Design of a High-Performance System for Secure Image Communication in the Internet of Things (Invited Paper)

Elias Kougianos*, Saraju P. Mohanty†, Gavin Coelho‡, Umar Albalawi§, and Prabha Sundaravadivel¶
NanoSystem Design Laboratory (NSDL, http://nsdl.cse.unt.edu) *†§¶
Department of Computer Science and Engineering †§¶
Department of Engineering Technology*
University of North Texas, Denton, TX 76207, USA. *†§¶
Peterbilt Motors Company, Denton, TX, USA. ‡
Email: elias.kougianos@unt.edu*, saraju.mohanty@unt.edu†, Gavin.Coelho@paccar.com‡,
UmarAlbalawi@my.unt.edu§, and prabhasundaravadivel@my.unt.edu¶.

***Abstract*—Image or video exchanges over the Internet of Things (IoT) is a requirement in diverse applications including smart health care, smart structures, and smart transportations. This paper presents a modular and extensible quadrotor architecture and its specific prototyping for automatic tracking applications. The architecture is extensible and based on off-the-shelf components for easy system prototyping. A target tracking and acquisition application is presented in detail to demonstrate the power and flexibility of the proposed design. Complete design details of the platform are also presented. The designed module implements basic PID control and a custom target acquisition algorithm. Details of the sliding-window based algorithm are also presented. This algorithm performs 20× faster than comparable approaches in OpenCV with equal accuracy. Additional modules can be integrated for more complex applications, such as search-and-rescue, automatic object tracking, and traffic congestion analysis.**

A hardware architecture for the newly introduced Better Portable Graphics (BPG) compression algorithm is also introduced in the framework of the extensible quadrotor architecture. Since its introduction in 1987, the Joint Photographic Experts Group (JPEG) graphics format has been the *de facto* choice for image compression. However, the new compression technique BPG outperforms JPEG in terms of compression quality and size of the compressed file. The objective is to present a hardware architecture for enhanced real time compression of the image. Finally, a prototyping platform of a hardware architecture for a Secure Digital Camera (SDC) integrated with the Secure Better Portable Graphics (SBPG) compression algorithm is presented. The proposed architecture is suitable for high performance imaging in the IoT and is prototyped in Simulink®. To the best of the authors' knowledge, this is the first ever proposed hardware architecture for SBPG compression integrated with an SDC.

***Index Terms*—Internet of Things (IoT), Object Detection, Robot Vision Systems, Secure Digital Camera (SDC), Better Portable Graphics (BPG), Image Compression, VLSI Architecture**

I. Introduction

In the last few decades personal computers (PCs) have had an enormous impact in society. PCs are omnipresent in various types and form factors such as desktops, laptops, tablets, smart-phones, and other application specific systems. It is envisioned that in the next few decades the personal robot will have the same, if not higher, impact on society [1]. Automation is currently widely used in many industries, performing a multitude of tasks. Robots, a mode of automation, are generally quicker, more productive and extremely accurate, and perform many of the tasks that are either too dangerous or undesirable for humans to do. Their applications cover a wide range from manufacturing, cleaning and maintenance, to bomb disposal. They are currently very specialized, only being able to perform a few predetermined tasks and are mostly used in industrial and military applications. However, they are starting to be used more for civil applications. Personal robots like the ones found in many futuristic novels and movies are intelligent, are able to perform a set of tasks, to make decisions on the fly allowing them to adapt to and interact with their surroundings. In order for this to become a reality, robots need to be not only smarter but must be enabled to be more aware of their surroundings. This is not far off and Willow Garage's PR2 [2] is a good example of the current state of personal robots.

Quadrotors are commonly used with an onboard camera and one or two operators. The operators control both the flight of the vehicle and the camera operation. In industrial applications this provides a quick, easy and relatively low-cost means for inspection of pipelines, bridges and large structures and navigating areas that are remote and otherwise hard to access. Civil applications include search and rescue, traffic congestion analysis, fire monitoring, HAZMAT operations and the inspection of dangerous sites as well as environmental assessments and nature conservation. In law enforcement they are useful for surveillance, documenting crime scenes and gathering intelligence. They can also be used for aerial photography, television and videography, real estate and property assessment. By enabling autonomous control with object recognition and video tracking many of these tasks can be automated allowing for more vehicles to be deployed with considerably fewer operators. For example, during a search and rescue mission, multiple quadrotors could be programmed to search a given area sending an alert to the search team

when a possible subject is found. If they are equipped with an infrared thermal imaging camera this would allow them to search through the night. Similarly, for law enforcement and surveillance a subject could be tracked by multiple quadrotors, all communicating with the base station or each other, forming a subnet of the Internet-of-Things (IoT), as shown in Fig. 1. These systems is expected to have significant applications in smart cities [3], [4]. The control algorithm can then analyze information from not just one vehicle but the whole swarm and the IoT itself allowing it to make more complex calculated decisions. A pictorial view of a search-and-rescue application is shown in Fig. 2. The quadrotor(s) continuously process information from external sources (such as GPS) and the IoT.

Fig. 1. The IoT-enabled aerial platform.

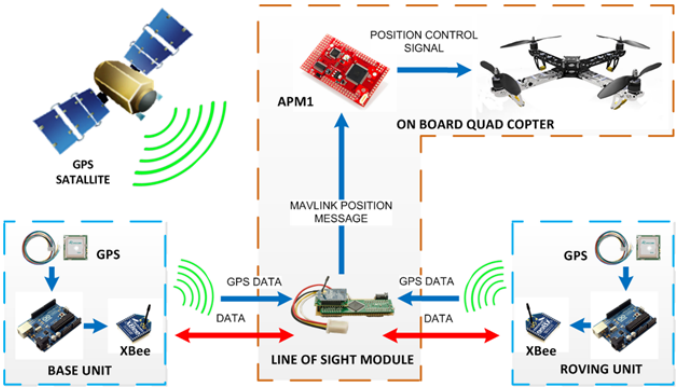


Fig. 2. Search-and-rescue module block diagram.

The Secure Digital Camera (SDC) is a novel approach in capturing digital images [5]. For better understanding the importance of the SDC, a comparison with a plain digital camera can provide useful insight. A simple digital camera is only able to capture digital images and maintain a visual record of events. However, it is not possible to track the source, ensure the authenticity and procession of custody for the digital images. Even the use of digital watermarking capabilities cannot provide indisputable authentication of the images. Data loss might also occur in a digital camera. Thus, with the digital camera there is also scope of eavesdropping and manipulation of the multimedia file. The SDC on the other hand, with is unique components is capable of tracking the identity of the photographer, corroborate image veracity and maintain the chain of custody with detailed records of point in time, day, month, year and other significant information [6]. From the above discussions, it is evident that the SDC is arguably one of the best proven appliances of capturing multimedia. It attempts to eliminate privacy and security concerns.

As multimedia usage continues to expand with an enormous number of applications, the demand for high quality images with acceptable size has been dramatically increased. BPG [7] is a novel step in the field of image compression that aims to supersede the decades-old *de facto* JPEG format [8] with its distinct attributes: meeting modern display requirements (high quality and lower size) of developers, programmers and graphic businesses. HEVC (High Efficiency Video Coding) [9] and compatibility considerations are accommodated in the form of a small JavaScript (56 KB) decoder which is one of the key composing elements of the new format. Unlike JPEG, BPG does not require supplementary browser plug-ins to display the compressed image. Other attributes that differentiate BPG from JPEG and make it an excellent choice include the following:

- The open source and royalty-free and patent-free nature of the BPG justifies it as a more appropriate choice for users because they do not need to be concerned with legal issues.
- BPG is close in spirit to JPEG and can offer lossless compression.
- With advanced quality features, BPG offers different chroma formats making it compatible with multiple video encoding schemes such as analog, digital, and JPEG encoding schemes. Chroma formats supported include grayscale, RGB, YCgCo, YCbCr, Non-premultiplied alpha, and Premultiplied alpha.
- BPG uses a range of metadata for efficient conversion including EXIF, ICC profile, and XMP.
- BPG is capable of cross-platform use through its JavaScript interpreter.

The rest of this paper is organized as follows: The novel contributions of this paper are summarized in Section II. Section III describes related prior work. Section IV discusses the relevant flight dynamics and control systems of quadrotors in general. Section V presents the hardware considerations taken into account for this design. Section VI describes the hardware setup and communication protocol followed in this design. Section VII presents the object detection and video processing algorithms used. Section VIII illustrates the secure digital camera for the quadrotor. In Section IX, the proposed algorithm and architecture for BPG image and video compression are discussed. Section X discusses the custom control software developed for initialization and autonomous operation, while Section XI concludes the paper and briefly discusses areas of future development of this platform.

II. Novel Contributions of This Paper

The main objective of this paper is to describe a unified and modular hardware architecture of an easily customizable quadrotor with integrated secure better portable graphics (SBPG) compression encoder that is integrated with an SDC. The objectives are twofold. On the one hand, the proposed SBPG architecture offers double-layer protection in the form of encryption and watermarking, which addresses issues related to security, privacy , and digital rights management (DRM) [6]. On the other hand, the paper proposes an SDC integrated with secure BPG compression for real time intelligent

traffic surveillance (ITS). The proposed architecture meets modern technology requirements: high quality and smaller size because of using BPG compression. To the best of the authors' knowledge, this is the first ever proposed hardware architecture of an SDC that is integrated with SBPG compression encoder.

The **novel contributions** of this paper are the following:

- Design of a functional low-priced quadrotor based on modification of existing proprietary and open-source platforms.
- Design of a medium resolution (640 $\times$ 480) optical camera system and attached to the quadrotor.
- A ground control station was designed and prototyped. It provides for autonomous wireless control of the quadrotor without affecting the vehicle's payload. PID control was also implemented on-board.
- Wireless video transmission was achieved between the quadrotor and the ground station using commonly available off-the-shelf components.
- The OpenCV computer vision software platform was modified to accomplish all video related tasks, including pattern recognition.
- A library of serial communication functions was custom developed for this project to allow the control of the quadrotor from the ground station if autonomous flight fails.
- A novel algorithm and architecture for SBPG that is integrated with SDC, suitable for real time intelligent traffic surveillance (ITS), is presented.
- A novel simplified hardware amenable architecture for hardware BPG compression is proposed.
- The first-ever Simulink®-based prototype of the BPG compression architecture is implemented.

III. Related Prior Research

In the last decade, a growing number of researches have been conducted toward using quadrotors in image communication. The authors in [10] present a brief history and overview of quadrotors. They also show the reliability of the whole system when a PID algorithm is implemented. The research also discusses the design and testing aspects of a functioning quadcopter test-bed. Similarly, in [11] the authors show various aspects of utilization of color histograms in a quadrotor. The applications are diverse, such as the ability to report location and size of any target image and the ability to position a quadrotor coming from any angle. In [12], a scheme is proposed to control quadrotors that makes use of image based visual servos which are integrated with adaptive back-stepping control. The proposed system minimizes the errors related to image feature positions. Shoufan [13] has considered an architectural design that ensures secure communications between a quadrotor and a ground station. A dedicated software application was also used to test the proposed security function. An extension board is used to integrate the system and the implementation is made on an FPGA. Another quadcopter system is proposed by Senkul *et al.*[14]. Tilting rotors are used in this system as an alternative propulsion design. This system fills gaps related to some of the drawbacks of regular quadrotors. Better performance of this system was seen when compared with a traditional quadrotor. Asymptotic trajectory tracking can be a serious problem for under actuated quadrotors and [15] presents a new controller to deal with the issue. This controller changes the kinetic description of the system. Lastly, the authors in [16] have presented a communication system that allows a quadrotor or UAV to have safe operations in unfavorable environmental conditions.

A work by Mohanty [6] demonstrated a unique approach for a secure digital camera with a double-layer of protection, integral watermarking and encryption capabilities. The proposed architecture considers hiding binary images and their secure authentication. It also provides a method for FPGA implementation. Its compatibility was also assessed with different multi-media constructing electrical devices, and system-on-a-chip (SoC) technology is a central component of the model. Darji *et al.*[17] show the development of hardware capable of entrenching invisible watermarks using a LeGall 5/3 Discrete Wavelet Transform (DWT). In the suggested structural design, the authors have considered all the limitations of a digital camera. JPEG compression was used to assess the algorithm, and Verilog HDL was used to prototype the processor in 0.18μm VLSI technology. In [18], a novel scheme is introduced to support pictures and illustrations captured by digital cameras. Two techniques proposed employed combination of semi-brittle and vigorous blind type watermarks. For development of watermarks, the authors used the rate of information recurrence and the proprietors biometric records. It is worthy to note that the proposed plan is capable of meeting the requirements needed for image verification and pattern protection. Lastly, [19] provided an innovative approach for putting into practice the two observable digital image-watermarking methods. These methods are principally based on the very large scale integration construction approach, which is capable of incorporating into any other available digital camera structure. The proposed designs follow a series of steps for dealing with the watermark on a pixel-by-pixel basis based on signal-to-noise ratio.

