Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 4, No. 1, April 2018.

UNMANNED GROUND VEHICLE WITH VIRTUAL REALITY VISION

Mahmood Al-Khalil, Rami Abu-Rhayyem, Ahmad Hammoudeh and Talal A. Edwan*

(Received: 21-Nov.-2017, Revised: 8-Feb.-2018, Accepted: 3-Mar.-2018)

ABSTRACT

This paper aims to describe the design and implementation of an Unmanned Ground Vehicle (UGV) and a smart phone virtual reality (VR) head mounted display (HMD) which enables visual situation awareness by giving the operator the feel of "head on rover" while sending the video feeds to separate operator computer for object detection and 3-D model creation of the UGV surrounding objects. The main contribution of this paper is of three folds: (i) the novel design of the HMD; the paper proposes an alternative design to the 3-D interface designs recently used in tele-operated search and rescue (SAR) UGVs. Unlike other designs that suggest to automatically move the whole UGV about two axes (pitch and yaw) with the movement of the head, this design suggests to let a separate unit of the UGV automatically move with the movement of the head and provide the user with VR. (ii) the distributed feature; the design allows multiple users to connect to the UGV using a wireless link in a secure way to receive video feeds from three on-board cameras. This feature facilitates cooperative team work in urban search and rescue (USAR) applications (a contemporary research issue in SAR UGV). (iii) a novel feature of the design is the simultaneous video feeds which are sent to the operator station computer for object detection using the scale-invariant feature transform (SIFT) algorithm and 3-D model construction of the UGV's surrounding objects from 2-D images of these objects. The design was realized using a smart phone-based HMD, which captures head movements in real time using its inertial measurement unit (IMU) and transmits it to three motors mounted on a rover to provide the movement about three axes (pitch, yaw and roll). The operator controls the motors via the HMD or a gamepad. Three on-board cameras provide video feeds which are transmitted to the HMD and operator computer. A software performs object detection and builds a 3-D model from the captured 2-D images. The realistic design constraints were identified, then the hardware/software functions that meet the constraints were listed. The UGV was implemented in a laboratory environment. It was tested over soft and rough terrain. Results showed that the UGV has higher visual-inspection capabilities compared to other existing SAR UGVs. Furthermore, it was found that the maximum speed of 3.3 m/s, six-wheel differential-drive chassis and spiked air-filled rubber tires of the rover gave it high manoeuvrability in open rough terrain compared to other SAR UGVs found in literature. The high visual inspection capabilities and relatively high speed of the UGV make it a good choice for planetary exploration and military reconnaissance. The three-motors and stereoscopic camera can be easily mounted as a separate unit on a chassis that uses different locomotion mechanism (e.g. leg type or tracked type) to extend the functionality of a SAR UGV. The design can be used in building disparity maps and in constructing 3-D models, or in real time face recognition, real time object detection and autonomous driving based on disparity maps.

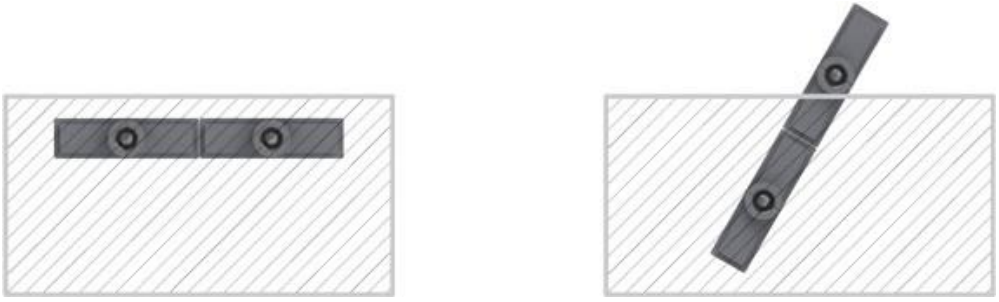
KEYWORDS

UGV, Virtual reality, Search and rescue, Robotics, Human-robot interaction.

1. INTRODUCTION

In tele-operated SAR UGV, the visual perception of the UGV environment and its presentation to the operator has a deep impact on human-robot-interaction (HRI) awareness of the UGV environment [1]-[2]. Many tele-operated SAR UGV designs rely on gathering as much data as possible from the UGV's surrounding via sensors and transferring this along with a video feed from an on-board camera to the operator's base station using wired and/or wireless communication links to enhance the cognitive ability of the UGV. [3]-[7]. The data is usually presented to the operator using a 2-D screen, hence only a proportion of the screen is used for video and the rest is used for displaying the data collected by the sensors in a user-friendly way. For example, in RAPOSA [8], only 29% of the total screen is

M. Al-Khalil, R. Abu-Rhayyem and A. Hammoudeh are students at Department of Computer Engineering, Princess Sumaya University for Technology, Amman, Jordan.
*Corresponding author: Talal A. Edwan is with Department of Computer Engineering, Princess Sumaya University for Technology, Amman, Jordan. Email: t.edwan@psut.edu.jo.



(a) Two cameras behind a barrier

(b) After rotating by 60 degrees, one camera is on top of the barrier

Figure 1. Two-camera design, where software rotation of images is not suitable.

used for video display. In addition to that, in many designs, if the operator wants to visually inspect the environment from a different viewpoint, he/she has to move the whole UGV. Similarly, Adora [9] uses a sensor box with single camera all mounted on a movable arm. More screen area was dedicated for the video feed, but some is reserved for viewing data collected from sensors. It has been found that a dichotomy in this type of research projects exists: either concentrating on conducting research on UGV mobility in rough terrain or increasing the cognitive/intelligent behaviour of the robotic assets, but rarely on both research domains [10].

Some implementations of user interfaces mitigated the problem by having a command for hiding all the clutter in the interface, leaving simply a clear cut view from the UGV visual output. The point of this is to quickly reduce sensory overload during complex situations [1]. Other implementations adopted a 3-D interface, for example, an attempt was made to increase the visual perception in RAPOSA using a 3-D interface [11]. This interface design is based on an HMD equipped with a head tracker; the HMD displays the images from a pair of video cameras located in the UGV frontal body, where the video stream of each camera is displayed to each operator eye. Because these two video feeds pertain to two slightly different viewpoints, it is possible with image rectification to endow the operator with depth perception (stereopsis). The pitch angle is used to control the UGV frontal body up/down, the yaw angle is used to rotate the UGV and the roll angle is used to rotate the images (the HMD has to counter-rotate the images to compensate for head movement in this direction). The shortcoming of this approach in our opinion is of two folds: first, the HMD highly depends on the type of the UGV in use, and second, is the inability to compensate for head movement in the roll direction by software-rotating the images in case more than one camera is used. To illustrate the latter point, we consider the situation depicted in Figure 1. a, where two cameras are initially behind a barrier, then after rotating by 60 degrees as shown in Figure 1. b, one of the cameras is positioned on top of the barrier and thus can provide a different view. We therefore conclude that the software compensation approach in [11] is suitable for a single camera, but not for multiple-camera design.

This paper proposes an alternative design to the 3-D interface designs recently used in tele-operated search and rescue (SAR) UGVs. Unlike other designs that suggest to automatically move the whole UGV about two axes (pitch and yaw) with the movement of the head, this design suggests to let a separate unit of the UGV automatically move with the movement of the head. A smart phone-based headset with VR capability captures head movements in real time using the built-in IMU in the smart phone and transmits it to three motors mounted on the same vertical axis on a six-wheel rover. The three motors are arranged to allow free movement about three axes (pitch, yaw and roll). The motors' vertical axis is equipped with a stereoscopic camera (i.e. two cameras separated by a distance) with the motors, constituting one separate unit, which captures video and transmits it in real time back to the headset. This gives the operator the flexibility to control the UGV view direction by just moving his/her head in the desired direction. We refer to this mode of operation as "automatic mode", in contrast to the "manual mode", where the operator can control the motors by a gamepad. The operator can switch between the two modes by pressing a button on the gamepad. A third camera in the midpoint between the two cameras is used to capture pictures on-demand by pressing a certain button on the gamepad. The resultant pictures are saved on an on-board SD-card and can be transmitted using a wireless connection to the operator's computer. Two special computer programs run on the

operator's computer: one is for object detection using the SIFT algorithm and the other is for 3-D model creation from the 2-D captured pictures of the UGV's surrounding objects. Further, the operator can control the six-wheel rover based on the visual feedback using a gamepad. The six-wheel rover offers relatively high climbing capabilities and performes very well over soft and rough terrain. The design allows multiple users to connect to the UGV in a secure way and receive the video feed from the cameras which facilitate cooperative team work in USAR applications.

The rest of the paper is divided as follows: the section "Design Requirements" describes the required specifications of the UGV; the section "Abridged Design" describes in a concise way the overall design; the section "Mechanical Design & Control" gives details about the hardware of the UGV, including the 3-axis rotating platform and chassis and how they are electrically controlled; the section "Stereoscopic Camera Set & Pi camera" describes the set-up of the cameras and their functions and means of communication with the operator station; the section "Data Processing Unit & Video Transmission" describes the hardware of the UGV's on-board processing unit and the software used to transmit the video to the operator station; the section "VR Headset" gives details about the HMD; whereas the section "Operator's Station" describes the operator station computer and the programs that run on it to handle the image processing; finally the section "Conclusion and Future Work" concludes the paper and gives an outline about future work.

