# Comparative analysis of visual navigation systems

**V M Sineglazov[1] and V S Ishchenko[1,2]**

[1] Department of Aviation Copmuter Intergated Complexes and Control Systems, National Aviation University, 1 Liubomyra Huzara ave., Kyiv 03058, Ukraine

[2] E-mail: ischenkoVitaly@gmail.com

**Abstract**. Nowadays, the requirements for UAV navigation have increased significantly; in addition, it is necessary to provide UAV navigation in the area of active radio interference, where signals from the satellite navigation system are blocked and only inertial system based on MEMS provide navigation [4]. To solve the navigation problem, additional sources of navigation information with a high degree of noise immunity are required. A visual navigation system (VNS) is proposed as an additional source of navigation information. The VNS includes: digital camera, additional navigation information calculator, gyro stabilized suspension, and autopilot information transfer interfaces for navigation and location correction due to refinement of landmarks. This article discusses the algorithmic software of visual navigation systems and compares the methods. The methods of the computer vision library (open CV) are considered in practice, their analysis and comparison of results for solving the problem of visual navigation are carried out. The main test of algorithms was carried out on frames of video recording of a flight with duration of more than 10 minutes. Analyzed the workability for autonomous flight in the absence of GPS signals. Considered the main requirements for local key points detectors and descriptors, performed measurement performance of different detectors and descriptors. During designing and developing software for the on-board (companion) computer, weight, size and computational restrictions are imposed. At the same time, the visual navigation system should provide highly accurate navigation close to the satellite navigation system. Tests for the computational power consumption of different algorithms have been carried out and the results are presented in tables and figures.

## 1. Introduction

During designing and developing software for the on-board (companion) computer, weight, size and computational restrictions are imposed. At the same time, the visual navigation system should provide highly accurate navigation close to the satellite navigation system. In most cases, computer vision methods (most often from the OpenCV library) are used as algorithmic support for detect and describe local features (key points) on image, match them using brute force or similar matcher, estimate homography by RANSAC algorithm, filter inliers from all the matches, calculate total shift UAV relative to ground landmarks. This article analyzes, compares and evaluates the accuracy of computer vision methods. A deep analysis of the applicability of computer vision methods for high-precision visual navigation of UAVs and error estimation has been carried out.

## 2. Literature review and problem statement

There is a video stream of frames from a camera on a gyro-stabilized gimbal, represented by a sequence of two-dimensional matrixes of pixels, due to the limited performance of the on-board computer, frames are resized to 520x380 pixels. To do this, it is necessary to select a detector descriptor and matcher of local features to achieve maximum accuracy and minimize computational load. The main goal is to achieve a stable search, description and determination of landmarks from frame to frame for a flight lasting at least 10 minutes in the area of active radio interference.

Review of existing solutions, also presented in official docs [1]

The following descriptors of computer vision are widely used:

- SIFT, Scale-invariant Feature Transform
- SURF, Speeded Up Robust Features
- FAST, Features from Accelerated Segment Test
- AKAZE, Accelerated KAZE
- ORB, Orientated BRIEF Rotated FAST
- BRIEF, Binary Robust Independent Features
- RANSAC Random Sample Consensus

As feature point matchers commonly used BF (Brute Force Matcher) and KNN (K – Nearest Neighbour), more information presented in official computer vision library docs [2] (different types of matching algorithms, also their accuracy and performance were compared in this topic).

## 3. The aim and objectives of research

Consider the stages of program execution. The video stream, were taken and saved from quadcopter DJI Phantom 3 pro, with 4k frames resolution which is too much for processing on the on-board computer. Video stream from drone camera processed by client server application in on-board computer and all details how it works presented by link [3]. That's why first step is resizing input video frame from 4k to 520x380 pixels and then converting to grayscale. Comparison of all existing computer vision local features detectors and descriptors with matching key point pairs were performed. All performance tests, has been performed on CPU Intel core i3 8145U 2.1GHz using 8GB ram memory under linux OS.

## 4. The method of tandem propeller hub losses reduction

For an illustrative example, used the most computationally laborious SIFT descriptor and BF matcher. There are input color frame presented in Figure 1.

**Figure 1.** Original colored input image.

Resizing and converting to grayscale are performed for faster calculations, then creating instance of descriptor SIFT to detect and describe local feature points. Repeat this procedure for next frame, after that use brute force matcher (BF matcher) for building points pairs and filter outliers (Figure 2).