The JPEG standard's successor, JPEG-2000, intends to overcome several of the existing shortcomings such as better compression ratios, compression scalability, and resolution accuracy. Ghodhbani *et al.*[20] suggested that hardware implemented JPEG-2000 encoding is more efficient and optimized than current software implementations and demonstrated an optimized EBCOT algorithm architecture implemented on an FPGA platform. Improved operational efficiency was observed for a pipelined BPC encoder implemented in the VHDL Hardware Description Language (HDL).

Liu *et al.*[21] particularly studied the HEVC which implements compression methods based on 64$\times$64 blocks and minimum recursive block partitions of 4$\times$4. Prediction modes and tree-structures improve HEVC coding efficiency. A fully pipelined parallel HEVC implementation with negligible Peak Signal-to-Noise-Ratio (PSNR) was demonstrated that allows real-time encoding, such as 1080p at 30fps with minimal hardware at 600 MHz.

The Ultra High Definition Television (UHDTV) format is

expected to support 3840×2160 and 7680×4320 resolutions at 120 fps. This implies a data throughput 100 times higher than current 1080p HDTV. Zhou *et al.*[22] proposed optimizations such as pre-normalization, hybrid path coverage, binarization components, context modeling and lookahead rLPS to reduce the path delay of the BAE. These optimizations are possible by exploiting the incompleteness of data dependencies in rLPS updating, which yields a Context-Adaptive Binary Arithmetic Coding (CABAC) encoder at 4.37 bins/s i.e. a 45.3% optimization costing 4.8% BPCC performance degradation and 62.5% better performance than current architectures.

Optimized VLSI architecture techniques allow high performance SAO encoding in HEVC. Mody *et al.*[23] demonstrated 4 K resolution at 60 fps at 200 MHZ using 0.15 mm^2 of silicon area in a 28 nm CMOS process with artifact avoidance algorithms, which provide 4.3% savings in SAO encoding.

IV. Quadrotor Sytem-Level Description

In this Section a description of the quadrotor is presented with brief discussion of each of the system-level components.

A. UAV

The unmanned aerial vehicle (UAV) is a powered, aerial vehicle that is either remotely piloted or controlled autonomously. A remotely piloted UAV is known as a drone. Drones have been around since the early 20th century. However, in recent years, many UAVs have been developed with autonomous control allowing them to carry out preprogrammed flight plans. They are most commonly used for military applications including reconnaissance and attack missions [24]. Furthermore, they are increasingly being used for other non-military tasks including fire fighting, surveillance, and inspection.

B. Quadrotor

A quadrotor is an aerial vehicle whose lift is provided by a set of four rotors. The rotors can be separated into two pairs, each pair rotating in the opposite direction to the other. Control is achieved by varying the speeds of the rotors either individually or in pairs. Unlike a regular helicopter, the blades on a quadrotor have a fixed pitch and the design as a whole is much simpler as the only moving parts are the four rotors. This reduces the maintenance time and cost of the vehicle. Quadrotors have several advantages over fixed wing aircrafts and traditional rotor helicopters. These advantages include the following: (1) simplified design, (2) the ability to hover, (3) vertical takeoff and landing (VTAL) capabilities, (4) maneuverability, and (5) ability to fly both indoors and outdoors. Small rotors impart less kinetic energy to quadrotors. This makes them safer and less prone to causing substantial damage when colliding with other objects. For these reasons, quadrotors have been gaining popularity as a platform for UAV research. The main disadvantage of a quadrotor over other aerial vehicles is its short flight time, which is usually limited to about 15 - 30 minutes.

C. Flight Dynamics

When considering the motion control of an aerial vehicle the three main parameters involved are the angles of rotation about the vehicles center of mass. These are known as the following: (1) pitch, (2) roll, and (3) yaw. They refer to the rotation about their respective axes: x (roll), y (pitch) and z (yaw), with x being the direction of flight and z being the perpendicular. Altering these parameters either individually or together enables the motion of the vehicle to be altered as needed [25]. In a quadrotor the only moving parts are the four rotors and are considered in two pairs. Two opposite motors form one pair that rotates counterclockwise while the other two motors form the second pair that rotates clockwise. All motions are obtained by varying the speed of each rotor individually or in pairs [26], [27]. The individual rotors are identified by compass directions (N, S, W, E) with North corresponding to the direction of flight.

1) Altitude: In order for the quadrotor to hold a constant altitude, the thrust generated by the rotors needs to equal the downward force due to the mass of the vehicle. Uniformly increasing the speed of all four rotors will increase the thrust generated resulting in a net upward force thus causing an increase in altitude. Similarly uniformly decreasing the rotor speed will result in reduced altitude.

2) Pitch: By decreasing the speed of the forward facing (N) rotor and increasing the speed of its opposite (S), the pitch of the vehicle is decreased. This results in a forward motion of the quadrotor. To move the quadrotor in the opposite direction the speed of N is increased while the speed of S is decreased. This results in a backward motion of the quadrotor.

3) Roll: A change in roll is obtained by similarly altering the speeds of the sideways (W, E) rotors. These speed changes result in motion in either the left or right directions.

4) Yaw: To prevent the quadrotor from rotating about its vertical axis, the torque produced by the rotors needs to be balanced. Therefore, rotors W and E need to rotate in the opposite direction as that of rotors N and S. When all four rotors are rotating at the same speed they generate an equal and opposite torque, which results in a net torque of zero and a constant yaw angle. When the speed of one pair of rotors is increased relative to the other pair, the total torque will no longer be zero, which results in rotation or yaw. The rotation of the quadrotor is opposite to that of the faster rotating rotor pair.

D. Control System

The quadrotor is a dynamically unstable nonlinear system. This makes the quadrotor difficult to fly without an embedded control system [28], [29]. This is highly essential for autonomous flights. A closed-loop feedback control system, as represented in Fig. 3, is implemented in the quadrotor to achieve stable flight. In the case of a quadrotor, the process is the vehicles dynamics and the output $Y(s)$ is the current position and orientation. The various sensors measure this information in real-time and compare it to the desired value. The difference of the desired input value and actual output is the tracking error ε. The controller processes this error

and the output is the new speed for each individual motor. The change in motor speed results in a change in the system output, which in turn is compared to the desired input and the cycle continues to repeat this process. Many studies have been conducted to test the response of various feedback controllers. These include Dynamic Contraction Method [30] and Linear-Quadratic Regulator (LQR) [31].

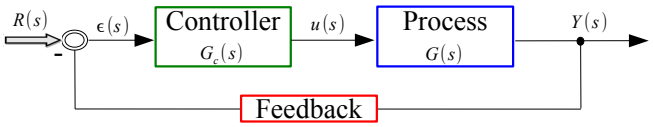


Fig. 3. Closed-loop feedback control system.

The Proportional-Integral-Derivative (PID) controller is a closed loop feedback controller consisting of three constant parameters, as shown in Figure 4. This controller is the most widely used in the industry mainly due to its simplicity and good performance in a wide range of operating conditions [32]. This type of control is implemented in the presented design.

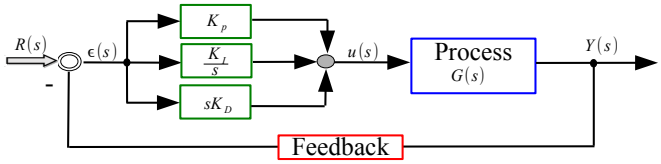


Fig. 4. Closed-loop feedback system with PID control.

The output in the time domain is given by the following expression:

$$u(t) = K_P\varepsilon(t) + K_I\int \varepsilon(t)dt + K_D\frac{\varepsilon(t)}{dt}, \qquad (1)$$

where $u(t)$ is the process input, $\varepsilon(t)$ is the error and K_P, K_I and K_D are the proportional, integral and derivative control constants, respectively and are derived by extensive tuning during pre-flight tests. The Laplace transformation of Eqn. 1 results in the transfer function of the controller $G_C(s)$:

$$G_C(s) = K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s}. \qquad (2)$$

Each of the three terms in Equation 2 has a different effect on the system and by adjusting the K_P, K_I and K_D values the system can be tuned for the desired output response [26]. The proportional control produces an output that is proportional to the error and is adjusted by changing the K_P value. For a given error a high proportional gain will result in an increased sensitivity and large change in the output while a small gain will result in reduced sensitivity and a slower response. However, too much increase in the proportional gain may result in an unstable control system. Adjusting the K_P value allows for the rise time and steady-state error to be reduced at the cost of increasing overshoot. There is always steady-state error with only proportional control. The integral term is proportional to the duration and magnitude of the error. This has the effect of eliminating the steady-state error. However, it may have a negative effect on the transient response of the control system. The derivative response is proportional to the rate of change of the process. Adjusting K_D can increase the stability of the system, reducing the overshoot and improving the transient response. The derivative term is highly sensitive to noise. Therefore, a low value is normally used.

V. Hardware Components of Quadrotor Architecture

Many different quadrotor platforms have been developed for both research and commercial applications. Most tend to be relatively costly ranging from $,2000 - $,7000. The ArduCopter [33] (succeeded by the APMCopter [34]) is an open source quadrotor UAV project with a large active community and low cost components. The controller is based on the popular Arduino platform and is undergoing constant development. This makes it a good platform for further development. The hardware was purchased from three vendors. The ArduCopter frame, controller and sensors were purchased from 3D Robotics, the manufacturer of the ArduCopter. The Radio Controller, video camera and transmitter, battery, battery charger, battery sensor, and the USB capture device, were purchased from various online resources. The total cost of the hardware was $1,276 which is very reasonable for its intended applications. The hardware will be discussed in three different sections: the ArduCopter (which consists of the frame, the drive system, the controller or autopilot and the sensors), the radio controller and the ground control station (that includes the wireless telemetry, video capture system and computer).

A. ArduCopter

1) Frame: The frame was purchased as a kit that contained the main plates, arms, motor mounts, battery mount, the carrier boards and the landing gear as shown in Fig. 5 [35].

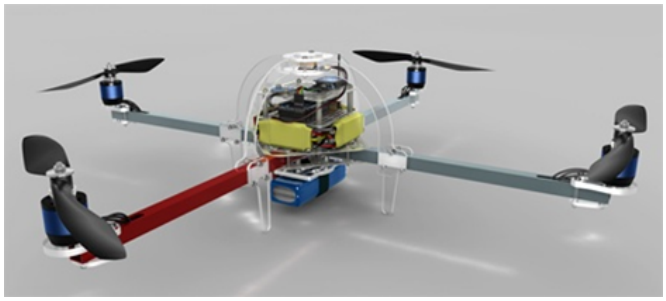


Fig. 5. Assembled quadrotor.

2) Drive system: The drive system consists of four electronic speed controllers (ESC) that drive the brushless DC motors. The system is powered by a battery pack via a power distribution board and the ESCs receive a Pulse-Width Modulated (PWM) control signal from the ArduPilot Mega (APM) controller, as shown in Fig. 6. The APM consists of an Arduino Mega microcontroller and associated firmware.

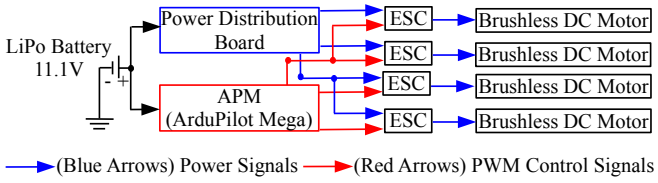


Fig. 6. Drive system block diagram.

3) Brushless DC motors: Brushless DC motors are used over brushed motors as they have a higher efficiency, experience less wear, produce less noise, allow for more accurate speed control and offer a better thrust to weight ratio. However they are more expensive and require more complex and costly control electronics. The current and torque relationship is linear as well as the frequency and speed relationship. Brushless DC motors are rated by their K_m (motor constant) and K_v (motor velocity constant) values. The K_v rating of a motor is the ratio of the unloaded Rotations Per Minute (RPM) to peak voltage. Depending on the motor configuration they are normally referred to as either "outrunners" or "inrunners". The conventional configuration is the "inrunner" and consists of three stator windings surrounding the rotor which contains the permanent magnets. An "outrunner" consists of the stator coils in the center with the permanent magnets attached to the rotor which rotates around the outside. The motors used have a K_v value of 850. Thus with an 11.1 V supply they are capable of reaching a maximum of 9435 RPM. They weigh 62 g each and can produce a maximum torque of 1095 g.