(a) Basic design concept

(b) Main components

(c) Climbing

(d) On tough terrain

Figure 2. The UGV.

2. DESIGN REQUIREMENTS

Our work aims to achieve a set of design requirements. A brief description of each design requirement is provided below:

(i) Build an UGV able to hold a minimum weight of 3 kilograms: for a UGV to be able to carry heavy weights, its wheels and suspension systems need to be flexible enough so that they do not break if a heavy weight is applied on them. Also, the chassis needs to be rigid enough for not to break or bend due to heavy weight.

(ii) Build a UGV that can move at an approximate speed of 4 m/s: to be able to achieve this

requiremet, the UGV motors need to have enough torque to rotate the wheels that carry the UGV body and the other components that are placed on the UGV. Also, the motors are required to rotate at an approximate speed of 4 m/s while a load is applied on their shafts. To achieve such a torque, high power motors with gear reduction ratio or motors that deliver high torque and decent rotation speed are needed.

(iii) Build a 3-axis rotating platform that is able to rotate in the three Euler angles (Yaw, Pitch and Roll): the 3-axis rotating platform must have two cameras mounted on it to simulate human eyes and a third special purpose camera. The average total weight of cameras must be approximately 120 grams; high-precision motors must be used in order to rotate at a speed close to normal human-head rotating speed.

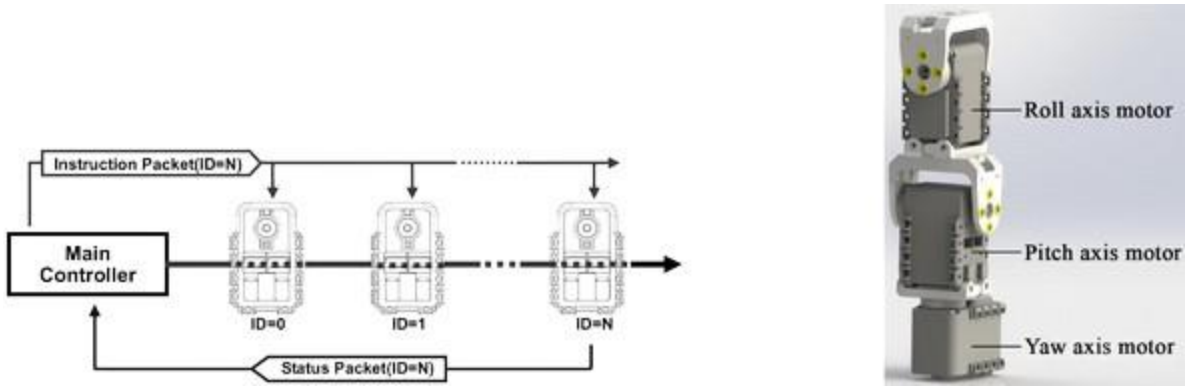
(iv) Transmit video feed from the UGV to a smartphone, with a transmission delay less than 1 second: wireless transmission is to be used to transmit video to the head set and operator's station, fast transmission to the headset is crucial to maintain synchronization between head movement and video feed. This requires the use of a wireless link with high date rates, large bandwidth, low latency and low interference.

(v) Operational radial distance from the operator's station is 50 meters: this requires a wireless link of low latency, high data rate and high signal power to operate efficiently over the required distance.

(vi) Minimum UGV trip time of 15 minutes: this requirement depends totally on the power source, the power supply must deliver enough current and voltage to drive all of the on-board circuitry and motors for the required operational time.

3. ABRIDGED DESIGN

The developed system consists of three subsystems: the operator's station, the VR headset and the UGV, all connected via an access point as depicted in Figure 2. a. The operator's station consists of a laptop and a gamepad, while the VR headset consists of a smart phone. Each of these two subsystems runs different software capable of controlling two different parts of the UGV. The laptop software reads the control data from a gamepad; this data can be used to simultaneously drive the UGV and move the stereoscopic camera by moving three control motors which constitute a 3-axis rotating platform. The smart phone (VR headset) runs different software which tracks head motion using the smart phone's IMU and transmits it to the UGV to control the 3-axis rotating platform. The operator can seamlessly switch between the manual mode (using the gamepad) and the automatic mode (using VR headset) to control the 3-axis rotating platform.



(a) Communication protocol overview (b) Assembly of the 3-axis rotating platform

Figure 3. 3-axis rotating platform.

Two video feeds are provided from the stereoscopic camera and Pi camera; the stereoscopic-camera video feed goes to the smart phone which in turn displays it to the operator using a VR software and to the operator's station for object detection, whereas the Pi camera video feed goes to the operator's station for 3-D model construction. The subsystems communicate using managed Wi-Fi (IEEE 802.11 standards). Network programming using Python and Java programming languages was used to create transmission control protocol (TCP) connections between the operator's station and the UGV on one

side and between the VR-headset and the UGV on the other side. Figure 2. b shows the main components of the UGV and Figures 2. c and 2. d show the UGV in real test over tough terrain.

4. MECHANICAL DESIGN & CONTROL

4.1 3-AXIS ROTATING PLATFORM

The 3-axis rotating platform is an essential part of the UGV; it mimics the human head movements. The platform was implemented using Dynamixel AX-12 actuator which consists of a gear reducer (to produce high torque), a high-precision DC motor and an internal controller for networking functionality. Table 1 shows the specifications of Dynamixel AX-12 actuator.

Table 1. Dynamixel AX-12 actuator specifications.

	Dynamixel AX-12
Weight (g)	55
Gear Reduction Ratio	1/254
Input Voltage	7V or 10V
Final Max Holding Torque (kgf.cm)	12 @7V and 16.5 @10V
Sec/60 degree	0.269 @7V and 0.196 @10V

The actuator has the ability to detect internal conditions, such as excessive voltage and internal temperature, using a built-in controller. Communication with the actuator is achieved by means of half duplex asynchronous serial communication channel, which uses only one wire for both transmitting and receiving the data. The size of a serial frame is 9-bits; 8-bits for data and one bit for parity. To control the Dynamixel actuator, a packet of bytes is sent through the half-duplex serial channel which contains the control command; this packet is called the "instruction packet". After the actuator's controller receives the packet, it performs the command the packet contains, afterwards, the actuator sends back a feedback packet called the "status packet", which contains either an error flag if an error occurred or data from the controllers' registers. An overview of the communication protocol is

(a)Schematic diagram of the actuators, powering source of the USB2AX device and
(b) Implementation of the 3-axis rotating platform with the stereoscopic camera mounted on it.

Figure 4. Design and implementation of the 3-axis rotating platform.

illustrated in Figure 3. a. In order to let the 3-axis rotating platform rotate in the three Euler rotation angles (Yaw, Pitch and Roll), the three Dynamixel AX-12 actuators were assembled as shown in Figure 3. b. Each actuator was given a unique ID which was written on the register of address 0x03 using the broadcast ID instruction; this was separately done for each actuator. The format of the instruction packet is as follows:

0xFF 0xFF ID LENGTH INSTRUCTION PARAMETER 1 . . . PARAMETER N CHECKSUM

- 0xFF: These two bytes indicate the beginning of an instruction packet.

- ID: This byte contains the unique ID of a Dynamixel actuator in the network; there are only 254 available ID values.

A broadcast ID is the when transmitted ID value is equal to 0xFE, which means that any instruction packet with this ID gets transmitted to all actuators connected in the network, and the actuators do not return a status packet.

- Length: This byte hold the length of packet; its value is calculated as (# of parameters + 2).

- Instruction: This byte contains the instruction that the actuator is required to perform; it can hold one of these values.

- Parameters: These bytes contain the parameters needed for the instruction (e.g.: Angle, Torque, ...).

- Checksum: A Check Sum byte is added at the end of the instruction packet to insure integrity of data being delivered. The check sum value is calculated as follows.

Checksum = ~[ID + Length + Instruction + Parameter1 + … + ParameterN]

where (~) represents the NOT logic operation.

The instruction packet sent to each actuator was of the following format:

0xFF 0xFF 0xFE 0x04 0x03 0x03 0x01 0xF6

- 0xFF: These two bytes indicate the beginning of an instruction word.

- 0xFE: This ID is the broadcast ID defined above and is used to send instruction packets to all the connected Dynamixel actuators in the network regardless of their ID values.

