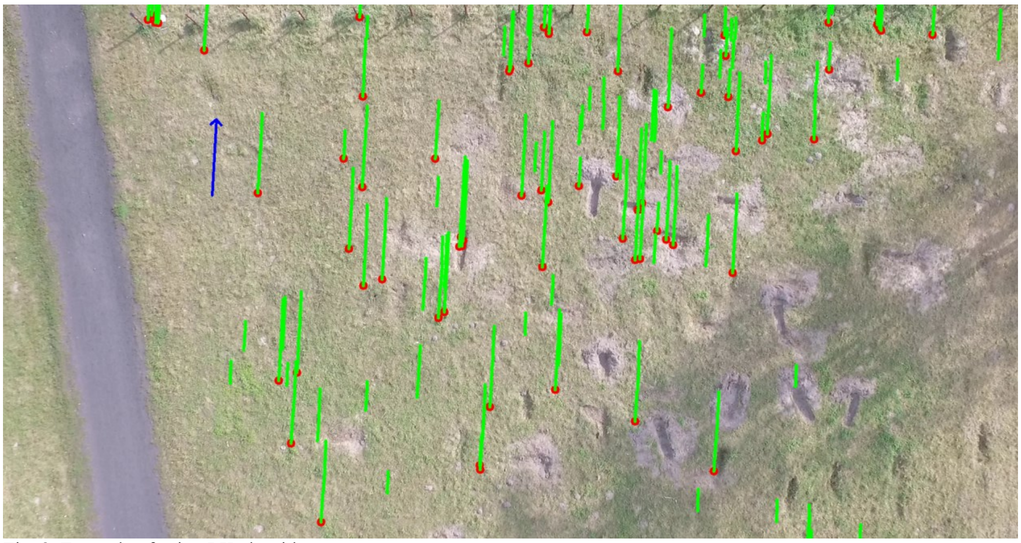
Fig. 8. Example of using LK algorithm
Source: Author's own work.

4. Conclusion

The proposed system envisions utilizing images to determine the actual speed of an aerial vehicle. This is an innovative approach to addressing the navigation challenges faced by Unmanned Aerial Vehicles (UAVs). The article presents the potential of using image recognition in navigation systems. This concept can be an auxiliary navigation system that corrects other system's errors.

In such systems, the primary sensor is a camera mounted on the UAV. This system's application is particularly valuable in situations where satellite signals are jammed or unavailable. Such a system can also complement Inertial Navigation Systems (INS). In this case, it can serve as a corrective tool. INS systems accumulate errors over time, which increase over time. GNSS systems introduce corrections that mitigate this error. In situations with limited GNSS signals, the proposed system could effectively serve as an alternative.

The primary application of the system is for military purposes. Armed conflicts often begin with disinformation tactics, including the jamming of satellite signals. In such scenarios, Unmanned Aerial Vehicles (UAVs) become highly valuable assets, and their precise mission execution is essential. The system ensures that UAVs can operate effectively even in environments where GNSS signals are jammed or in indoor settings where satellite signals cannot be received. Additionally, it is designed to perform reliably in urban areas with dense infrastructure and buildings, making it a versatile solution for various challenging scenarios. The system can also be mounted on commercial drones. The only requirement is to mount the camera as a sensor.

One of the system's strengths is its simplicity. It consists of only a camera and a computational unit. The computational unit doesn't need to be placed on the UAV itself; calculations can be performed on the ground station, where only the image needs to be transmitted. The primary direction for future development is aimed at improving the accuracy and stability of edge detection algorithms, which is a crucial factor. Additionally, the system can be equipped with other types of auxiliary sensors. A laser altimeter could also indicate the real height of the surface. It can help to make the algorithm more precise to know the height from the barometer and laser altimeter. The limitation of this type of system is a camera. If vision is blocked, then the system will not work correctly. Thermal and noctovision cameras, for instance, would enable the system to operate in conditions with limited access to visible light like fog or smoke on the battlefield.

In this paper, two approaches for this subject – edge and corner. The edge approach uses the Canny algorithm to detect an object`s edges and image histograms to compare edges on different video frames. The corner approach uses the Shi-Tomasi algorithm to detect corners and the LK algorithm to search for the same points in different video frames. This article proves that the concept of a navigation system based on a camera is possible to complete.

This topic addresses a contemporary navigation problem. It is a modern approach to navigation that uses image recognition. Future research will focus on scaling the speed value from the altitude of an unmanned aerial vehicle. For this purpose, a study of the camera's field of view mounted on the UAV will be conducted. Then, an appropriate approach to the problem will be chosen. The study will concern the optimality of methods. The most important thing in this program will be the speed of the algorithm's operation so that it can work in real-time. An important element is also the optimization of algorithms for object detection. Algorithms should work optimally and not delay the program and examine the influence of camera parameters like quality, resolution, or frame rate.

References

Bruce D.L., Takeo K., *An Iterative Image Registration Technique with an Application to Stereo Vision*, 1981.

Canny J., *A Computational Approach To Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986.

Chang Y., Cheng Y., Manzoor U., Murray J., *A review of UAV autonomous navigation in GPS-denied environments*, Robotics and Autonomous Systems 170, December 2023, 4–5.

Chmielewski P., Sibilski K., *Ground Speed Optical Estimator for Miniature UAV*, Sensors 2021, 21(8):2754. https://doi.org/10.3390/s21082754.

Doherty P., Conte G., *An Integrated UAV Navigation System Based on Aerial Image Matching*, Big Sky, MT, USA, 2008, DOI: 10.1109/AERO.2008.4526556.

Haydar Abdulameer Kadhim, *Waleed Abdullah Araheemah, A method to improve corner detectors (Harris, Shi-Tomasi & FAST) using adaptive contrast enhancement filter*, pp. 508–511, March 2020.

Lu Y., Xue Z., Xia G.S., Zhang L., *A survey on vision-based UAV navigation*, Geo-Spat. Inf. Sci. 2018, 21–32.

Nisha Gandhi, *Harris Corner Detection and Shi-Tomasi Corner Detection*, https://medium.com/pixel-wise/detect-those-corners-aba0f034078b, 24.7.2018 [accessed: 15.10.2023].

OpenCV documentation, *Canny Edge Detection*, https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html [accessed: 15.10.2023].

OpenCV documentation, *Optical flow*, https://docs.opencv.org/4.x/d4/dee/tutorial_optical_flow.html [accessed: 12.10.2023].

OpenCV documentation, *Shi-Tomasi Corner Detector & Good Features to Track*, https://docs.opencv.org/3.4/d4/d8c/tutorial_py_shi_tomasi.html [accessed: 18.10.2023].

Sasani S., Perera, *OpenCV: Image Histogram Calculations*, https://medium.com/@sasasulakshi/opencv-image-histogram-calculations-4c5e736f85e, 14.7.2023 [accessed: 20.09.2023].

Sofiane S., *Canny Edge Detection Step by Step in Python – Computer Vision*, published in Towards data science, 25.1.2019.

Wang F., Zou Y., Zhang C., Buzzatto J., Liarokapis M., del Rey Castillo E., Lim J.B., *UAV navigation in large-scale GPS-denied bridge environments using fiducial marker-corrected stereo visual-inertial localization*, Automation in Construction 156, December 2023.