4) Electronic speed controller: The motors are driven by a programmable electronic speed controller (ESC) which receives a 50 Hz PWM control signal from the APM and switches a network of field effect transistors (FETs). The position of the motor is determined by measuring the back EMF, which allows for the controller to energize the correct coil causing the motor to rotate.

5) LiPo batteries: Two 11.1V lithium polymer (LiPo) 3-cell batteries were purchased, each with a capacity of 3000 mAh and weight of 269 g. This allows for a flight time of approximately 10-20 minutes.

6) Rotors: A set of four 10-inch rotors consisting of two pushers and two pullers, was used. The pushers are used on the clockwise rotating motors (N and S) while the pullers are used on the counter-clockwise rotating motors (W and E). It is important to ensure that the rotors are balanced so as not to introduce vibrations into the system, as this will induce errors in the sensor readings.

7) Controller (APM): The controller is responsible for the stabilization of the vehicle by continually processing the data from the sensors and adjusting the speed of the rotors accordingly. The ArduPilot Mega (APM) is a controller board based on a 16 MHz ATMega1280 microcontroller and is responsible for the stabilization and navigation. A PID feedback control loop is implemented in the controller in order to stabilize the vehicle.

8) Sensors: Various sensors are used to determine the current position, orientation and velocity of the vehicle. The majority of these sensors are located on the Inertial Measurement Unit (IMU), while others are mounted on the frame and connected to either the IMU or the APM. A separate board (ArduPilot Mega IMU Shield) is required to interface the sensors with the main controller. This board attaches to the APM and has an onboard *gyroscope* and *three-axis accelerometer*. It also allows for a *magnetometer* and various other sensors to be connected. A *GPS module* attaches directly to the controller and communicates via a USB/UART interface. It is a MediaTek MT3329 and allows for positioning of the vehicle within 3 m of the desired location. In order to hold the same position it is important to continuously know the direction the vehicle is facing, as the controller needs to continuously calculate the corrections that need to be made to account for the drift in the yaw gyroscope. GPS can only calculate a directional vector when the vehicle is in motion thus a magnetometer is required to determine the direction while hovering in a single position. The magnetometer used is based on Honeywell's HMC5843 and communicates through an I^2C interface. It was soldered onto the IMU, leaving the I^2C port free to be used for other peripherals if required.

9) Video camera and transmitter: The onboard video system consists of a ½ inch CCD NTSC Sony camera with a resolution of 510 × 492 and a 2.4 GHz four-channel video transmitter used to transmit the video back to the ground station. The transmitter has two connectors, a 2-pin power input connector that was soldered to the power distribution board, and a 5-pin video in connector. As seen in Fig. 7, two pins are used to power the camera, two are used for the microphone input (not connected) and the remaining pin is the analog video signal.

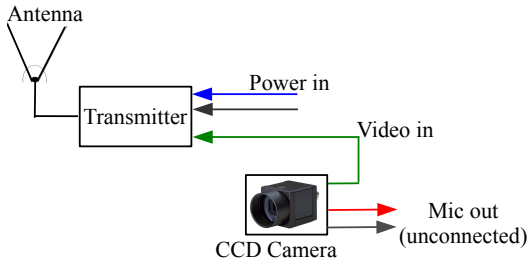


Fig. 7. Camera system block diagram.

B. Radio Controller

It is important to have a manual control that can override the control signals sent by the computer at all times, hence a standard 6 channel RF Radio Control (RC) unit and receiver is used. For each channel the RC transmits a PWM signal that is decoded by the ArduCopter. The controller consists of two control sticks each with two degrees of freedom that make up the first four channels. These channels provide the basic flight controls, namely throttle, yaw, pitch and roll by sending a varying PWM signal that ranges from 1000 to 2000 microseconds. The remaining two channels are toggle switches that only have two states, low and high, represented by a PWM value of 1000 or 2000 microseconds respectively. These toggle switches can be used to set custom control modes for the ArduCopter. In this design channel 5 is used to toggle

between a manual stabilized control mode and the altitude hold mode and channel 6 is left unused. The layout of the first four channels is described by 4 different modes, with Mode 1 and Mode 2 being the most common. Mode 1 is more popular in the United Kingdom and has the throttle and yaw on the right hand side stick. Mode 2 is used in this design. It is the favored mode in the United States and has the throttle and yaw on the left hand stick. The PWM range will differ between RC controllers, thus the ArduCopter needs to be calibrated. This can either be done by setting the values manually with the command line interface or by using the Mission Planner GUI based calibration [36].

C. Ground Control Station

The ground control station (GCS) which is presented in Fig. 8 handles all the video processing and consists of a laptop computer, wireless video receiver, USB video capture device and a USB XBee wireless module for telemetry [35]. Telemetry is the transmission of the sensor data and commands between the ArduCopter and the GCS. The sensor data includes current position data from the GPS, altitude, velocity, orientation and RC PWM values. The ground station is connected to the XBee module via an XBee Explorer which allows for the serial commands to be sent and received over USB.

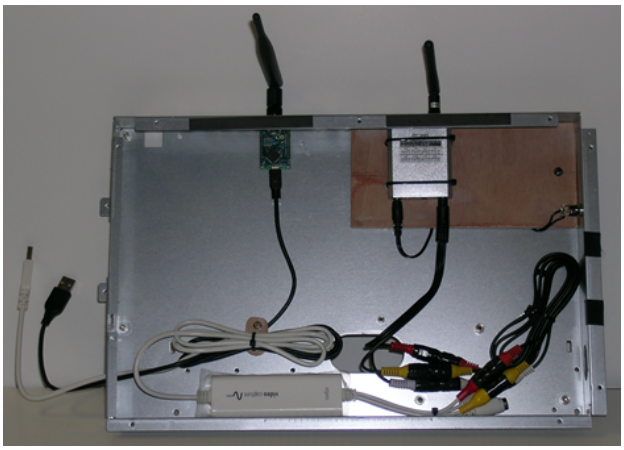


Fig. 8. Ground control station.

1) Wireless video receiver: The video receiver requires a 12 V power supply. Therefore, a 2.1 mm power jack was placed on the right hand side of the GCS. This allows for the receiver to be powered with a wall wart or a 12 V battery may be used.

2) USB capture device: A capture device is required to convert the analog video output from the wireless video receiver to a digital format that can be used by the software. These capture devices are available as either internal capture cards or external devices that are commonly connected via USB or FireWire. An Elgato 10020840 external USB 2.0 device was chosen, as it can be used with either a desktop or portable computer and USB connections are more common than FireWire. It supports a video resolution of 640 $\times$ 480 and uses either the H.264 codec at 1.4 Mb/s or the MPEG-4 codec at 2.4 Mb/s.

3) Xbee: XBees are small, low-powered radios well suited to low bandwidth RF applications. They implement a simple serial command set making them ideal to wirelessly interface a microcontroller and personal computer. The two XBee modules considered were the XBP09 and the XBP24, which operate at 900 MHz and 2.4 GHz, respectively. The XBP09 is capable of point-to-point, peer-to-peer and point-to-multipoint networking and has a range of up to 10 km and a data rate of 156 Kb/s. The XBP24 modules are based on the IEEE 802.15.4 standard (the basis for ZigBee) and have a higher data rate of 250 Kb/s but a reduced range of 1.6 km. The XB09 module was selected due to its greater range and operating frequency of 900 MHz, which would not cause interference with the 2.4 GHz RC. In order to interface the XBee module with the APM an XtreamBee board is used in the current design.

4) Computing System: A command line program is executed on a laptop running Mac OS 10.6.8 with OpenCV 2.3.1 installed. Two USB connections are required for the XBee and video capture device.

VI. Wireless Communication in Quadrotor

Video processing requires a substantial amount of processing power and is beyond the capability of the AVR microcontroller used to control the APM. To perform the processing onboard the quadrotor a microcomputer (BeagleBone, Raspberry Pi or similar board) capable of running OpenCV would be required. This would add weight and require significantly more power thus reducing the battery life and flight time considerably. Instead, a ground control station (GCS) is used to do all the heavy processing. This is accomplished by sending the video to the GCS via a wireless transmitter. The GCS processes the video and determines the actions to be taken, sending back control commands to the ArduCopter. As seen in Fig. 9, the video link is a one-way downlink while the control link is bidirectional, allowing for the GCS to send and receive data from the ArduCopter. It is important for these links to operate at different frequencies so as not to cause any interference. The RC operates at 2.4 GHz, thus a 5.4 GHz video transmitter was selected. The data link consists of two XBee modules.

A. Serial Communication

The RS-232 standard serial communication protocol involves sending a byte of data one bit at a time over a single transmission line (TX) and it receives a packet of data over the RX line. The serial packet consists of 10 bits. First the start bit (always logic low) is sent. Next, the character is sent from least significant bit (b_0) to most significant (b_7) and finally the stop bit is sent, which is logic high. Some connections make use of a parity bit, which comes before the stop bit. However, the XBee model does not make use of the parity bit. This connection thus has a 2-bit overhead and is known as an 8N1 connection (start/stop bit with no parity). Serial connections were designed to transmit data over relatively long distance and have a range of about 15 m. These connections make use of voltages up to 25 V. Long data lines however

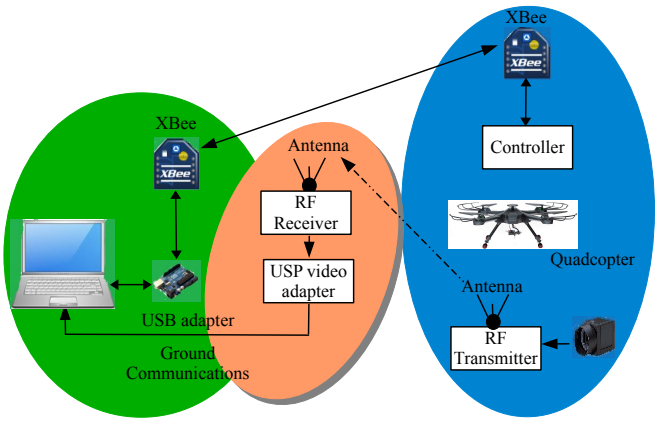


Fig. 9. Wireless communication setup in which the quadrotor is an integral component.

are not needed, so the XBee uses TTL levels of 0 - 3.3 V. The XBee has a full duplex connection and can thus send and receive data at the same time. The speed of the connection is described as the baud rate and is measured in bits per second (bps). A serial connection traditionally consists of a 25-pin D-sub connector however a minimal serial connection can be made with just 3 or 5 wires. The XtreamBee serial break out board for the XBee uses the 5-wire connection. Serial communication is implemented by using the standard UNIX library `termios.h`, which has functions for setting up and connecting to a serial port. Passing the serial port name and the required baud rate to the serial port initialization function opens the port and sets up the required option flags for an 8N1 connection.

B. MAVLink

The ArduCopter uses MAVLink [37], a two-way communication protocol based on the W-CAN and SAE AS-4 standard that was developed specifically for micro air vehicles. MAVLink consists of a header library that has implemented various commonly used messages that allow for settings to be updated, sensor readings to be read, modes to be changed and control commands to be sent. If custom messages are required they may be generated in either C or Python. A message packet varies in length and can be between 8 and 263 bytes. Every packet consists of at least 8 bytes and the message type determines the length of the payload which can be between 0 - 255 bytes. The first byte is the packet start sign (0x55) and is followed by the payload length (0-255). The third byte is the packet sequence and represents the number of messages that have been generated during the current session. This number is automatically incremented when each new message is packed. The next two bytes are the system ID and the Component ID. This allows for multiple systems to communicate with multiple vehicles as a message can be addressed to a specific one. The sixth byte is the Message ID that determines the function of the message. There are a number of commonly used messages as well as some that are specific to the ArduCopter. The payload can vary in length from 0 - 255 bytes depending on the message and it carries the information either being sent to or received from the ArduCopter. The payload can contain 8/16/32/64 bit signed or unsigned integers, floats, doubles or char values. Finally the last two bytes consist of a 16 bit checksum generated by the same means as that used in the ITU X.25 and SAE AS-4 standards. The number of messages that can be sent per second is determined by the speed of the serial connection and the payload size. The XBee modules have a baud rate of 57600 bps, therefore the number of messages per second can be described by the following:

$$\begin{aligned}\text{Messages per second} &= \left(\frac{\text{Baud rate}}{10 \times \text{Number of bytes in message}}\right) \\ &= \left(\frac{57600}{10 \times (8 + \text{Payload length})}\right).\end{aligned} \tag{3}$$

This results in a message rate of between 21.9 Hz and 720 Hz depending on the payload length.