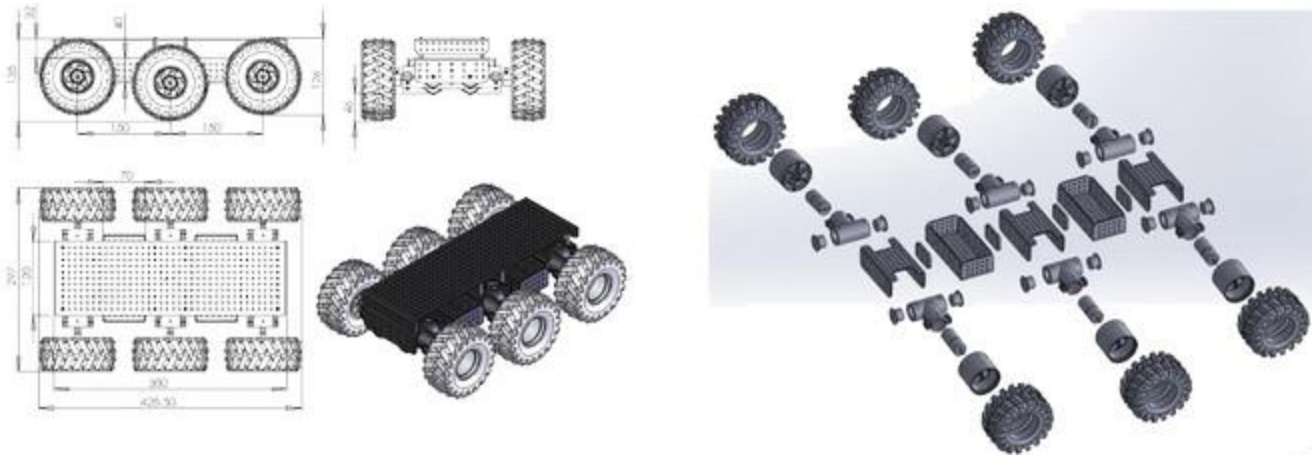
- 0x04: This byte represents the length of the instruction packet (2 parameters + 2 = 4).

- 0x03: This byte indicates a WRITE DATA instruction.

- 0x03: This byte represents the first parameter which is the address of the wanted registers for this packet. In this case it is the ID register.

- 0x01: This byte represents the second parameter which is the data to be written in the desired register (0x01, 0x02 and 0x03), respectively, for each actuator.

- 0xF6: This byte represents the checksum for the instruction packet.



(a) Chassis dimensions (b) Chassis dissemble view

Figure 5. Six-wheel rover model.

The on-board processing unit is a load-balancing cluster of two Raspberry Pi 3 model B: #1 Pi and #2 Pi. The actuators are connected to #2 Pi using a half-duplex asynchronous USB-to-TTL module (Xevelabs mini USB2AX v3.2a). The schematic diagram of the connection is illustrated in Figure 4.

(a) and the implementation in Figure 4. (b). Since the processing unit has native support for Python, the library Pyax12 was used in conjunction with Python's native socket library to continuously construct instruction packets and send them to the 3-Axis rotating platform actuators.

The processing unit has two modes for operating the 3-axis rotating platform; VR-headset mode and manual mode, (i) VR-headset mode: The processing unit uses the data extracted from the movement of the operator's head in the operator's station to rotate the actuators accordingly; this mode is the default mode. (ii) Manual mode: The processing unit uses the data received from the gamepad at the operator's station to rotate the actuators accordingly.

The processing unit initializes the serial port of the Xevelabs USB2AX device, with a baud rate of 1 Mbps. Then, it sends an instruction word to all the actuators to set the compliance slope value to 0x92, in order to avoid damage to the actuator's shafts due to continuous change in direction. The processing unit then initializes two network sockets: one for the VR mode and one for the manual mode. The sockets then listen for incoming connections. Once a connection has been established between the operator's station or VR headset and the UGV, the UGV processing unit begins receiving data (Yaw, Pitch and Roll values) via wireless link continuously as a stream of 12 bytes for each frame, then it reconstructs the Yaw, Pitch and Roll integer values using the native "struct" library, where each integer value consists of 4 bytes. Afterwards, the processing unit constructs an instruction packet for each of the Yaw, Pitch and Roll values and transmits each packet to its corresponding actuator on the 3-axis platform. If the operator's station sends a switch mode command via the wireless link by pressing a certain button on the gamepad, the processing unit stops constructing instruction packets from the Yaw, Pitch and Roll values and starts using the data coming from the gamepad of the operator's station instead. Sending another switch mode command toggles the operation mode again.

The 3-axis rotation platform is powered by the main power source of the UGV, which is a LiPo battery. The LiPo battery has a full-charge voltage of 13.4 volts. This voltage cannot be directly connected to the actuators because the operational voltage range of the actuators is 7-12 volts; therefore, a variable-voltage regulator was used to regulate the voltage of the LiPo battery to 10.5 volts so that the actuators can operate properly.

4.2 UGV Chassis

The locomotion mechanism used in the UGV is enhanced wheel type; particularly, a six-wheel driven rover with a differential-drive chassis. Vehicle-turning is accomplished by driving the motors on the two sides of the platform at different directions and speeds. For example, to rotate the rover around itself in counter-clockwise direction, the right-side tires should be moved clockwise and the left-side tires should be moved counter-clockwise. The wheels are spiked-rubber tires filled with air. The rover is "6-Wheels Wild Thumper", from Pololu [12]; the driving motors included in the chassis have a gear reduction ratio of 34:1, which provides the required speed and torque. Also, every two opposing tires are connected with a unique "super-twist" suspension system which gives the rover climbing capability. Figure 5. (a) shows the rover's dimensions and Figure 5. (b) shows a disassemble view of its 3-D model. The main specifications of the rover are listed in Table 2.

We added six DC motors; three motors on each side controlled by two H-bridge module, one for each side. The rover's controller (an Arduino UNO board) is connected to # 2 Pi by an USB-to-TTL module (CH340G). The Arduino UNO board is programmed to receive two bytes from #2 Pi each time the operator moves the joystick of the gamepad; the first byte defines the direction of the rover's movement according to Table 3, where the second byte defines the speed of the rover's movement which is a value from 0 to 9; a value of 0 means the rover is stopped and a value of 9 means that highest speed in the specified direction. Speed data is dictated to the board from # 2 Pi; accordingly, the board controls the speed of the motors using Pulse Width Modulation (PWM); that is, by adjusting the duty cycle.

The Arduino UNO board first initializes serial communication port with a baud rate of 9600 bits per second, then initializes the PWM duty cycle of all motors to zero. Once data is available on the serial communication channel, the Arduino reads two bytes. Then, the Arduino starts increasing/decreasing the PWM duty cycle factor with value of one each three milliseconds. In this way, the Arduino UNO keeps the motors safe from current spikes and instant torque on the gearbox of the motors shaft.

Table 2. UGV rover specifications.

Size	420 x 300 x 130 mm (16.5" x 12" x 5").
Weight	2.7 kg (6.0 lb) (including wheels and motors only).
Max. payload	5 kg (11 lb).
Motor voltage	2 -- 7.5 V.
Stall current	6.6 A per motor.
No-load current	420 mA per motor.
No-load speed	350 RPM for the 34:1 gear reduction motors.
Stall torque at 7.2V	5 kg-cm (70 oz-in) per motor for the 34:1 gear reduction motors.

Table 3. Bytes sent to the rover's microcontroller.

First byte	Corresponding direction
0x00	Forward
0x01	Backward
0x02	Left
0x03	Right

Two H-bridge modules (two IBT-2 H-bridge modules each having two In neon BTS7960B half bridges) were used; each H-bridge controls three wheels on the same side. The module has a large heat sink and can withstand currents up to 43 amperes, with a maximum drive voltage of 27 volts. The module also has a built-in voltage regulator that provides 5 volts for the user to use in other applications. However, since the motors require 7.2 volts to operate, a DC-DC voltage step down converter was used to convert the battery's 12.4 volts (full-charge voltage) to 7.2 volts.

5. STEREOSCOPIC CAMERA SET & PI CAMERA

(a) Stereoscopic camera model, (b) Actual developed design of the stereoscopic camera set

Figure 6. Stereoscopic camera.

Two types of camera were used: a stereoscopic camera (Genius F100 widecam 12-megapixel sensor) and a Pi camera. The former is used to get video feed of the UGV point of view (POV), while the latter is used to take high-resolution pictures used to build a 3-D model of the rover's surrounding objects.

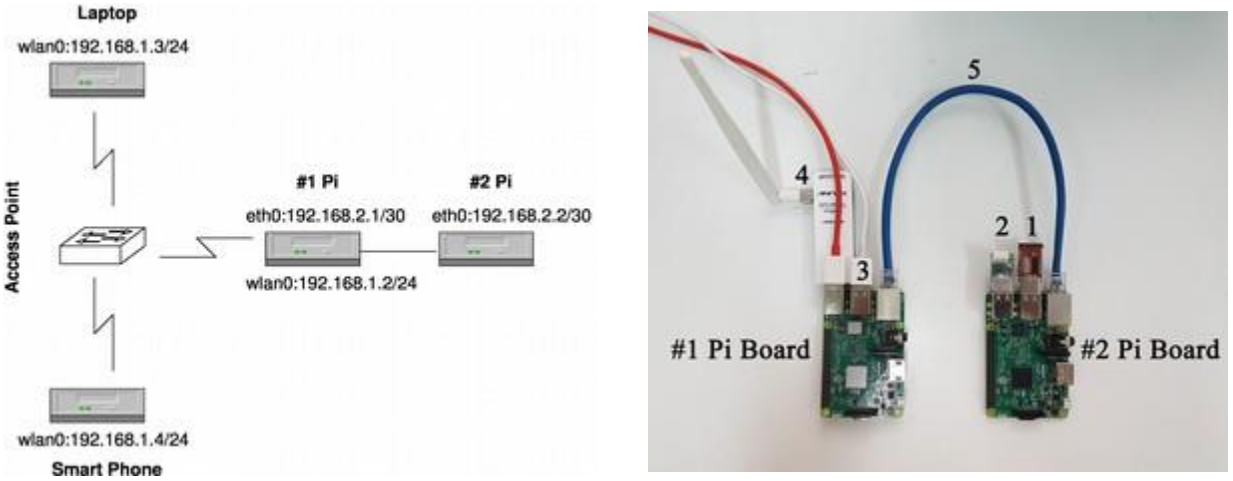
The stereoscopic camera set consists of two cameras aligned horizontally and separated by a distance of 10.5 cm measured from the centre of each lens as shown in Figure 6. (a). The distance between human eyes is usually referred to as the "Pupillary Distance"; the average value of this distance is 6.5 cm. However, larger distance results in more depth that can be felt when using the VR application. The stereoscopic camera is mounted on a "Plexiglass" plate that has a thickness of 4 mm, a length of 15.5 cm and a width of 5.5 cm. The plexiglass plate is mounted on the 3-axis rotating platform as depicted in Figure 6. (b). The plexiglass plate can be seen underneath the cameras. The two cameras were chosen mainly for their wide-angle lens, high-resolution and manual focus lens. These three specifications are crucial for any VR application. Table 4 lists the specifications of the cameras. Serial communication is achieved via a USB-2 cable connected to # 1 Pi.