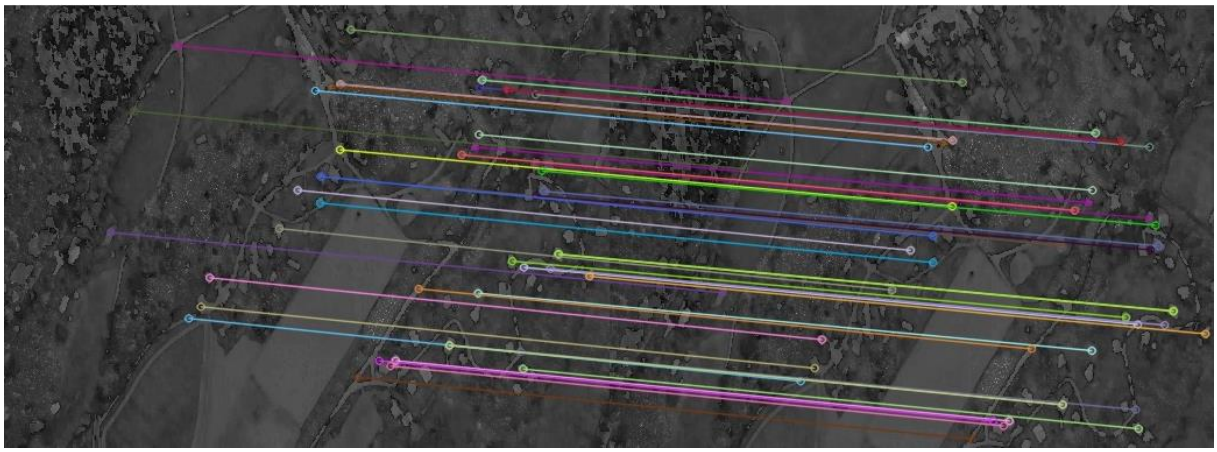

**Figure 2.** Building, matching feature points pairs using BF and SIFT.

Iteratively processing frame by frame live video stream from camera to provide visual navigation, calculation pixel shifts, conversion to meters and obtaining current GPS coordinates of UAV is performed. A local feature points detection, building key points descriptors and matching pairs for UAV position shift calculation between previous frame ad current frame is shown in Figures 3,4,5,6.

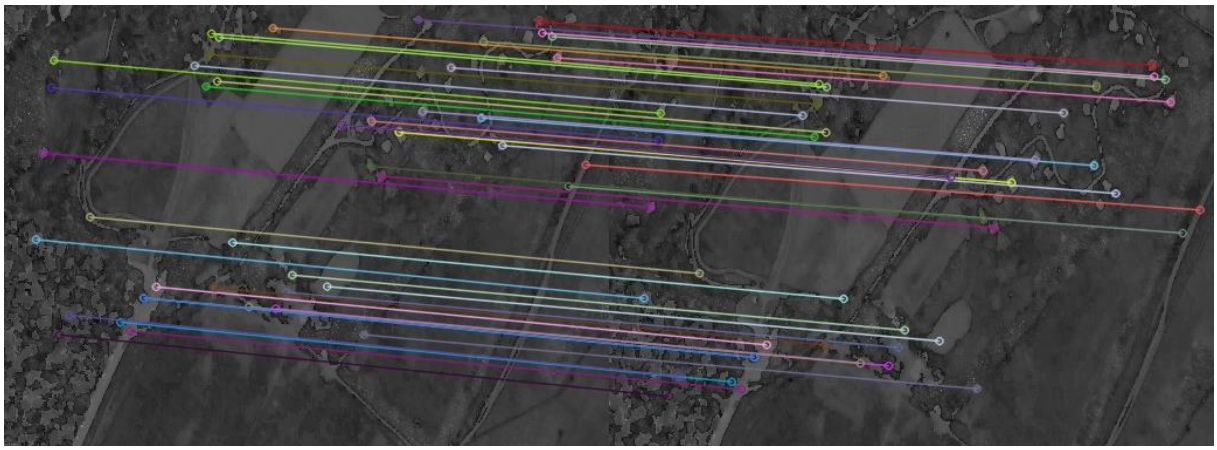

**Figure 3.** Local feature points detection, building key points descriptors and matching pairs between previous frame and current frame.