The main message used in the program is the RC Channel Override message, which allows for the program to set the PWM values for each channel overriding the RC values for one or multiple channels. The message payload consists of 18 bytes. The first two bytes are the target system and component ID, which are both set to one. The remainder of the payload consists of eight 16 bit unsigned integers representing the PWM value for each channel. To override the channel this value should be between the minimum and maximum PWM values for that channel, normally between 1000 and 2000. To release control of a channel back to the ArduCopter a value of 0 is set and to leave the current value for the channel unchanged a value of -1 (0xFFFF) is set. Each message has a corresponding pack function that takes the required values and generates the message. MAVLink has already been implemented on the ArduCopter and thus, only needed to be incorporated in the ground control station program. On the GCS the message is packaged with the required MAVLink pack method and then sent out over the serial port. The ArduCopter then receives the message and checks the data integrity by comparing the checksum. If there is no data lost then the message is unpacked and the data is passed to the required function determined by the message ID. A number of helper functions were created to pack and send messages out over the serial port. To arm the motors, after the ArduCopter has booted up, the left control stick is held in the bottom right position for a few seconds. This corresponds to a throttle value of 0% and a yaw right value of 100%. A function was created to arm the motors by sending a message with the corresponding PWM values for channels three and four, waiting 4 seconds and then handing control back to the RC.

VII. Object Detection and Video Processing in Quadotor

Object tracking is performed using a vision system. It consists of three steps: (1) the detection of the desired object, (2) the tracking of the object between frames and (3) the analysis of the changes in object position to determine the behavior of the object. This has many applications including

vehicle navigation, surveillance and motion-based recognition. The OpenCV based control software running on the GCS performs the object tracking. This section will briefly cover computer vision, the OpenCV library and the custom template matching algorithm used.

A. Computer Vision, Object Recognition and Video Tracking

Object recognition is used to find a given object in an image or video sequence. This can be an extremely difficult task as the appearance of the object to be detected is dependent on various factors, such as the position of the object relative to the camera, lighting variations and differences in the object models. There are three main methods used in computer vision to detect objects: geometry-based, appearance-based and feature-based. Geometry-based methods use geometric models and were used in the initial attempts of object detection, however this proved to only be effective in limited situations. Appearance-based and feature-based methods [38] are currently more common and will be discussed briefly. Appearance-based methods use a reference image to identify the object; however the object may not be reconcilable under different lighting conditions and at different angles. Therefore multiple images are generally used. Some of these techniques include edge matching, divide and conquer search, grey scale matching, and gradient matching. Feature-based methods involve analyzing the image for possible feature matches with the object features. Surface patches, corners and linear edges are features that could be used to match an object. Some common methods include interpretation trees, hypothesize and test, pose consistency, pose clustering, invariance, geometric hashing, scale-invariant feature transform (SIFT) and speeded up robust features (SURF). A video tracking method must be selected according to the object to be tracked as each method has better performance for a specific subject. Blob tracking [39] is useful for detecting human movement as the shape is continuously changing. Other algorithms include kernel based tracking and contour tracking. Filtering algorithms are more complex and can be used for tracking objects that have been partially obscured or have a more complex shape. Common filtering algorithms include the Kalman filter and the particle filter.

B. OpenCV

OpenCV is an open source real-time video-processing library originally developed by Intel in 1999 that can be freely used for commercial or non-commercial applications. It is now maintained by Willow Garage and is under active development with a new release every six months. It was originally written in C, however the latest version has a new C++ interface as well as wrappers for various other languages including Python, Ruby, Java and C#. At its core OpenCV consists of a collection of cross-platform C functions and C++ classes that allow for high-speed common computer vision functions to be performed. Libraries for many applications are available including facial recognition, motion tracking, machine learning and mobile robotics. As can be seen in Fig. 10, CXCore implements all the basic structures, algorithms and drawing functions while CV handles all the image processing and vision algorithms. HighGUI is responsible for the creation of windows, handling mouse and keyboard events, adding trackbars, reading and writing images from or to disk, capturing video from a camera and writing video to a file. CvCam allows for video to be captured and processed and all the machine-learning algorithms reside in the ML library.

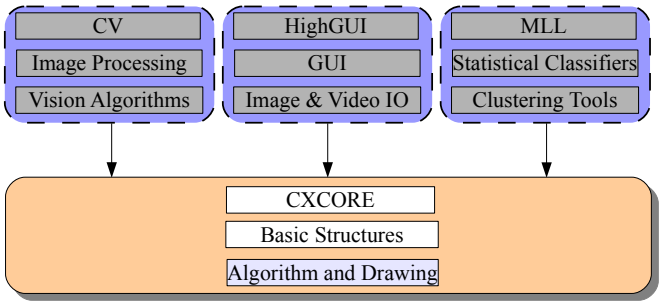


Fig. 10. OpenCV software architecture.

C. Template Matching

Template matching is one of the most common techniques used for object detection and has been implemented in OpenCV as the `matchTemplate function`. In order to perform the match, a template image is used to identify an object in a source image by comparing the pixel values of the target with those in a portion of the source image. A result image is created with each pixel value representing the confidence of the match at the corresponding location. There are six different comparison methods available in OpenCV [40]. The template image T with pixel width T_w and pixel height T_h is initially overlaid on the source image S with pixel width S_w and pixel height S_h, starting in the upper left corner as shown in Fig. 11.

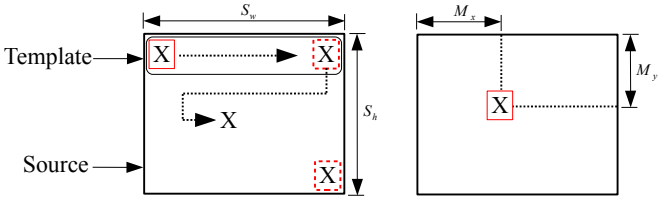


Fig. 11. Sliding window method used for template matching.

A comparison of the pixels is performed and the result R is the confidence of the match at this location. The template image is moved 1 pixel to the right ($x = x + 1$) and another comparison is performed. When the right edge of the template image reaches the right edge of source ($x = S_w - T_w$) the template image is then moved down 1 pixel ($y = y + 1$, $x = 1$) and the process is repeated until the template reaches the bottom right corner ($x = S_w T_w$, $y = S_h - T_h$). The result image will have a width of $R_w = S_w T_w$ and a height of $R_h = S_h T_h$. The location in the result image with either the highest or lowest value, depending on the method used, has the highest match probability. This point is relative to the result

image and thus needs to be converted to a point relative to the source image, using the following equations:

$$M_x = R_x + S_x - \frac{T_w}{2} \quad (4)$$

$$M_y = R_y + S_y - \frac{T_h}{2}, \quad (5)$$

where (M_x, M_y) are the coordinate locations of the match with respect to the source and (R_x, R_y) are the coordinate locations of the match with respect to the result image. One of the disadvantages of this method is that it can be slow if the source image is relatively large or the template image is much smaller than the source. In this case the source image from the video feed is 640 $\times$ 480 pixels. Assuming a template image of 100 $\times$ 100 pixels then 205,200 comparisons of the template image would be performed as seen below:

$$\text{No. of comparisons} = (640 - 100) \times (480 - 100) = 205,200. \quad (6)$$

Another disadvantage is that the target in the template image needs to be the same size as the target in the source image. This is a problem as the size of the target in the source image is dependent on the distance between the quadrotor and the target. There are two possible ways to overcome this, the first is to get an altitude reading from the ArduCopter and scale the target image accordingly to match the expected size of the target in the source. This would require that the size of the target is known and that an accurate measurement of the distance to the target can be obtained. The other method would be to attempt a match and if one is not found then scale the target and repeat until a match is found. However this would be significantly more computationally expensive and would result in a much lower frame rate. To perform a template match in OpenCV the template image is loaded as a gray-scale image and stored in a native data structure. The video stream is opened using the desired camera source and then a single frame is grabbed and stored as the source image. The image that is captured form the camera is an 8-bit image and has three channels, red, green and blue. The color channels are not needed, as the target image is black and white, the image is converted to gray scale thus using just one channel. This greatly reduces the size of the matrix (by a factor of almost 3) and results in a significant speed increase.

D. Fast Template Matching

One method of improving the time taken to perform the template matching is to equally scale down the source and template images. By doing this the number of comparisons performed can be reduced greatly. For example, if a source image of 640 $\times$ 480 pixels and a template image of 100 $\times$ 100 pixels are scaled down by a factor of 4 then the number of comparisons performed will be reduced from 205,200 to 12,825 This results in a significantly large computational saving as now 93.25% fewer comparisons are performed, however the accuracy of the location of the match will be less accurate. The confidence of the match will also be affected, as information is lost when the image is down-sampled. An OpenCV function [40] uses this method to down-sample the source and template images and find multiple match locations. The original source image is then searched around these match locations by creating a Region of Interest (RoI) that is centered on the match location with a size that is slightly larger than the template image. This allows for the match location to be determined without a great cost in computation or resolution. The final algorithm used to perform the template matching was based on the `FastMatchTemplate` with image pyramid [41]. The original function was updated to use the new C++ data structures, which do not require memory management. All unneeded code was removed, as only one match location was required, thus simplifying the function. For this application the goal is to detect an object while flying above it and at high altitudes; it is therefore likely that the target is only going to move by a small number of pixels between each pass through the loop. An option was thus added to skip the down sampling step if there was a match previously and instead just search around the location of the previous match. This can produce a small improvement in the time taken to find a match for a slow moving target. This method will also filter out any large changes in the match location that could be due to false positives or other targets being detected and will prevent the quadrotor from making sudden changes or losing track of the current object. The first thing the algorithm does is to check that the template image is smaller than the source image and that the number of channels in each image is equal, otherwise the function will fail. Next, copies of the source and template are made so as to not alter the originals. In the case when there was not a match the last time through the loop, then down sample the images and try to find a match with confidence above the desired value. If a match is found then the match location is set to then new value. When there was a previous match or there is a new match, then the original source image is searched over a small user defined RoI. If a valid match is found in this region then the match location is converted back to the source image coordinates. Finally, the target location may be highlighted on the source image and displayed for the user. This custom developed fast template matching algorithm achieved an average speedup of 20$\times$ over the unoptimized `FastMatchTemplate`.

VIII. Secure Digital Camera for Quadrotor

As quadrotors will be widely used in places where it would be hard for direct access, the images obtained need to be completely trusted as a faithful reflection of the exact situation. Images obtained using a conventional camera are more vulnerable to tampering. So they cannot be used for targeting in a quadrotor. For example, when a quadrotor is used for critical applications like documenting a crime scene, if the images are tampered by hackers, then the very purpose of documenting will not be achieved. Similarly, when a quadrotor is used for civil applications like traffic congestion analysis if secure image processing is not performed, then hackers can easily distort it leading to chaos during important hours of the day. In situations such as environmental assessments, the image can be easily modified by anyone using image editing tools widely available. Hence using a Secure Digital Camera

for these applications is very important as there is a need to protect these images against intrusion.

SDC is a device that has the standard features of a digital camera and built-in facility for real-time, and low-cost and low-power operation [5]. In this paper, we present a novel concept of a secure Better Portable Graphics (SBPG) encoder with built-in watermarking and encryption facility. The BPG compression has several advantages over JPEG compression including high quality with lower size than JPEG, which makes it suitable for real time and bandwidth requirements. SDC is integrated with SBPG and typically designed as an SoC. Using double-layer protection, watermarking and encryption, the SDC addresses many DRM-related tasks including ownership rights, usage tracking, detection and extent of tampering, and facilitating content authentication. Thus, the SDC is arguably one of the best proven ways to facilitate real-time rights management, and is considered to be very suitable for real time applications such as the IoT in highway surveillance systems, as introduced in this paper.