The UGV is equipped with additional camera (Pi Camera Module v2 based on Sony IMX219 8-megapixel sensor); this camera is located at the mid-point between the two other cameras (see Figure

Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 4, No. 1, April 2018.

Table 4. Stereoscopic camera specifications.

	Genius F100 WideCam
Image sensor	1080p full HD pixel CMOS
Focus type	Manual focus lens
Interface type	USB-2
View angle	120 degrees
Body weight (g)	82



(a) Network set-up (b) Raspberry Pi boards' cluster set-up

Figure 7. Network & cluster set-up.

2. (b)) and connected to #1 Pi using the on-board 15-pin camera serial interface (CSI-2). The camera is activated in the manual mode and is used to capture pictures for the purpose of creating a 3-D model of the UGV surrounding objects. The Python script which controls the 3-axis rotating platform is also used to capture the pictures using the Pi Camera. This is done when the operator presses the manual mode button on the gamepad then presses the capture button on the gamepad. If the operator did so, each captured picture is saved as an image le in a folder stored in an on-board storage SD card on #1 Pi board. To retrieve the captured images, a secure le transfer protocol (SFTP) server was created on #1 Pi board, where the operator can access this SFTP server by using an SFTP client software on the operator-station laptop. After the operator accesses the SFTP server, he/she can copy the images to the operator-station laptop and pass the images to dedicated software (RealityCapture) to process the images and create a 3-D model of the UGV surrounding objects. It is worth to emphasise that each camera has a separate TCP connection to the operator's laptop. that is, the stereoscopic camera has a dedicated connection to TCP port number "41222" at the laptop, while the Pi camera connects to TCP port number 22 (SFTP). Two feeds from the stereoscopic camera are sent to the HMD; one per eye. There are lenses which are placed between the operator's eyes and the screen of the smart phone. These lenses focus and reshape the image for each eye and create a stereoscopic 3D image by angling the two 2D images to mimic how each of the operator's two eyes views the UGV surrounding.

6. DATA PROCESSING UNIT & VIDEO TRANSMISSION

The data processing unit consists of a cluster of two Raspberry Pi 3 model B boards connected back-to -back using a tethered Ethernet link. As shown in Figure 7. (b), a number of devices are attached to the two boards:

1. USB-to-TTL device, which communicates with the UGV chassis microcontroller.
2. USB-to-TTL device, which communicates with the internal microcontrollers of each motor in the 3-axis rotating platform.
3. USB cable of the stereoscopic camera set.
4. USB Wi-Fi dongle, which connects #1 Pi board to the access point.
5. The Ethernet cable, which connects the Pi boards together.

For a general overview of the network set-up, a network diagram was drawn in Figure 7. (a) to aid the illustration in this section. The #1 Pi board was programmed using Java programming language and utilizing OpenCV library to transmit two video feeds; one from the stereoscopic camera to the VR headset and another from the Pi camera to the operator-station laptop. In video transmission, we can tolerate a certain amount of packet loss for low delays, that is no need for retransmission which increases delay. For this reason, UDP or RTP (unreliable data transfer) are usually used instead of TCP (reliable data transfer) when transmitting audio and/or video. Delay and jitter are the important factors when transmitting audio/video. However, in our case we used an optimised version of TCP congestion control algorithm (TCP-BBR [21]) that has low latency, yet insures reliable data transfer. TCP was used as a transmission protocol for reliable data transfer of both control commands – where reliable data transfer is necessary – and video. Inthis way, one connection and one protocol were used for data transfer. Figure 8 shows a flowchart of the main thread which runs on the #1 Pi.

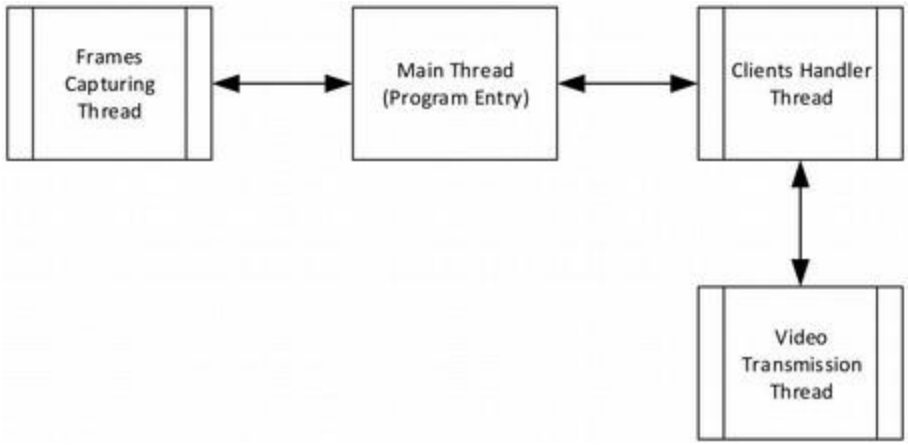


Figure 8. Flowchart of the main thread which runs on #1 Pi.

The main thread starts another two threads: the "client's handler" thread and the frames' "capturing thread". Another thread called the "video transmission thread", however, is started by the client's handler thread. When the main thread starts, it creates data containers for the compressed cameras' frames, then it starts the capturing thread and the client's handler thread. Finally, it stays in wait mode until the program is terminated.

When the capturing thread starts, it loads the OpenCV library, then it initializes the stereoscopic camera set by setting the resolution of the camera frames to 640x480 pixels. This resolution is relatively low and provides the required frame rate. To illustrate, the Raspberry Pi CPU, can process a maximum of 15 frames/second at this resolution. At higher resolutions, the frame-processing rate drops drastically. After initialization, the CPU starts receiving frames from the stereoscopic camera set. Eeach time it receives a frame, it checks the size of the frame, then it compresses the captured frame using JPEG lossy compression with a compression factor of 78%. Incorrect frame size results in the frame being dropped. Figure 9. a shows the flowchart of this thread.

The client's handler thread runs concurrently with the capturing thread, yet the CPU time is shared between the two threads. The client's handler thread is responsible for initializing a server on the #1 Pi, which accepts connections from the VR headset and the operator's station. The flow chart of the thread is shown in Figure 9. b. When the thread starts, it initializes a server socket and binds it to port number "41222", then it keeps listening on this port for incoming connections from the VR headset and/or the operator's laptop. Any new connection is passed to the video transmission thread. It is worth noting that the design can allow multiple VR headsets or operator laptops to connect and receive video feeds from the cameras, which facilitates cooperative team work in SAR operations.

The video transmission thread transmits the compressed frames by first checking whether the compressed frame's container is accessible; that is because the container is also accessible by the capturing thread and it cannot be accessed simultaneously by two threads. If the container is accessible, the thread further checks whether the container is empty or not. In case it is empty, no data

Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 4, No. 1, April 2018.

is sent. On the other hand, if the container is not empty, the thread writes the frame's bytes on the socket output stream. The #2 Pi board was programmed to deliver commands to all on-board microcontrollers through the USB-to-TTL serial interfaces.

7. VR HEADSET

The VR headset consists of two parts: the VR glasses (Arealer virtual reality glasses headset) and the smart mobile phone (Samsung S7 edge). The smart phone is used to receive video feed from the UGV stereoscopic camera and view it on a vertically split screen. The VR glasses use two lenses to correct the light angle that comes out of the smart phone screen which is placed inside the VR glasses as shown in Figure 10. a. This gives the operator a 3-D view.

The smart mobile phone runs a software which traces the user head movement based on the angles' data gathered from the built-in IMU. This data is then transmitted to the UGV. The same software is also responsible for receiving and displaying video feed from the stereoscopic camera. The two main parts of this software are discussed in the following sub-sections.