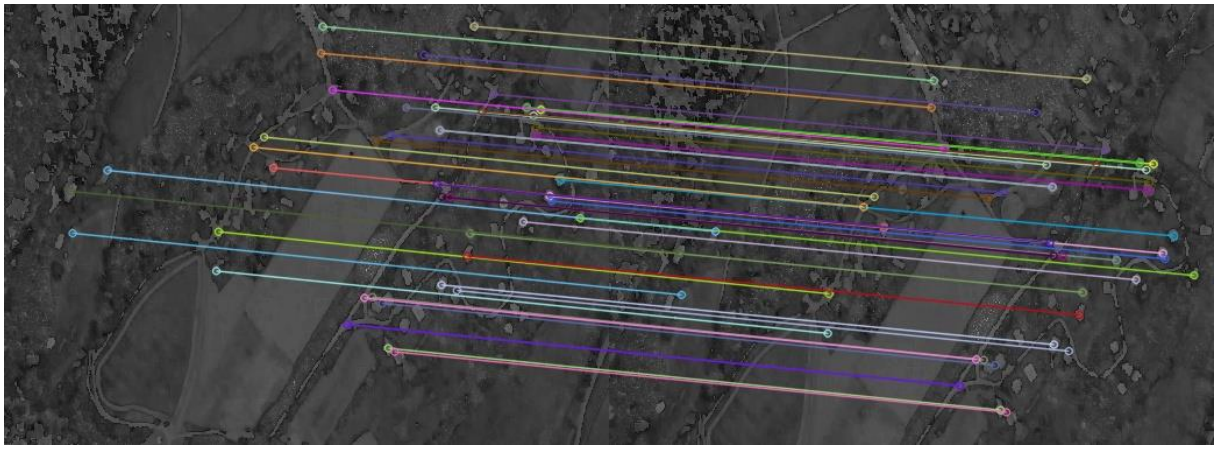

**Figure 4.** Local feature points detection, building key points descriptors and matching pairs between previous frame and current frame.

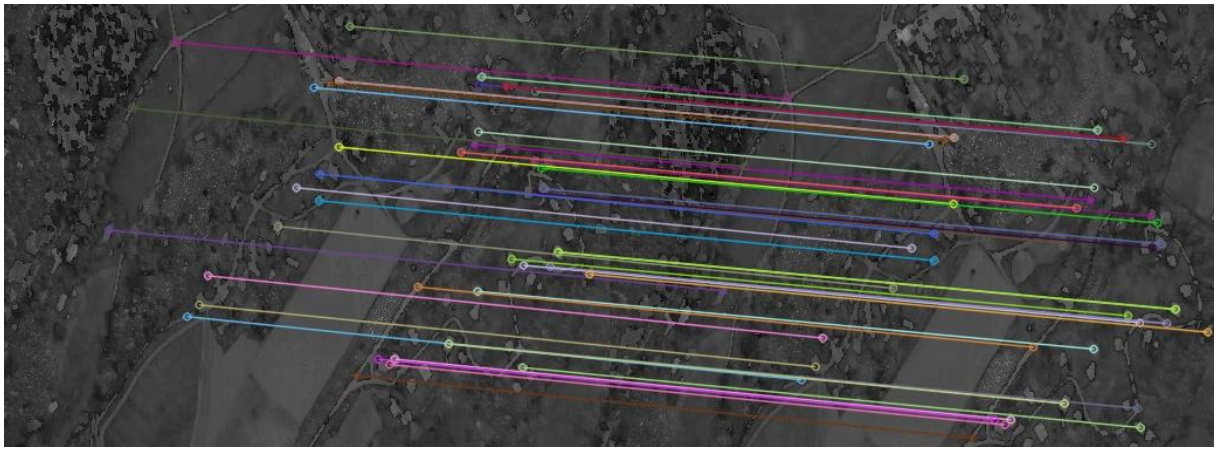

**Figure 5.** Local feature points detection, building key points descriptors and matching pairs between previous frame and current frame.

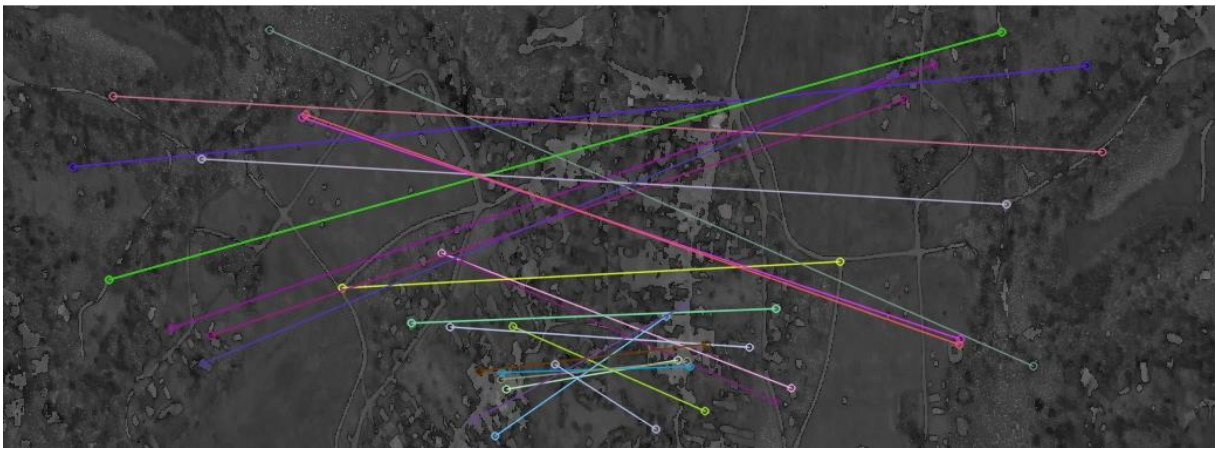

**Figure 6.** Local feature points detection, building key points descriptors and matching pairs between previous frame and current frame.