The proposed architecture is the digital camera with built-in capabilities: watermarking and encryption, for the protection of the images that it captures. The system-level block diagram of the proposed SBPG integrated with SDC is shown in Fig. 12. It consists of an active pixel sensor (APS) unit, analog-to-digital converter, liquid crystal display, encryption unit, watermarking unit, and compression unit. The image is captured by an image sensor and converted to a digital signal, then stored temporarily in scratch memory. The image then is further transmitted to the SBPG. The controller unit is responsible for controlling the entire sequence of events. In the proposed architecture, the encryption, watermarking, and compression modules are working together in the system-on-a-chip (SoC) architecture of the SDC.

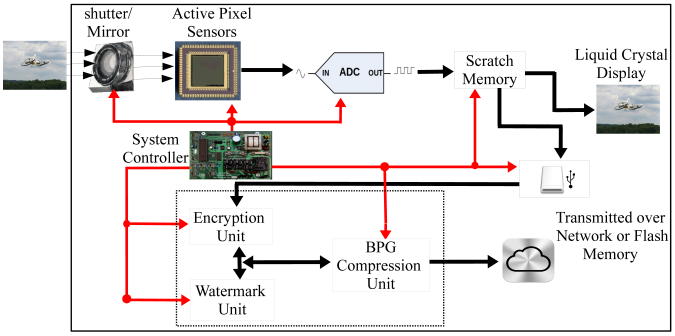


Fig. 12. System-level block diagram of SBPG integrated with SDC.

In the SBPG, an invisible robust blind watermarking approach is used along with Rijndael advanced encryption (AES) [42]. The BPG performs the BPG compression process, which achieves higher compression ratio with smaller size than JPEG for similar quality. BPG compression is based on high efficiency video coding (HEVC), which is considered a major advance in compression technique. The proposed architecture provides two layers of protection, which are the AES encryption algorithm and invisible robust blind watermarking. The encryption or watermarking algorithm alone does not address all the issues related to DRM. For example, an encryption algorithm prevents unauthorized access of the digital content; however, it does not prevent an unauthorized user from doing illegal replication of the decrypted content, which can be addressed by using digital watermarking. Digital watermarking establishes owner rights, and prevents illegal replication; thus, it protects images against unauthorized modification and false ownership claims. Applying both encryption and watermarking algorithms addresses this issue and provides full protection: confidentiality and integrity. Starting with the watermarking and encryption process, then preforming the BPG compression is more secure than starting with the compression process because information is again changed during compression. In the latter case, the original data in a host image is changed by the compression process before applying the watermarking, which means the watermark is altered since it is based on changed information of the host image.

IX. The Proposed Algorithm and Architecture for BPG Image and Video Compression

Proper usage of image compression can make a significant difference in terms of size and quality in order to meet modern display requirements. BPG is considered as the newest technique in image compression. BPG is dependent on HEVC, which is considered as the latest standard in video compression. Unlike other compression techniques such as JEPG, BPG is essentially an I-frame of the HEVC. By using HEVC, the I-frame is encoded more efficiently. Comparing with JPEG which encodes each block independently, HEVC encodes only the differences between blocks rather than deal with the full block information. This can be done by reducing the redundancy between different blocks in the HEVC I-frames.

A. A Simplified BPG Compression Algorithm

BPG is a new image format offering several advantages over the JPEG format. It achieves higher compression ratio with smaller size than JPEG for similar quality. In the BPG format, lossless compression, animation, various color spaces (grayscale, YCbCr, RGB, YCqCo), and chroma formats are supported [7]. The reference BPG image library and utilities (libbpg) can be divided into four functions: BPG encoder, BPG decoder, Javascript decoder, and BPG decoding. The BPG encoder takes JPEG or PNG images as input, performs BPG compression and provides the corresponding BPG image. The BPG decoder does the reverse function. With a small Javascript decoder, the BPG format is supported by most web browsers. The BGP decoding allows any BPG image to be decoded in any program. In the proposed architecture, the focus is in the BPG image encoder compression. The BPG encoder is based on HEVC encoding [9]. HEVC is considered the prime candidate to replace H.264 encoders due to its compression efficiency [43]. The HEVC project aims at reducing the bitrate compared to H.264/AVC because it is more parallel-friendly [44], [45]. Fig. 13 shows the initial steps of the BPG encoder algorithm. It can be seen from the figure that at some point the encoder must check whether an input is a video (dynamic

image) or static image. If the input is video, the algorithm proceeds to the video encoder, shown in Fig. 14; otherwise, the algorithm continues to the image encoder, illustrated in Fig. 15.

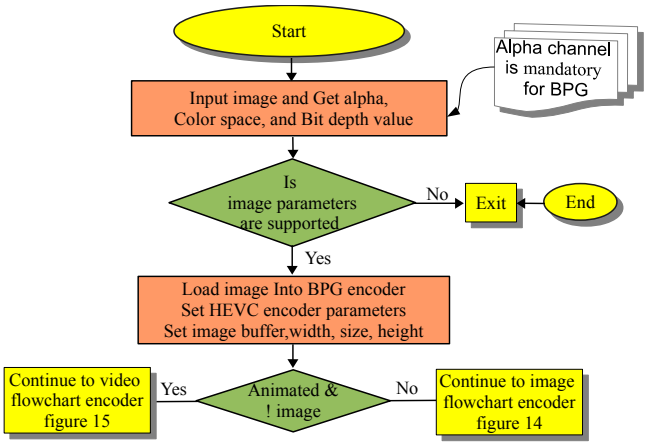


Fig. 13. BPG encoder algorithm.

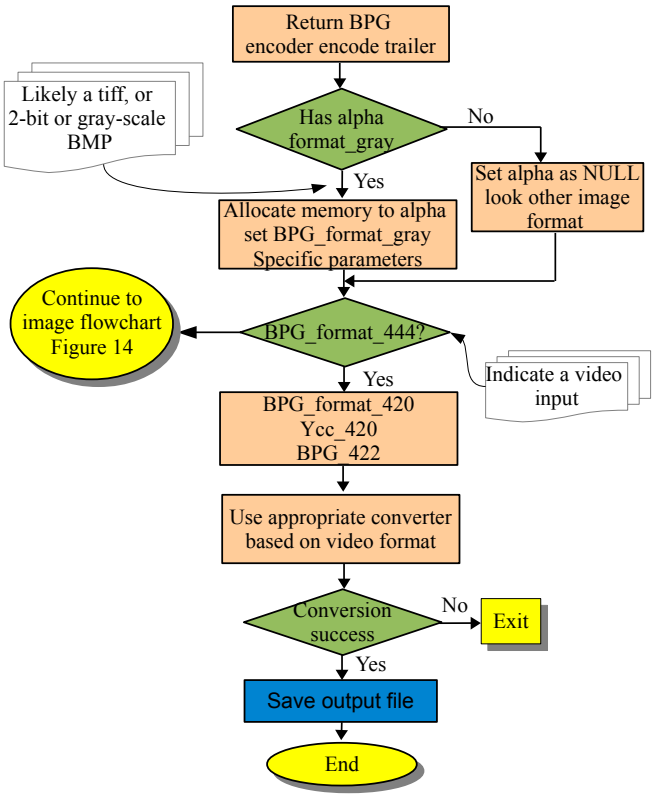


Fig. 14. BPG video encoder algorithm.

After reading the image, the encoder does an initialization processes to read meta data, color space, bit depth, etc. There is an essential step in which the algorithm must check two conditions: bit depth and color space. Bit (color) depth refers to the amount of data that can be used to indicate the color of each pixel [6]. It can be represented by different numbers: 8, 10, 12, $\cdots$. The concern with images that have high bit depths are data storage, and required transmission bandwidth.

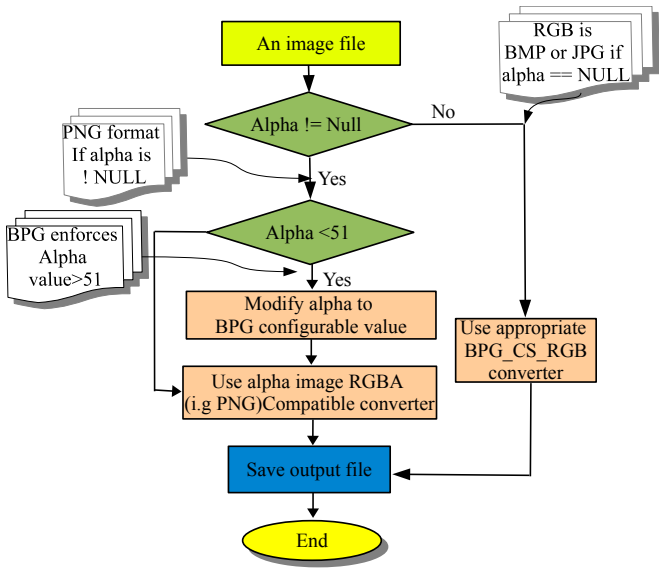


Fig. 15. BPG image encoder algorithm.

Also, some displays are not capable of reproducing all of these colors. Undoubtedly, there must be a trade off between quality and bit depth. The BPG compression encoder strictly considers images with bit depth of 8 for each color's channel.

B. Proposed Hardware Architecture for the BPG Encoder

The hardware architecture of the BPG encoder is presented in this section. BPG compression encoding can be divided into two phases: the pre-encoding (initialization) phase and HEVC encoding, which are shown in Fig. 16. In general, compression can be classified into two categories, lossless and lossy. When the exact original data is recovered, this is called lossless compression, while in the lossy case, a close approximation of the original data is obtained. BPG is capable of both lossy and lossless compression.

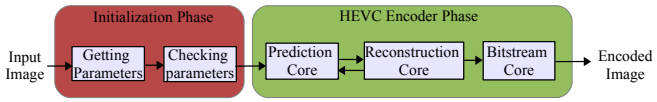


Fig. 16. BPG encoder block diagram.

1) Initialization Phase: Images can have different pixel depth, color spaces, and alpha channel. There are initialization procedures that have to be completed before doing the compression encoding. The first procedure is to obtain the image details: meta data, color space, pixel depth, and alpha. The BPG compression encoder algorithm requires images with bit depth of 8 and true color or grayscale color spaces. These are essential requirements, otherwise the encoder provides an error message indicating that bit depth or color space are not supported. Algorithm 1 illustrates the steps of the initialization phase.

2) HEVC Encoder Phase: BPG encoding is based on the HEVC encoder, which is considered a major advance in compression techniques. HEVC offers high coding efficiency

Algorithm 1 Initialization phase of the proposed BPG compression.

$Parameters \leftarrow \{PixelDpth, ColorSpace, AlphaChannel\}$
$Resolution \leftarrow \{pixels/inch\}$
$ColorType \leftarrow \{TrueColor, GrayScale\}$
if $Length > 2$ **then**
    $Bitdepth \leftarrow \{MateData/ImageSize\}$
    if $Bitdepth \neq 8$ **then**
        $AlphaChannel \leftarrow \emptyset$
        PRINT "ERROR: while opening bitdepth encoder."
Else
if $Bitdepth \neq 8$ **then**
    $AlphaChannel \leftarrow \emptyset$
    PRINT "ERROR: while opening bitdepth encoder."
if $ColorType < 12$ **then**
    PRINT "ERROR: Color space is not supported."
end
PRINT "Bit Depth and color space are supported."
PRINT "Image accepted for BPG compression."
end

because of the intelligent approach that is used to reduce the area (pixels) that is encoded [46]. HEVC uses an 8×8 block as the basic coding unit, and the Discrete Cosine Transform (DCT) or the Discrete Sine Transform (DST) as the transformation mechanism to the frequency domain. In HEVC, the amount of information content (entropy) is considered via context-adaptive binary arithmetic coding (CABAC), which is more efficient compared to the method that JPEG utilizes: Huffman entropy coding [47]. The HEVC encoder encodes the pictures into a bitstream, which contains a sequence of data known as a Network Abstraction Layer (NAL). The encoder stores pictures in the Decoder Picture Buffer (DPB) as illustrated in Fig. 17. A picture in HEVC is divided into one or multiple slices, which contain one or multiple slice segments.

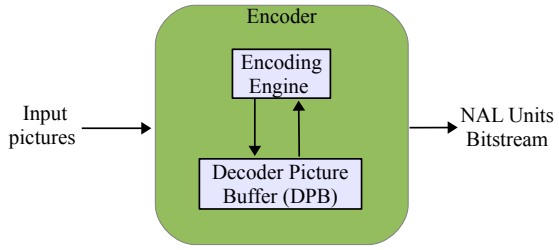


Fig. 17. HEVC Encoding Block Diagram.