7.1 Head-Movement Tracking & Transmitting Thread

The software which runs on the smart phone runs a head-movement tracking thread which captures the Yaw, Pitch and Roll rotation angles of the smart phone using the data gathered from the Accelerometer and Geomagnetic Field Sensor of the smart phone's IMU. The data of the three angles is then transmitted to the UGV's #1 Pi board which in turn sends it to #2 Pi board. We adhere to the following definitions of the three rotation angles (we refer the reader to Figure 10.b for the orientation of the axes):

(i) Yaw (degrees of rotation about the z-axis): "This is the angle between the device's current compass direction and magnetic north. If the top edge of the device faces magnetic north, the azimuth is 0 degrees; if the top edge faces south, the azimuth is 180 degrees. Similarly, if the top edge faces east, the azimuth is 90 degrees and if the top edge faces west, the azimuth is 270 degrees".

(ii) Pitch (degrees of rotation about the x-axis): "This is the angle between a plane parallel to the device's screen and a plane parallel to the ground. If you hold the device parallel to the ground with the bottom edge closest to you and tilt the top edge of the device toward the ground, the pitch angle becomes positive. Tilting in the opposite direction and moving the top edge of the device away from the ground causes the pitch angle to become negative. The range of values is -180 degrees to 180 degrees".

(iii) Roll (degrees of rotation about the y-axis): "This is the angle between a plane perpendicular to the device's screen and a plane perpendicular to the ground. If you hold the device parallel to the ground with the bottom edge closest to you and tilt the left edge of the device toward the ground, the roll angle becomes positive. Tilting in the opposite direction and moving the right edge of the device toward the ground causes the roll angle to become negative. The range of values is -90 degrees to 90 degrees".

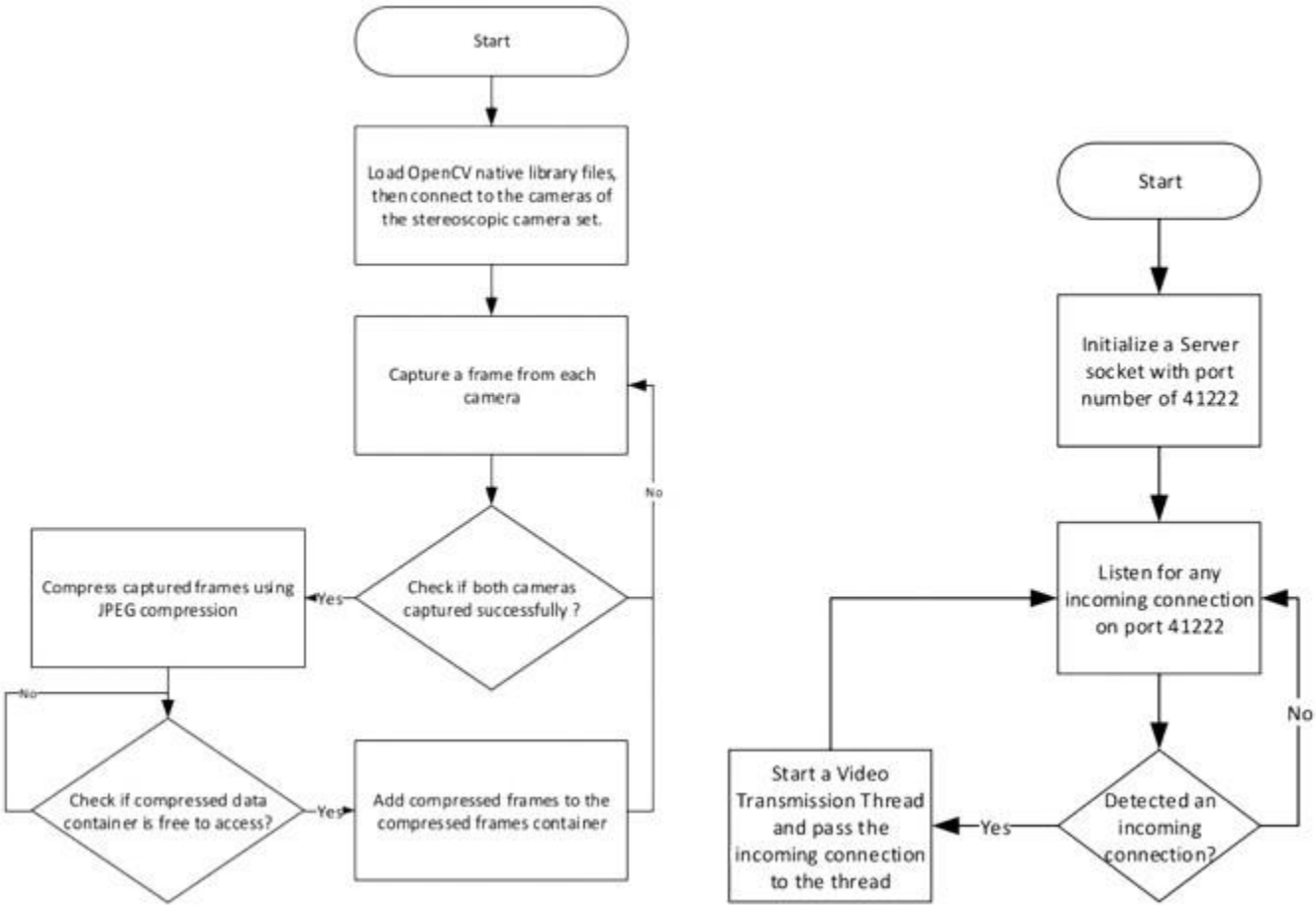
The android development environment has a native hardware support library for the IMU, which contains a number of classes: Sensor, Sensor Event, Sensor Event Listener and Sensor Manager. The thread is based on these classes:

(i) Sensor: this class is used to register the sensors in the device. It should be noted that not all android devices contain the same sensors and features.

(ii) Sensor Event: this class represents a Sensor event and holds information such as the sensor's type, accuracy and of course the sensor's data.

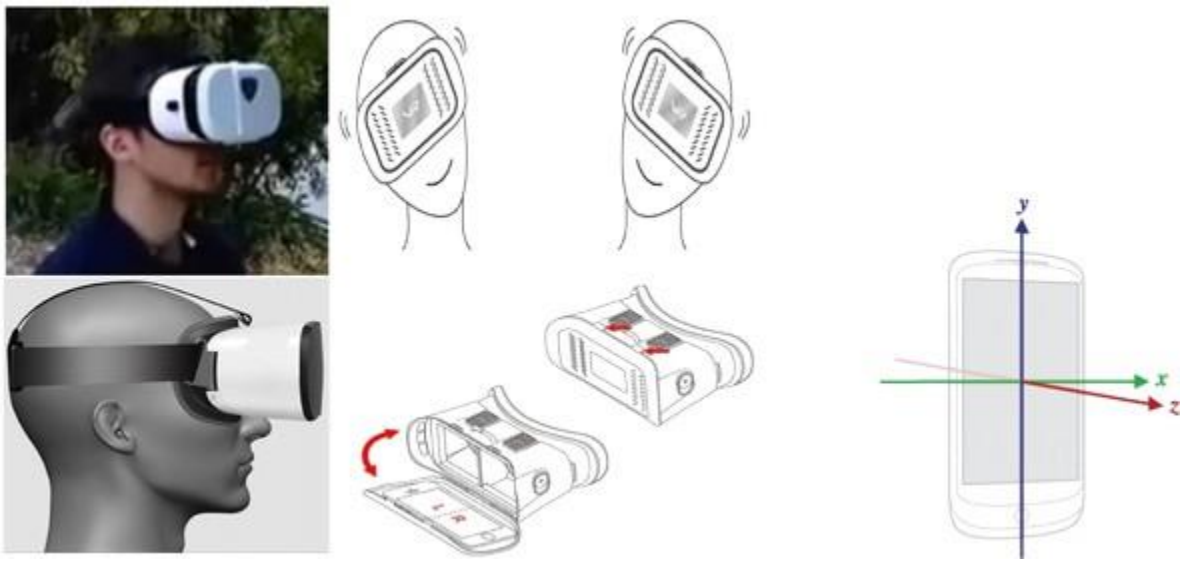
(iii) Sensor Event Listener: this class is used for receiving notifications from the Sensor Manager when there is new sensor data.

(iv) Sensor Manager: this class allows access to the device's sensors. Initializing an instance of this class is done by calling Context.getSystemService() with the argument SENSOR_SERVICE.



(a) Capturing thread (b) Client's handler thread

Figure 9. Flowcharts of threads.



(a) VR head mounted display (b) Axes orientation of the smart phone

Figure 10. VR Headset.

The flowchart of the thread is depicted in Figure 11. When the thread starts, it initializes an instance of type "Sensor Manager" in order to register the sensors which the thread wants to use. In our case, these are the Accelerometer and Geomagnetic Field Sensors. A method (onSensroChanged()) is called each time the sensors' values change to pass the new values to the getRotationMatrix() method, which

in turn computes the inclination matrix and the rotation matrix, transforming a vector from the device coordinate system to the world's coordinate system (direct orthonormal basis). The rotation matrix is then passed to another method (getOrientation()) to compute and return the three rotation angles (Yaw,Pitch and Roll angles) according to the mathematical formulae described in [13]. Once the angles are obtained, they are parsed into three integers which are then sent to the UGV #2 Pi board via #1 Pi board to rotate the 3-axis rotation platform accordingly.