**6. Results of frames processing by different algorithms and frame sizes**

Pay attention to processing time of the frame (SIFT + BF) with a sizes of 520x380, 720x576 and classic 800x600 in seconds in table 1.

**Table 1.** Processing time of one frame in seconds.

| Frame size 520x380 | Frame size 720x576 | Frame size 800x600 |
|---|---|---|
| 0.126498 | 0.261223 | 0.366817 |
| 0.125311 | 0.259818 | 0.307732 |
| 0.124878 | 0.256252 | 0.325083 |
| 0.124943 | 0.257062 | 0.308956 |
| 0.126292 | 0.257318 | 0.310021 |
| 0.126852 | 0.256654 | 0.320367 |

Resizing frame from 4k to 520x420 takes near 0.01 second, this is amazing fast and processing frames with less size as shown in Table 1 gives multiple times increasing performance. Table 2, shows performance comparison between typical local feature points descriptors.

**Table 2.** Processing feature points descriptors in seconds.

| SIFT | SURF | ORB | AKAZE |
|---|---|---|---|
| 0.149122 | 0.0522411 | 0.0341239 | 0.0326113 |
| 0.165375 | 0.0499295 | 0.0298967 | 0.0297523 |
| 0.180101 | 0.0549136 | 0.0299566 | 0.0297916 |
| 0.176874 | 0.0487743 | 0.0298781 | 0.0296977 |

Results if performance matching using SURF + BRUTE FORCE MATCHER (BF) and SURF + KNN (K – Nearest Neighbors) presented in Table 3.

**Table 3.** Comparison of performance two matching algorithm.

| SURF + BRUTE FORCE matcher | SURF + KNN matcher |
|---|---|
| Times passed in seconds: 0.0640599 | Times passed in seconds: 0.0632252 |
| Times passed in seconds: 0.0553908 | Times passed in seconds: 0.055204 |
| Times passed in seconds: 0.064352 | Times passed in seconds: 0.0630322 |
| Times passed in seconds: 0.0571646 | Times passed in seconds: 0.0583795 |
| Times passed in seconds: 0.0541796 | Times passed in seconds: 0.0749743 |

Visual navigating system integrate with autopilot using UART interface and Mavlink protocol [5], more details about integrating external systems with autopilot shown in [6], [7].

**7. Discussion the research results**

The obtained results indicate the need to supplement the methods for calculating local features with alternative ones, for example, by calculating the optical flow to reset the accumulating error and improve the accuracy of visual navigation system. Additional approach to increase accuracy of visual

navigation system is implementing convolution neural network and switch to it in case of homogeneous terrain like flight over water, snow.

**Conclusions**

Computer vision methods based on local feature detection and description provide very low computational load on the on-board computer, however, they introduce a system error during sharp maneuvers of the UAV and must be compensated with additional software modules like neural networks, optic flow calculation to make high accurate visual navigation system. During experiments in this topic was used read autopilot Pixawh that integrated with our on-board computer – JetsonTX2 and provide autopilot correction using UART interface and MAVLINK protocol. More details about autopilot and the way of integration with external systems presented by link [8], and flight controller specification [9].

**References**


[1] OpenCV-Python Tutorials. Feature Detection and Description https://docs.opencv.org/3.4/db/d27/tutorial_py_table_of_contents_feature2d.html

[2] OpenCV Tutorials. 2D Features framework https://docs.opencv.org/3.4/d5/d6f/tutorial_feature_flann_matcher.html

[3] V G Olifer and NA Olifer 2004 Computer Networks, Principles, Technologies, Protocols (St. Petersburg) p 864

[4] E N Pyatnyshev, M Lurie, I Popov and A N Kazakin 2001 The specifics of MEMS technology devices pp 32-35

[5] Malink protocol used for communication autopilot and on-board computer https://mavlink.io/en/services/mission.html

[6] V A Rogozhin, V M Sineglazov and N K Filyashkin 2004 Aerobatic-navigation complexes of aircrafts (Ukraine: National Aviation University)

[7] V A Rogoshyn, V M Sineglazov and N K Filyashkin 2004 *Flight Navigation Complexes of Aircrafts* (Ukraine: National Aviation University) p 238

[8] Integration autopilot and on-board computer https://ardupilot.org/dev/docs/companion-computer-nvidia-tx2.html

[9] Flight controller Pixhawk http://pixhawk.org