HEVC encoding is performed in three stages: prediction, reconstruction, and bitstream core. The prediction core is the essential stage because it handles intra and inter prediction in parallel, where the reconstruction code constructs reference frames at each time of the encoded frame [46]. The bitstream core performs CABAC. The three core stages of the HEVC encoder are the following:

- **Inter Prediction:** in this block the essential task is to reduce temporal redundancy by comparing a current prediction unit with neighboring prediction units, which can be done by motion estimation. In motion pictures, the Inter-picture is performed to make predictions in which motion vectors specify the movement of the suggested image in the direction existing image. The suggested picture is chosen from the interpreted image buffer in order to perform Intra prediction on a block basis, which indicates the displacement location according to the selected picture and its prediction block. Planar motion of rigid object prompts the displacement.
- **Intra Prediction:** is the process of disconnecting the link surrounded by the regions of the picture, if the prediction block process is carried out. Intra prediction is applied to reduce spatial redundancy.
- **Transform and Quantization:** transform is the next step, which is performed after reducing the temporal and spatial redundancy. The size of transform can be 4×4, 8×8, 16×16, or 32×32. Two transforms are used: Discrete Cosine Transform (DCT), and Discrete Sine Transform (DST). The sinusoidal transform is considered appropriate for decorrelation. After the process of transform, the sample is quantized and transformed by entropy coding. Quantization is the process that resolves signals to an assembly of denoted values; however, scalar quantization process involves the quantization of individual values.
- **Entropy Coding:** it is the process of plotting syntax elements into the bitstream. Syntax elements are the transformed quantized elements, which also include intra prediction modes, motion vectors etc. However, the coding stages of entropy differ according to the design criteria. Two categories of syntax elements are used: fixed length codes are applied when syntax elements identify eminent characteristics of the bitstream. Variable length codes (VLCS) are known for coding information on the images. The main objective is to eliminate redundancy which has not been removed by the prediction stage.

The encoder control is responsible for all the encoding of images of the series of shots of video into the bitstream as required by the function. It also ensures that the encoding of video is according to the required videos coding specification. It also makes sure that the decoder buffer of the receiving bitstream has sufficient size. The decisions that the encoder control takes involve the division of the image and the division of code blocks. The proposed hardware architecture of the HEVC encoder is shown in Fig. 18.

C. Simulink® Implementation of the Proposed BPG Encoder Architecture

The system-level architecture of the proposed BPG compression encoder is illustrated in Figure 19. The blocks shown in the dotted lines perform the initialization phase and HEVC encoder. The initialization phase preprocesses the input image and obtains image details: bit depth, alpha, chroma format, and a code for color space. Postprocessing verifies bit depth and color space. The HEVC encoder receives the verified image and then starts performing the splitting, intra frame prediction, DCT, IDCT, and quantization processes.

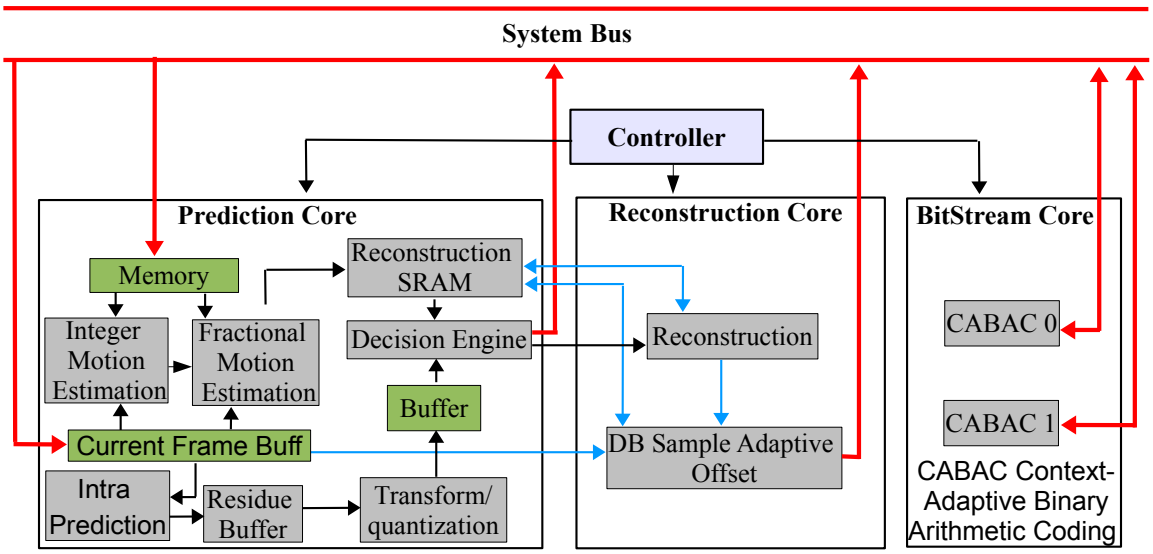


Fig. 18. HEVC Encoder System.

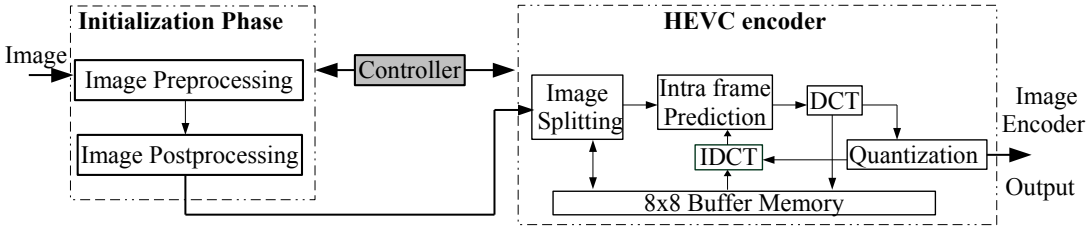


Fig. 19. System Level Architecture of the Proposed Algorithm.

1) Simulink® Based Modeling: The proposed algorithm is prototyped in MATLAB®/Simulink® Version 8.3 (R2014a), with the Computer Vision System Toolbox Version 9.7 [48]. The BPG encoder model is shown in Fig. 20. The methodology that is used to represent the high level system modeling is bottom-up. The first step is focused on building function units; the next step is to integrate these units into sub-systems; and finally, verifying and testing overall system functionality. MATLAB®/Simulink® offers image processing functions and modules that facilitate fast prototyping. Another advantage of using MATLAB®/Simulink® is the availability of function units such as DCT/IDCT and block processing. In addition, the system-level modeling can be accomplished using different modules: Color Conversion and DCT domain compression.

2) Validation of the BPG Encoder: Four standard test images were selected: Bear, IceClimb, Lena, and Wallpaper, with different spatial and frequency characteristics. The test images are encoded using the proposed BPG compression encoder. Describing the type and amount of degradation in reconstructed compressed images is considered a major concern in evaluating picture quality in image compression systems. It has been proven [49] that some measures of image quality correlate well for a given compression algorithm but they are not reliable for an evaluation across different algorithms. Thus, the most common measures of image quality were used in this paper. The mathematical expressions for Root Mean Squared Error (RMSE) given in Eqn. 7 [50] and Peak Signal to Noise Ratio (PSNR) given in Eqn. 8 [51] :

$$RMSE = \frac{1}{\sqrt{MN}} \sum_{j=1}^{M-1} \sum_{n=1}^{N} ||(I_O(i,j) - I_{O'}(i,j)||^2 \quad (7)$$

$$PSNR = 10 \log \left(\frac{(2^n - 1)^2}{MSE} \right) \quad (8)$$

$$= 10 \log \left(\frac{255^2}{MSE} \right), \quad (9)$$

where the image has dimensions $M \times N$, I_O is the original image and $I_{O'}$ is the compressed image.

The need of metrics that correlate well with human perception of quality justifies introducing additional quality assessments: Structural SIMilarity (SSIM), Multi-scale MSSIM, and Visual Information Fidelity (VIF). SSIM and MSSIM aim to measure image resolution and viewing condition. Uncompressed images are used as references and then the similarity between the compressed images and references is measured. Details for the calculation of SSIM and MSSIM are given in [52]. The quality assessment VIF utilizes mutual information between reference images and distorted images [53]. Table I summarizes the image quality assessments that have been used in this paper.

The test images and the corresponding BPG images are shown in Fig. 21, Fig. 22, Fig. 23, and Fig. 24. Table II

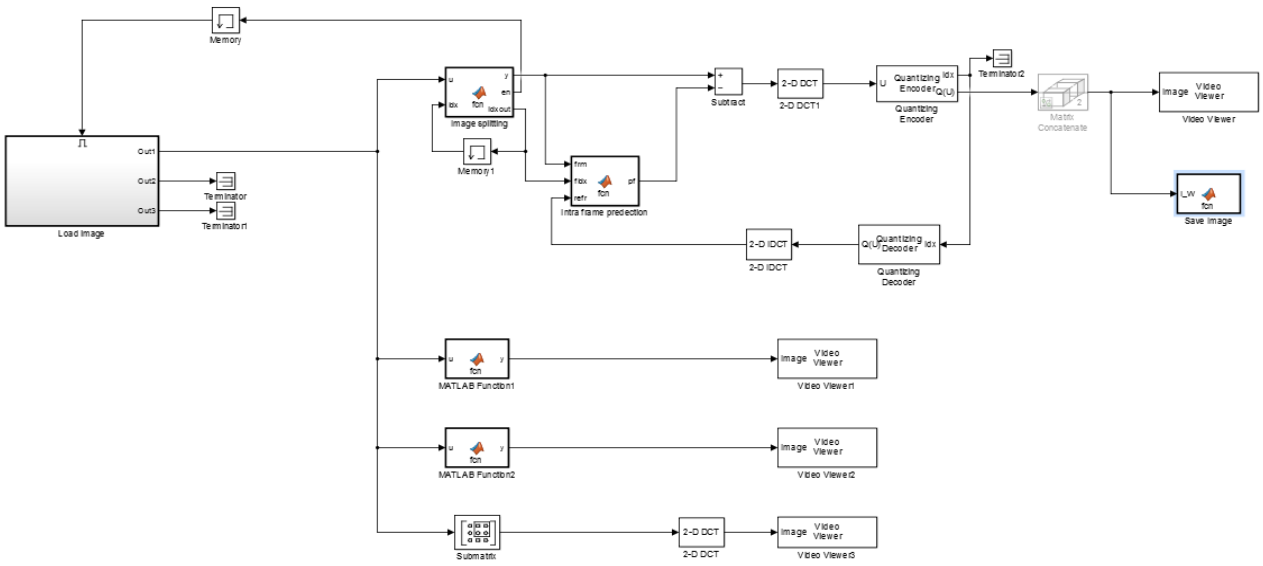


Fig. 20. BPG Compression Encoder in Simulink®.

illustrates the related metrics for each compression technique and test image. It is observed that for essentially the same PSNR, the size of the BPG image is substantially reduced. According to the experimental results, the proposed hardware architecture achieves better quality conforming to the quality assessments, shown in table II. Figures (25(a)), (25(b)), (25(c), 26(a)), (26(b)), (26(c)), 27(a)), (27(b)), (27(c)), 28(a)), (28(b)), and (28(c)) analyze and illustrate the related metrics for Bear, IceClimb, Lena, and Wallpaper images respectively.

TABLE I
SUMMARY OF QUALITY METRICS USED FOR THE COMPRESSION TECHNIQUE AND TEST IMAGE

Quality Metric	Remarks.
RMSE	The average squared difference, pixel-by-pixel.
PSNR	Luminance Component.
SSIM	Correlates with human perception: luminance, contrast, and structure.
MSSIM	Variance and cross-correlation.
VIF	Mutual information.

X. AUTONOMOUS CONTROL OF QUADROTOR

Autonomous control is achieved by means of a command line program running on the GCS computer. In order to start the program, the serial port path (at a minimum) needs to be passed to the program as well as the baud rate, target image and source device id. For example the program can be run as follows:

```
$ ./searchcopter -p
/dev/tty.usbserial-A600eGfb -b 57600 -t
/target.jpg -s 0
```

This will open a serial connection at 57600 baud with the specified USB serial device, load the target image and open the default video source. There are other user options that can be set at runtime; these are viewable along with their default values by printing the help menu with the “h” or “?” Flags. A basic flow diagram of the program can be seen in Fig. 29 which gives an overview of the program function.