7.2 Headset Video Receiver Thread

Another thread was developed to handle the video feed from the stereoscopic camera. Initially, the software asks the operator to enter the internet protocol (IP) addresses of the Pi boards (see Figure 12.a), subsequently control is moved to the video feed receiver thread to create a socket and establish a connection with the #1 Pi board. Afterwards, the headset starts receiving video feed and transmitting the head movement data. Figure 12.b illustrates the flowchart of the thread. Once the operator taps the connect button on the smart phone, the application starts a new activity that projects the received video feed in a vertically split screen, where each portion of the screen shows a different camera feed in full screen mode, then two threads are started; a thread that receives the data from the UGV and projects it on the screen, and another thread (discussed in the previous sub-section) which tracks the head movement and transmits it to the UGV.

The video feed receiver thread creates a client socket and establishes a connection to the UGV (#1 Pi) using TCP protocol. The thread then starts reading the transmitted frames from the socket's input stream: a frame of the right camera, then a second frame of the left camera, and so forth. The frames are then decompressed and displayed on the smart phone screen using Android ImageView class.

8. OPERATOR'S STATION

The operator's station consists of one processing unit and a laptop computer (Lenovo Y700, Core i7 6700HQ CPU, 16GB RAM, 1TB HDD). This unit is responsible for two operations: First, reading control data from the gamepad – which is attached to the laptop – and transmitting it to #2 Pi via #1 Pi. Second, receiving video feed from the Pi camera and performing the required image processing for: object detection using the SIFT algorithm [14] and 3-D model construction using RealityCapture software.

8.1 SIFT Algorithm

The SIFT algorithm extracts distinctive key points from images that can be used to match images of the same objects, but with different viewpoints. The algorithm is invariant to scaling and rotation and very robust for matching over affine changes, change in 3D viewpoint, noise and illumination. The extracted key points can be used for object recognition, in fact, it has been shown [14] that the algorithm is capable of robustly identifying objects among clutter and occlusion in near real-time. We give a brief overview of the SIFT algorithm steps below.

(i) Scale-space Extrema Detection

A corner may not be a corner if the image is scaled. To illustrate; in the image in Diagram – 1, a curve in the left image within a small window is linear when it is zoomed in the same window. It is therefore, obvious that we cannot use the same window to detect key points with different scale. In order to detect larger corners, larger windows are needed. SIFT algorithm uses scale-space filtering for this purpose. The scale space $L(x,y,\sigma)$ is defined as:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y),$$

where, $I(x,y)$ is the input image, the "*" is the convolution operator and the $G(x,y,\sigma)$ is the Gaussian kernel function, defined as:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma} e^{-\left(x^2+y^2\right)/2\sigma^2}$$

In principle, σ acts as a scaling parameter. For example, in Diagram 1 , a Gaussian kernel with low σ gives high value for small corner, while Gaussian kernel with high σ gives high values for larger

corner. So, local maxima across the scale and space can be found, which produces a list of (x,y,σ) values that act as a good candidates for key point at (x,y) at σ scale.

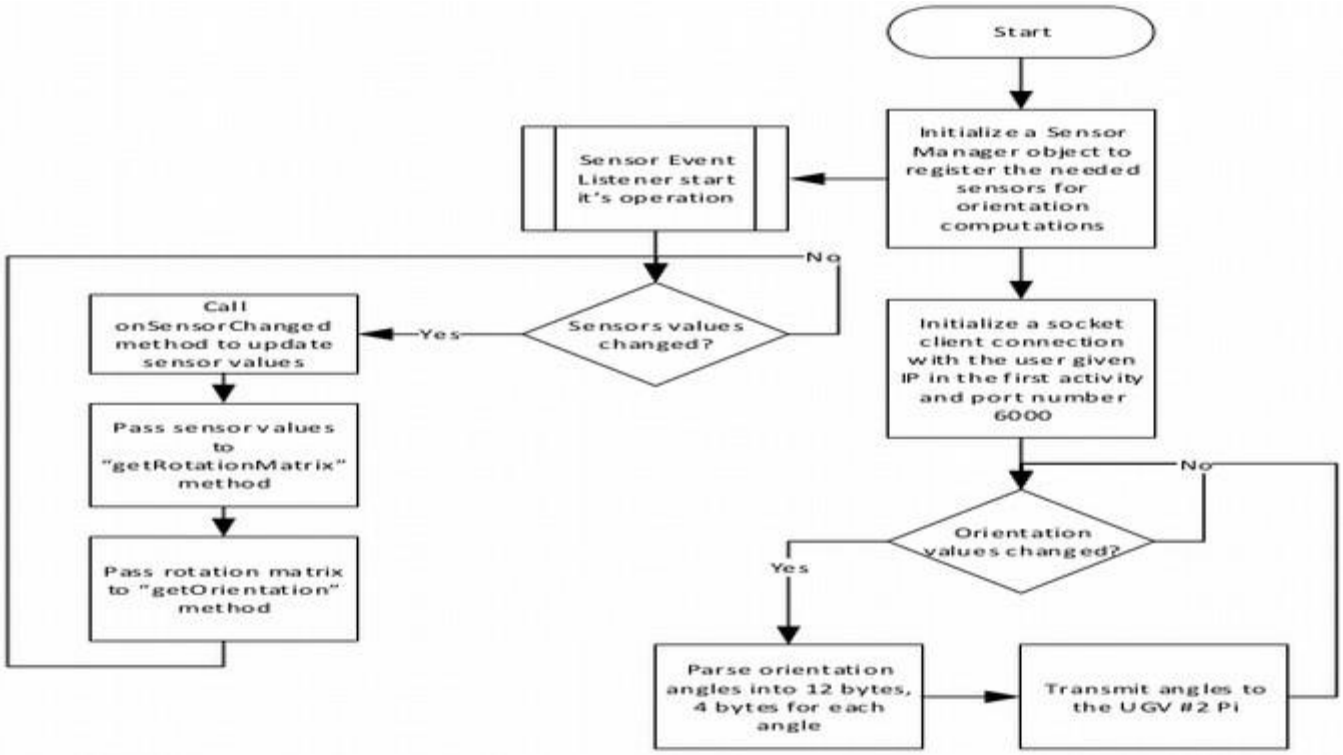


Figure 11. Flowchart of the head-tracking thread.

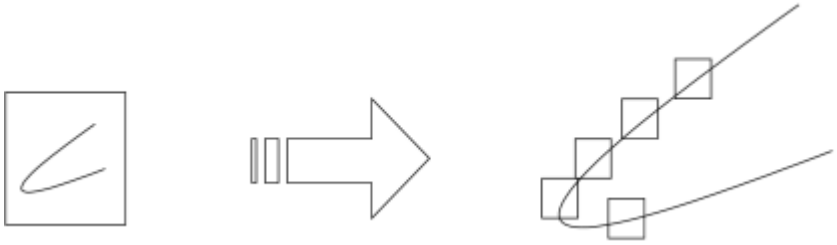


Diagram 1. Basic idea of scaling.

To improve the performance, SIFT algorithm uses Difference of Gaussians (DoG), illustrated mathematically by the following equation:

$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y)$$
$$D(x,y,\sigma) = L(x,y,k\sigma) - L(x,y,\sigma)$$

DoG is obtained as the difference of Gaussian blurring of an image with two different σ; that is σ and $k\sigma$. This process is done for different octaves of the image in Gaussian Pyramid, as shown in Diagram 2.

Once this DoG is found, images are searched for local extrema over scale and space. For example, one pixel in an image is compared with its 8 neighbours as well as 9 pixels in the next scale and 9 pixels in the previous scales – see Diagram 2. If it is a local extrema, it is a potential key point. In fact, this means that key point is best represented in that scale.

(ii) Key point Localization

The potential key points found in the first step are refined for higher accuracy. The algorithm uses Taylor series expansion of the scale space to get more accurate location of extrema. In addition to that, if the intensity at this extrema is less than a predefined threshold (0.03), it is rejected. It is worth noting that DoG has higher response for edges; so, edges also need to be removed. This is achieved by the

algorithm using a 2x2 Hessian matrix to compute the principal curvature. For edges, one eigen value is larger than the other. If the ratio of one eigen value to the other is greater than a threshold (10), the key point of the edge is discarded.

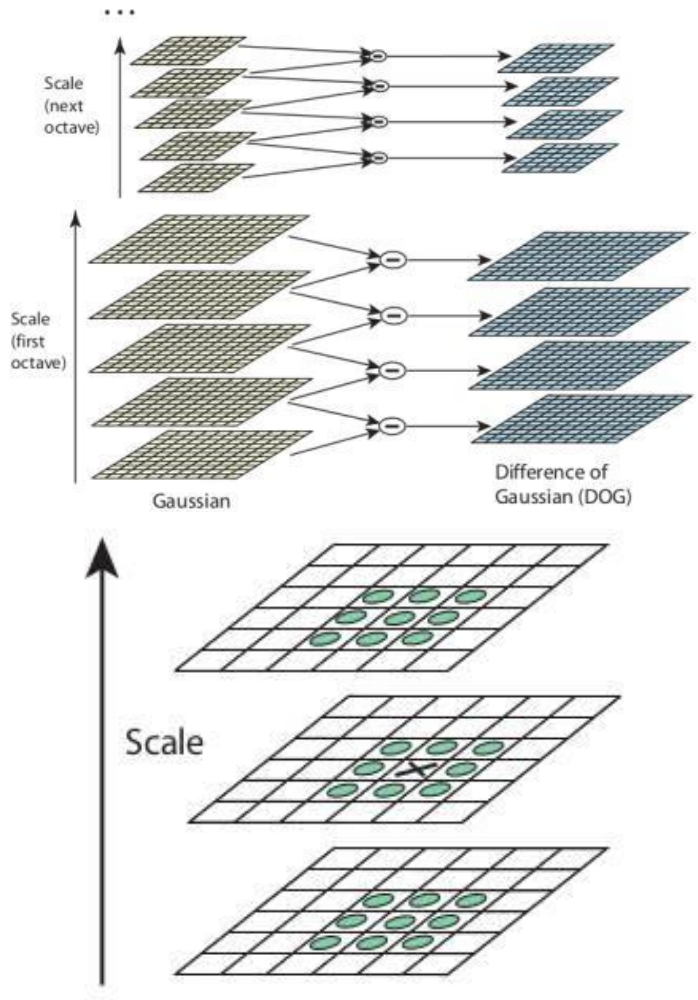


Diagram 2. Difference of Gaussians between images (Adopted from [14]).