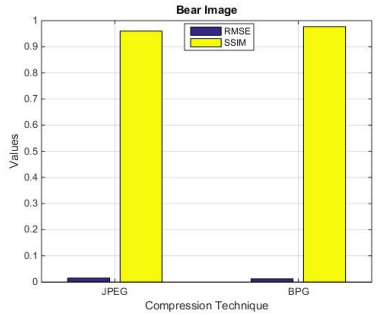
On launch, the user inputs are parsed and the default values of the other options are set. Next an attempt is made to open a serial connection with the specified port. If this is unsuccessful the programs exits with a failure status, otherwise the target image is loaded and video stream is opened. If they are loaded successfully the program enters the main loop and the target is searched for in the source image. If a match is found the location is displayed on the video stream and the required action is determined. The response is proportional to the distance to the target. The greater the distance, the faster the ArduCopter will move towards it. As it approaches the desired location it will slow down and finally stop when within the given region. Once the PWM values are calculated, a message is packed and sent to the ArducCopter. At the end of the loop there is a 1 msec delay to allow for keyboard actions to be detected. Currently “a” will arm the motors, “d” will disarm them and “spacebar” will hand control back to the RC. The program can be terminated by pressing either “ESC” or “q”. Several distinct communications protocols are involved and their interdependence is shown in Fig. 30.

XI. CONCLUSIONS AND FUTURE RESEARCH

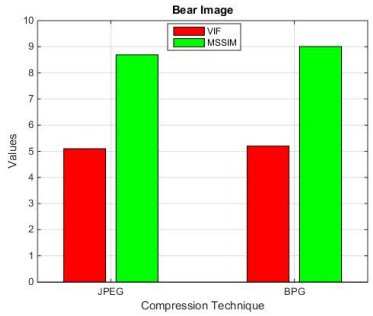
A versatile and extensible quadrotor platform, shown in Fig. 5, based on open-source hardware and software was designed and is described in detail in this paper. As an example application of the platform’s capabilities, a target

TABLE II
QUALITY METRICS FOR THE BPG COMPRESSION FOR TEST IMAGES.

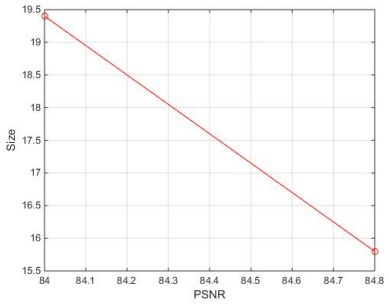
Test Image	Compression	Size (KB)	RMSE	SSIM	VIF	MSSIM	PSNR
Bear Image	JPEG (input image)	19.4	0.015	0.960	5.097	8.691	84.0
	BPG image	15.8	0.012	0.977	5.201	9.008	84.8
IceClimb Image	JPEG (input image)	85.3	0.012	0.990	5.293	9.092	86.0
	BPG image	78.4	0.010	0.999	5.890	9.537	86.1
Lena Image	JPEG (input image)	29.3	0.200	0.987	4.285	8.103	80.6
	BPG image	26.4	0.023	0.999	4.310	8.287	80.7
Wallpaper Image	JPEG (input image)	06.2	0.190	0.948	4.315	8.131	81.1
	BPG image	04.4	0.022	0.953	4.391	8.305	81.2



(a) RMSE & SSIM

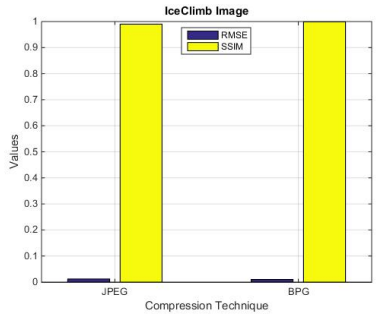


(b) VIF & MSSIM

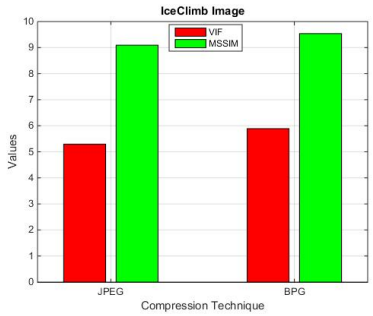


(c) Size versus PSNR

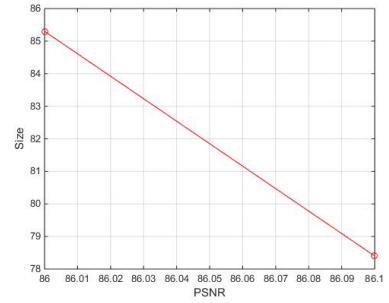
Fig. 25. Bear image



(a) RMSE & SSIM

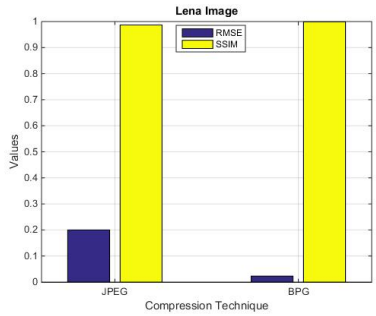


(b) VIF & MSSIM

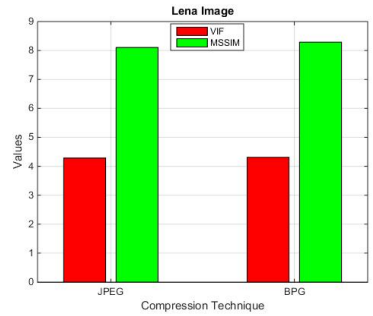


(c) Size versus PSNR

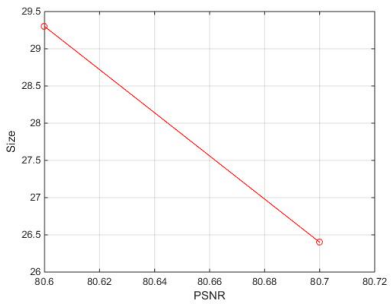
Fig. 26. ICeClimb image



(a) RMSE & SSIM

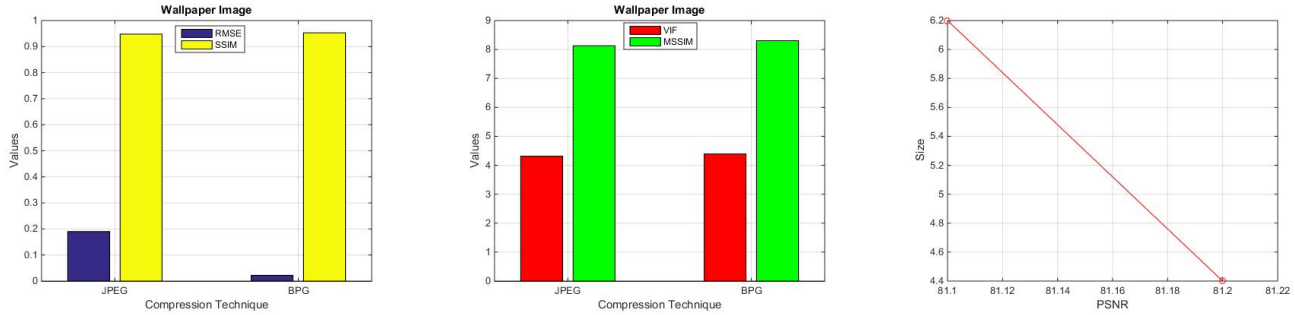


(b) VIF & MSSIM



(c) Size versus PSNR

Fig. 27. Lena image



(a) RMSE & SSIM (b) VIF & MSSIM (c) Size versus PSNR

Fig. 28. Wallpaper image

recognition system was designed, programmed and implemented using custom and published algorithms with outstanding performance. Research work is currently underway to extend the functionality of the vehicle by incorporating line-of-sight optimal communications location for search and rescue operations. Further work will concentrate on improving the on-board computing capabilities so that most of the computation burden is removed from the base station thus reducing the large amount of wireless traffic currently incurred. To securely transfer image and video data, the quodrotor architecture is equipped with an on-board secure digital camera (SDC) [54]. In such an situation, the quodrotor can be deployed to capture and transmit sensitive information reliably through the use of its on-board SDC. This is particularly important in certain applications in which a 3rd party can tamper with the data transmitted from the quodrotor.

A hardware architecture to perform BPG compression encoder in images is also presented. The encoding scheme can be divided into two phases. First is the initialization phase, which reads an image and extracts its details then verifies specific parameters such as bit depth, alpha, and color space. The second phase is HEVC encoding, which is considered a major advance in compression techniques. The proposed architecture is prototyped in Simulink®. The experimental results are compared with existing JPEG techniques in terms of quality and size and indicate the superior compression characteristics of BPG. Further work could include proposed hardware architecture as prototype using a hardware description language like Verilog and also building hardware in actual silicon. Also, since this paper only considers image compression, further work can be done on BPG video compression; the algorithm is clarified in Section IX. The BPG architecture will soon be integrated with encryption and or digital watermarking capabilities [48], [55]. Energy efficiency will remain a challenging problem for the battery operated systems [56], [57]. Future research directions also include developing energy-efficient as well as high-performance architectures which can be used in image or video communications in Internet of Things (IoT) frameworks.

ACKNOWLEDGMENTS

This research was supported in part by NSF award DUE-0942629. Preliminary results of this paper have been presented at the following conference papers: [58], [59].

REFERENCES

[1] M. Ingebretsen, "Where's My Personal Robot?" *IEEE Intelligent Systems*, vol. 24, no. 6, pp. 90–93, Nov 2009.

[2] Willow Garage, "PR2 robot for research and innovation," http://www.willowgarage.com/pages/pr2/overview.

[3] S. P. Mohanty, U. Choppali, and E. Kougianos, "Everything You wanted to Know about Smart Cities," *IEEE Consumer Electronics Magazine*, vol. 6, no. 3, July 2016.

[4] M. L. Rajaram, E. Kougianos, S. P. Mohanty, and U. Choppali, "Wireless Sensor Network Simulation Frameworks: A Tutorial Review," *IEEE Consumer Electronics Magazine*, vol. 6, no. 2, April 2016.

[5] S. P. Mohanty, *Nanoelectronic Mixed-Signal System Design*. New York, NY: McGraw-Hill Education, 2015, no. 9780071825719.

[6] S. P. Mohanty, E. Kougianos, W. Cai, and M. Ratnani, "VLSI Architectures of Perceptual Based Video Watermarking for Real-time Copyright Protection," in *Proceedings of the 10th International Symposium on Quality Electronic Design (ISQED)*, 2009, pp. 527 – 534.

[7] F. Bellard, "The BPG Image Format," http://bellard.org/bpg/, last Accessed on 09/20/2015.

[8] G. K. Wallace, "The JPEG Still Picture Compression Standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, Feb 1992.

[9] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, pp. 1649 – 1668, 2012.

[10] J. Jun, Q. Juntong, S. Dalei, and H. Jianda, "Control platform design and experiment of a quadrotor," in *Proceeding of 32nd Chinese Control Conference (CCC)*, 2013, pp. 2974 – 2979.

[11] S. Darma, J. L. Buessler, G. Hermann, J. P. Urban, and B. Kusumoputro, "Visual servoing quadrotor control in autonomous target search," in *Proceeding of IEEE 3rd International Conference on System Engineering and Technology (ICSET)*, 2013, pp. 319–324.

[12] S. Kim, D. Lee, and K. H. J, "Image based visual servoing for an autonomous quadrotor with adaptive backstepping control," in *Proceeding of 11th International Conference on Control, Automation and Systems*, 2011, pp. 532 – 537.

[13] A. Shoufan, "A hardware security module for quadrotor communication," in *Proceeding International Conference on Field-Programmable Technology (FPT)*, 2012, pp. 253 – 256.

[14] F. Senkul and E. Altug, "Modeling and control of a novel tilt Roll rotor quadrotor UAV," in *Proceeding of International Conference on Unmanned Aircraft Systems (ICUAS)*, 2013, pp. 1071 – 1076.

[15] N. D. Hao, B. Mohamed, and H. Rafaralahy, "Trajectory-tracking control design for an under-actuated quadrotor," in *Proceeding of European Control Conference (ECC)*, 2014, pp. 1765–1770.

[16] G. Crespo, G. G. de Rivera, and R. Ponticelli, "Setup of a communication and control systems of a quadrotor type Unmanned Aerial Vehicle," in *Proceeding Conference on Design of Circuits and Integrated Circuits (DCIS)*, 2014, pp. 1–6.

[17] A. Darji, A. N. Chandorkar, S. N. Merchant, and V. Mistry, "VLSI Architecture of DWT Based Watermark Encoder for Secure Still Digital Camera Design," in *Proceedings 3rd International Conference on Emerging Trends in Engineering and Technology (ICETET)*, 2010, pp. 760 – 764.