(iii) Orientation Assignment

The orientation of each key point is then assigned to make the algorithm invariant to image rotation. Depending on the scale, a neighbourhood is taken around the key point; the gradient is then calculated in the vicinity of the key point and orientation histogram with 36 bins the width of each is 10 degrees (i.e., covering 360 degrees) is established. This is weighted by gradient magnitude and Gaussian-weighted circular window with σ equal to 1.5 times the scale of the key point. The orientation is determined using the angles that belong to the bins with peaks more than 80% of the maximum peak.

(iv) Key Point Descriptor

A key point descriptor for each key point is created by first taking 16x16 neighbourhood around the key point which is further divided into 16 sub-blocks of 4x4 size. Then, for each sub-block, 8-bin orientation histogram is created (i.e., total of 128 bin values). Finally, this is represented as a vector to form the key point descriptor.

(v) Key point Matching

Matching between key points is achieved by means of identifying their nearest neighbours. Due to noise, it is possible to have cases where the second closest-match is very near to the first. In that case, ratio of closest-distance to second-closest distance is taken and if it is greater than 0.8, they are

rejected. This eliminates approximately 90% of false matches while discarding only 5% of the correct matches [14].

8.2 The Gamepad

The gamepad used in our work is "Logitech F310". It is attached to the laptop and is used to control the 3-axis rotation platform in the manual mode. The gamepad consists of three analog joysticks; two of them move in two axes (see Figure 13.a) and the third one moves in one axis (Figure 13.b). One of the two-axes analog joysticks is used to drive the UGV and the other one with the one-axis analog joystick is used to move the 3-axis rotation platform in the manual mode.

A software was developed (see Figure 14) using Java programming language and utilizing the "Jinput" library to communicate with the gamepad using serial communication. Also, the software uses Java stream sockets to communicate with the UGV #2 Pi board using a dedicated TCP connection. Initially, the software searches for all input devices which are attached to the laptop. When the gamepad is recognized, it initializes a connection to the gamepad and starts receiving data from it through the USB port; at this stage, it also establishes a TCP connection to #2 Pi board. The gamepad sends data whenever the operator does an action on any of the buttons or joysticks of the gamepad. The data which is read from the gamepad contains information about the ID of the part that was moved and the movement (both magnitude and direction). The ID is checked by the software to determine which part was moved, e.g. if the ID matches the ID of the left joystick, this means that the operator wants to move the UGV; the software then determines the direction which the joystick was moved in (see Table 3) and maps the value of the analog joystick to a number in the range of 0-9; where number 0 means slowest speed (rover is stopped) and number 9 means the highest speed. Subsequently, the software sends the information about the magnitude and direction to the UGV #2 Pi board.

The software is also responsible for moving the 3-axis rotating platform in the manual mode. The toggling between manual mode and VR-headset mode is done by pressing the "A" button on the gamepad (see Figure 13.a). When the operator presses the button, the software sends a toggling command to the UGV #2 Pi board; subsequently, the software detects any movements of the relevant joysticks (mentioned in the first paragraph of this sub-section) and transmits the information to UGV #2 Pi board using the same TCP connection.

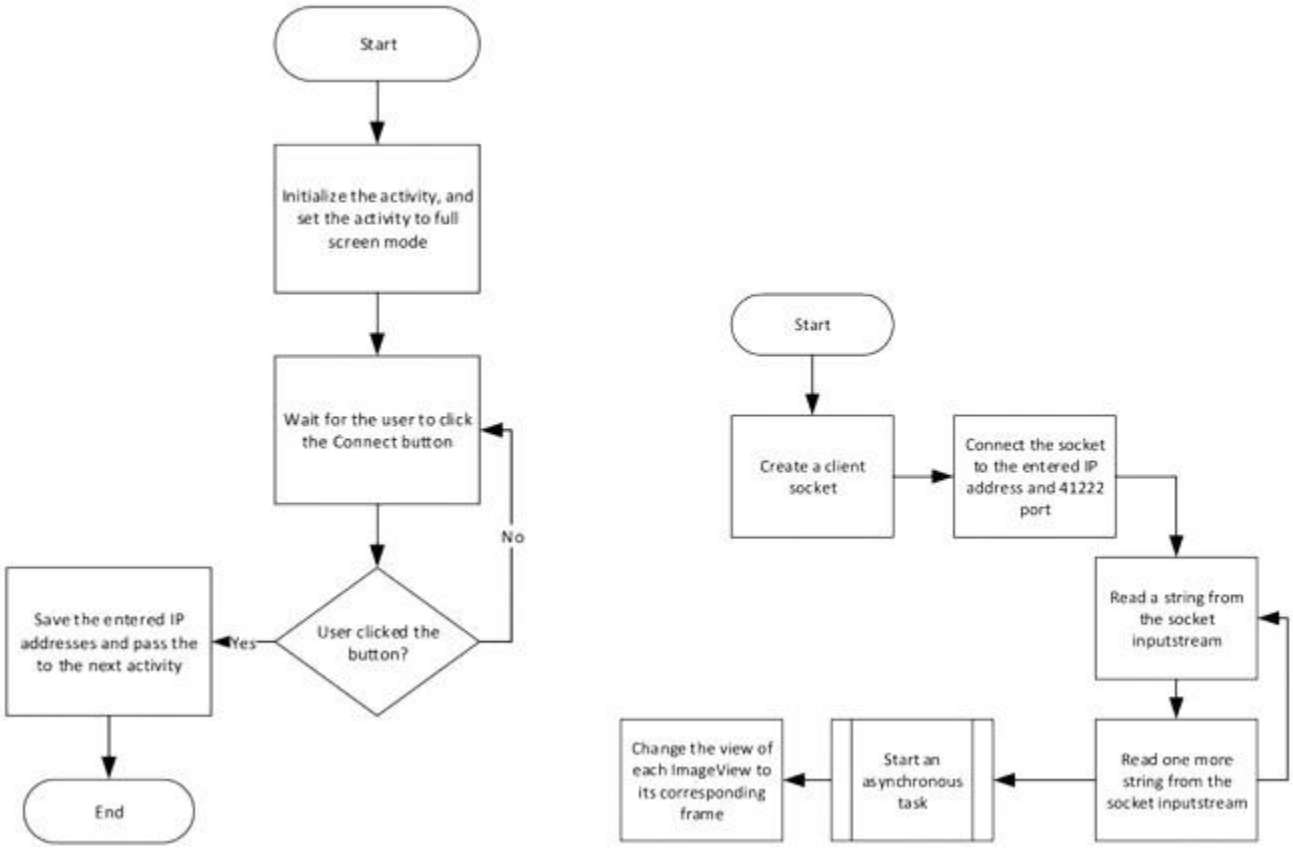
8.3 Object Detection & 3-D Model Creation

The operator-station laptop receives a video feed from both the stereoscopic camera and the Pi camera and applies image processing as follows:

1. The SIFT algorithm is used to compare the features extracted from right and left pictures received from the stereoscopic camera with the features of a reference object which the operator want to detect. To illustrate, for any object in a picture, interesting points on the object can be extracted to provide a "feature description" of the object. This description, extracted from a training reference image, can then be used to identify the object when attempting to locate the object in the pictures captured by the camera which contain many other objects. Technically speaking, this is implemented in two threads (flowcharts of which are shown in Figure 15.a and Figure 15.b). The receiving thread saves the pictures in a "List of Mat". In OpenCV library, the class "Mat" represents an n-dimensional array, which is used to store gray-scale or color images. Then, the SIFT algorithm computes key points and descriptors for both the reference image and the captured image; the distance is computed and if it is less than or equal to 25 – found experimentally to be a good match – the object is considered detected.
2. Another software (RealityCapture) is used to build a 3-D model of the UGV's surrounding environment from multiple of 2-D pictures taken by the operator using the Pi camera. The 2-D pictures are stored on an on-board SD card on #1 Pi and retrieved by the operator using a secure shell (SSH) connection. The software then automatically aligns images and detects matching keypoints of the images to create a 3-D mesh map. Finally, the operator can either choose to obtain a coloured or gray-scale high-quality texture of the object. Figure 16.a shows multiple 2-D images taken from the Pi camera in one of the rover trips, and Figure 16.b shows the 3-D model that was created from these images.