(a) JPEG Image (b) BPG Image

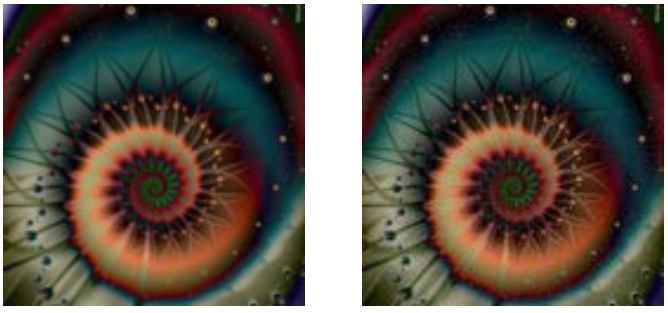
Fig. 21. Compression of Bear image (256×256).

(a) JPEG Image (b) BPG Image

Fig. 22. Compression of IceClimb image (512×512).

(a) JPEG Image (b) BPG Image

Fig. 23. Compression of Lena image (512×512).



(a) JPEG Image (b) BPG Image

Fig. 24. Compression of Wallpaper image (128×128).

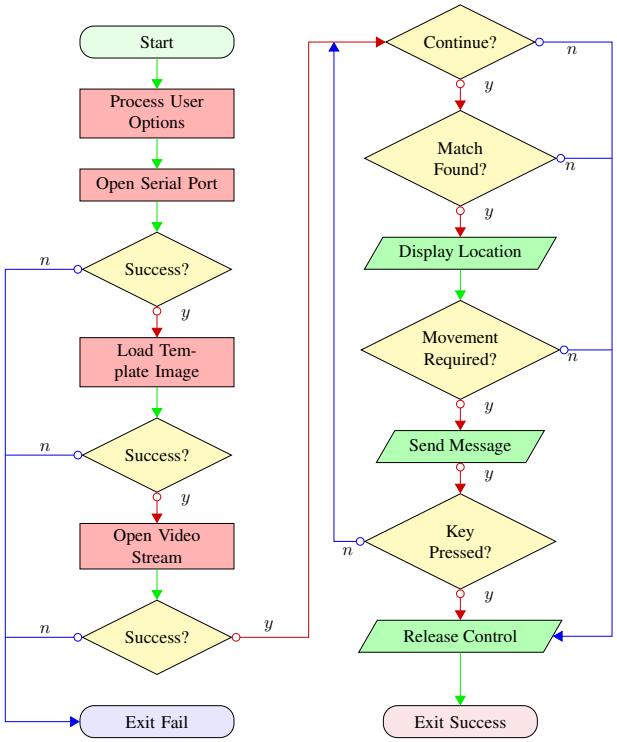


Fig. 29. Simplified flow chart of the control algorithm.

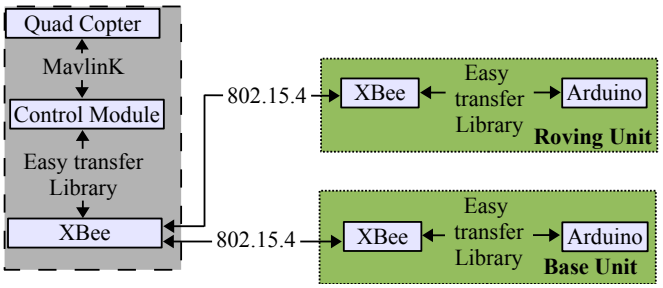


Fig. 30. Communication protocols interdependence.

[18] L. Tian and H. M. Tai, " Secure Images Captured by Digital Camera," in *Proceedings International Conference Consumer Electronics*, 2006, pp. 341 – 342.

[19] S. C. Ramesh and M. M. I. Majeed, "Implementation of visible watermarking in a secure still digital camera using VLSI design," in *Proceedings AFRICON*, 2009, pp. 1 – 4.

[20] K. Sarawadekar and S. Banerjee, "Area Efficient, High-speed VLSI Design for Ebcot Block Coder in JPEG 2000," in *Proceedings of International Conference on Electronics and Information Engineering (ICEIE)*, vol. 2, 2010, pp. V2–110 – V2–113.

[21] C. Liu, W. Shen, T. Ma, Y. Fan, and X. Zeng, "A Highly Pipelined VLSI Architecture for All Modes and Block Sizes Intra Prediction in HEVC Encoder," in *Proceeedings of IEEE 10th International Conference on ASIC (ASICON)*, 2013.

[22] D. Zhou, J. Zhou, W. Fei, and S. Goto, "Ultra-High-Throughput VLSI Architecture of H.265/HEVC CABAC Encoder for UHDTV Applications," in *Proceedings of IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, 2014, pp. 497 – 507.

[23] M. Mody, H. Garud, S. Nagori, and D. K. Mandal, "High Throughput VLSI Architecture for HEVC SAO Encoding for Ultra HDTV," in *Proceedings of IEEE International Symposium on Circuits and Systems*, 2014.

[24] M. Tarhan and E. Altuğ, "A Catadioptric and Pan-Tilt-Zoom Camera Pair Object Tracking System for UAVs," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1–4, pp. 119–134, January 2011.

[25] L. R. García Carrillo, E. Rondon, A. Sanchez, A. Dzul, and R. Lozano, "Stabilization and trajectory tracking of a quad-rotor using vision," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1–4, pp. 103–118, January 2011.

[26] E. Altuğ, J. P. Ostrowski, and C. J. Taylor, "Control of a Quadrotor

Helicopter Using Dual Camera Visual Feedback," *The International Journal of Robotics Research*, vol. 24, no. 5, pp. 329–341, May 2005.
[27] G. M. Hoffmann, H. Huang, S. L. Wasl, and C. J. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *In Proc. of the AIAA Guidance, Navigation, and Control Conference*, vol. 2, 2007, pp. 6461–6481.
[28] R. C. Leishman, J. C. MacDonald, R. W. Beard, and T. W. McLain, "Quadrotor and Accelerometers: State Estimation with an Improved Dynamic Model," *IEEE Control Systems*, vol. 34, no. 1, pp. 28–41, January 2014.
[29] R. Mahony, V. Kumar, and P. Corke, "Multirotor Aerial Vehicles: ModModel, Estimation, and Control of Quadrotor," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.
[30] R. Czyba, "Design of Attitude Control System for an UAV-type Quadrotor Based on Dynamic Contraction Method," in *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2009, pp. 644–649.
[31] B. Yu, Y. Zhang, I. Minchala, and Y. Qu, "Fault-tolerant Control With Linear Quadratic and Model Predictive Control Techniques Against Actuator Faults in a Quadrotor UAV," in *Proceedings of the Conference on Control and Fault-Tolerant Systems (SysTol)*, 2013, pp. 661–666.
[32] T. Ryan and H. J. Kim, "LMI-based gain synthesis for simple robust quadrotor control," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 4, pp. 1173–1178, 2013.
[33] "Arducopter - The Full-Featured Multicopter UAV!" http://code.google.com/p/arducopter/, Last accessed on 27 Sep 2015.
[34] "APMCopter," http://copter.ardupilot.com/, Last accessed on 27 Sep 2015.
[35] G. Coelho, "OTA-Quadrotor: An Object-Tracking Quadrotor for Real-Time Detection and Recognition," Master's thesis, Department of Engineering Technology, University of North Texas, Denton, TX, USA.
[36] "APM Mission Planner," http://code.google.com/p/ardupilot-mega/wiki/MissionPlanner, Last accessed on 27 Sep 2015.
[37] QGROUNDCONTROL, "MAVLink Air Vehicle Communication Protocol," http://qgroundcontrol.org/mavlink/start, Last accessed on 27 Sep 2015.
[38] I. Mondragón, M. A. Olivares-Méndez, C. Campoy, and L. Mejias, "Unmaned aerial vehicles UAVs attitude, height, motion estimation and control using visual systems," *Autonomous Robot*, vol. 29, no. 1, pp. 17–34, July 2010.
[39] D. Eberli, D. Scaramuzza, S. Weiss, and R. Siegwart, "Vision based position control for MAVs using one single circular landmark," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1–4, pp. 495–512, January 2011.
[40] "The OpenCV Reference Manual," http://docs.opencv.org/opencv2refman.pdf, Last accessed on 27 Sep 2015.
[41] OpenCV, "Fast Template Matching with Image Pyramid," https://opencv-code.com/tutorials/fast-template-matching-with-image-pyramid/, Last accessed on 27 Sep 2015.
[42] "Advanced Encryption Standard (AES)," Federal information processing standards publication 197, Tech. Rep., 2001. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
[43] M. U. K. Khan, M. Shafique, and J. Henkel, "Software Architecture of High Efficiency Video Coding for Many-Core System with Power-Efficient Workload Balancing," in *Proceedings Automation and Test Design in Europe Conference and Exhibition (DATE)*, 2014, pp. 1–6.
[44] C. C. Chi, M. Alvarez-Mesa, B. Juurlink, G. Clare, F. Henry, S. Pateux, and T. Schierl, "Parallel Scalability and Efficiency of HEVC Parallelization Approaches," *IEEE Transactions on Circuits and Systems for Video Technology.*, vol. 22, no. 12, pp. 1827 – 1838, 2012.
[45] M. Shafiquea, M. U. K. Khan, and J. Henkel, "Power Efficient and Workload Balanced Tiling For Parallelized High Efficiency Video Coding," in *Proceedings IEEE International Conference on Image Processing*, 2014, pp. 1253 – 1257.
[46] V. Sze, M. Budagavi, and G. J. Sullivan, Eds., *High Efficiency Video Coding (HEVC)*. Springer International Publishing, 2014.
[47] J. Wu, W. Dai, and H. Xiong, "Regional context model and dynamic Huffman binarization for adaptive entropy coding of multimedia," in *Proceedings IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2014, pp. 1 –6.
[48] S. P. Mohanty and E. Kougianos, "Real-Time Perceptual Watermarking Architectures For Video Broadcasting," *Elsevier Journal of Systems and Software (JSS)*, vol. 19, no. 12, pp. 724 – 738, 2011.
[49] M. Mrak, S. Grgic, and M. Grgic, "Picture Quality Measures in Image Compression System," in *Proceedings The IEEE Region 8 Computer as a Tool EUROCON*, 2003, pp. 233–236.
[50] C. J. Willmott and K. Matsuura, "Advantages of the Mean Absolute Error (MAE) over the Root Mean Square error (RMSE) in assessing average model performance," in *Proceedings Climate Research*, vol. 30, 2005, p. 79 82.
[51] Q. Huynh-Thu and M. Ghanbari, "Scope of Validity of PSNR in Image/Video Quality Assessment," in *Electronics Letters*, vol. 44, 2008, pp. 800 – 801.
[52] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, pp. 1057–7149, 2004.
[53] H. Sheikh, M. Sabir, and A. Bovik, "A Statistical Evaluation of Recent Full Reference Image Quality Assessment Algorithms," *IEEE Transactions on Image Processing*, vol. 15, pp. 1057–7149, 2006.
[54] S. P. Mohanty, "A Secure Digital Camera Architecture for Integrated Real-Time Digital Rights Management," *Journal of Systems Architecture*, vol. 55, no. 10-12, pp. 468–480, October 2009.
[55] N. M. Kosaraju, M. Varanasi, and S. P. Mohanty, "A High-Performance VLSI Architecture For Advanced Encryption Standard (AES) Algorithm," in *Proceedings of the 19th International Conference on VLSI Design*, 2006, pp. 481–484.
[56] S. P. Mohanty, N. Ranganathan, and S. K. Chappidi, "Peak Power Minimization through Datapath Scheduling," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*. IEEE, 2003, pp. 121–126.
[57] S. P. Mohanty, N. Ranganathan, and V. Krishna, "Datapath Scheduling using Dynamic Frequency Clocking," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*. IEEE, 2002, pp. 58–63.
[58] G. Coelho, E. Kougianos, S. P. Mohanty, P. Sundaravadivel, and U. Albalawi, "An IoT-Enabled Modular Quadrotor Architecture for Real-Time Aerial Object Tracking," in *Proceedings of the 1st IEEE International Symposium on Nanoelectronic and Information Systems*, 2015, pp. 197–202.
[59] U. Albalawi, S. P. Mohanty, and E. Kougianos, "A Hardware Architecture for Better Portable Graphics (BPG) Compression Encoder," in *Proceedings of the 1st IEEE International Symposium on Nanoelectronic and Information Systems*, 2015, pp. 291–296.