9. RESULTS & DISCUSSION

The diversity in UGV designs and specifications (e.g. weight, payload, speed, size, ...) renders a comprehensive performance evaluation of existing UGVs a none-trivial task. We restricted our comparison to the ADORA [9] and RAPOSA [11] UGVs, since both have similar design objectives as our UGV, specifically: mobility, victim identification, mapping (i.e., ability to generate 3D maps, of the surrounding areas and present them in an intuitive way), and HRI (i.e., ability to provide the operator with visual information about the robots' immediate surroundings). In addition to that, since delay is an important factor in teleoperated UGVs [23, 24] and because our UGV relies on TCP for transmitting video over WiFi, we studied the effect of delay on received video over a range of distances.



(a) Initial activity (b) Video feed receiver

Figure 12. Flowcharts showing initial activity and video feed receiver.

(a) Gamepad buttons assignment (top view) (b) Gamepad buttons assignment (front view)

Figure 13. Gamepad top and front views.

We have found that the developed UGV outperforms ADORA UGV in terms of speed, which is depicted in Figure 17.b. The developed UGV has a speed of 3.3 m/s, while ADORA UGV has a speed of 0.61 m/s. However, ADORA UGV can handle around 1kg extra payload compared to the developed UGV, as illustrated in Figure 17.c. We look at this as a trade-off between speed and payload weight. Figure 17.a shows the weight of each UGV; a large difference in weight can be seen between the developed UGV and the other two UGVs. Subject to transmission delay constraints, we believe that the relatively light weight and high speed of the developed UGV give it high manoeuvrability.

We considered the effect of transmission delay on the quality of received video. Particularly, we studied TCP performance over a range of distances (5m – 70m) with increments of 5m. This range is the typical operational range of the developed UGV. In each experiment, we transferred a total of 5 MB from the UGV to the laptop at the operator's station and recorded the round trip time (RTT), average packet size and TCP throughput. The results are shown in Figure 17(d-f). The RTT was stable in the operation range with an average value around 50µs. The perceived quality of video over this range at this value was acceptable. The throughput and average packet size are depicted in Figure 17.e and Figure 17.f, respectively. The average packet size is around 1500 byte and the throughput is approximately 2Mbps. No packet loss or packet time-outs were reported. However, it is worth to mention that we invoked TCP-BBR [21] congestion control algorithm to manage the congestion window of TCP. TCP-BBR is optimized to achieve significant bandwidth improvements and lower latency.

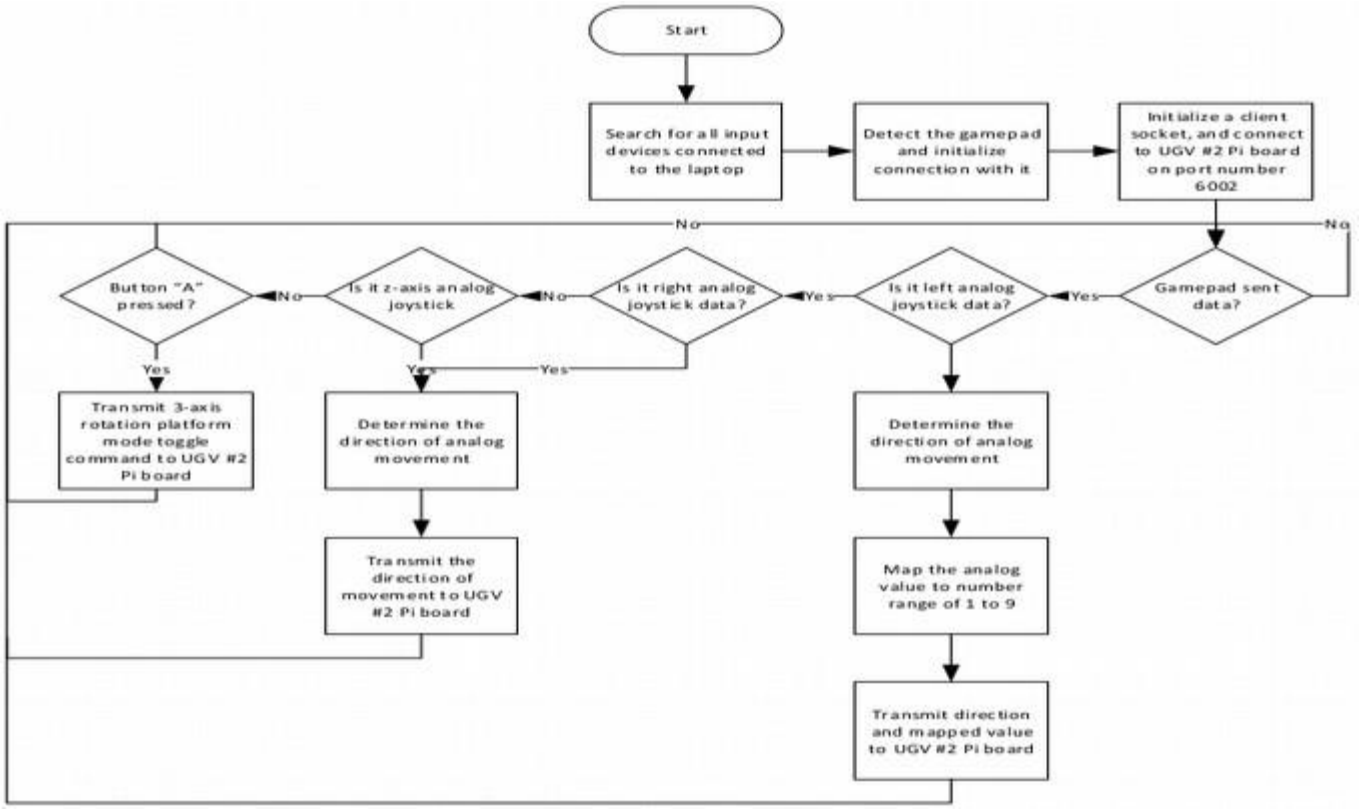
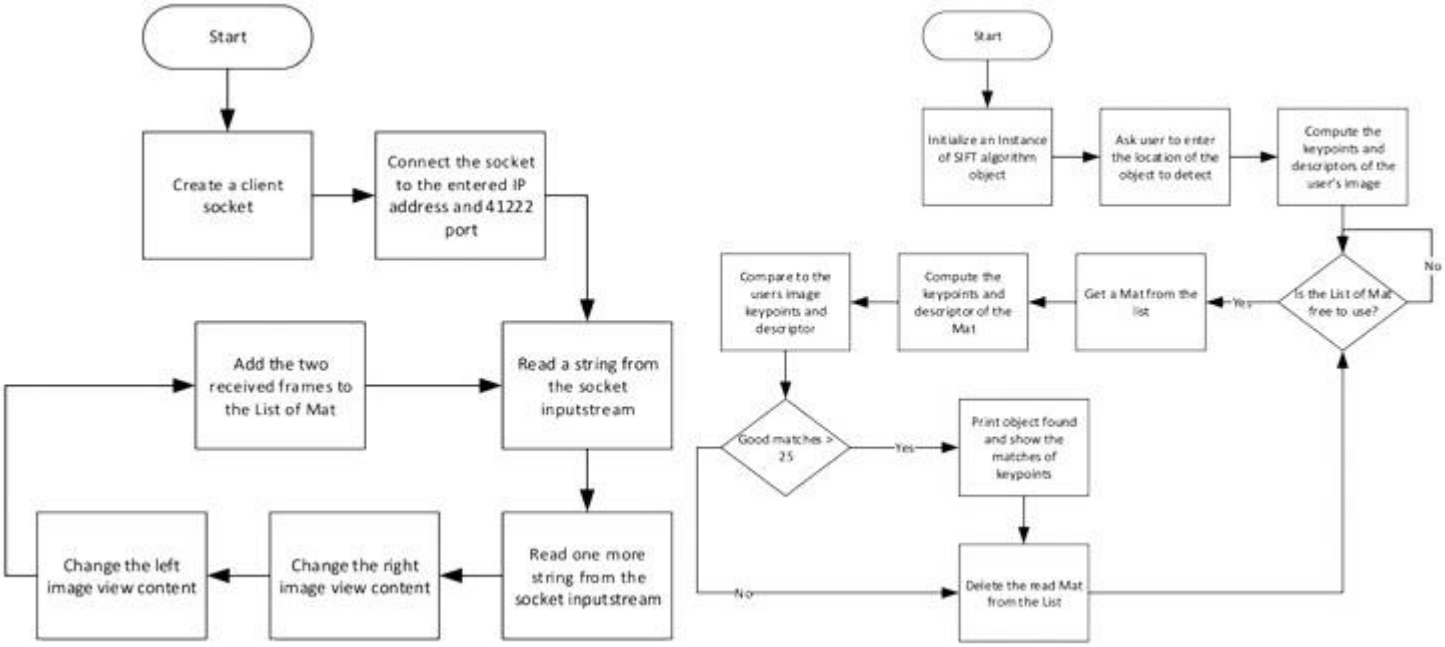


Figure 14. Flowchart of the gamepad software.

10. CONCLUSION & FUTURE WORK

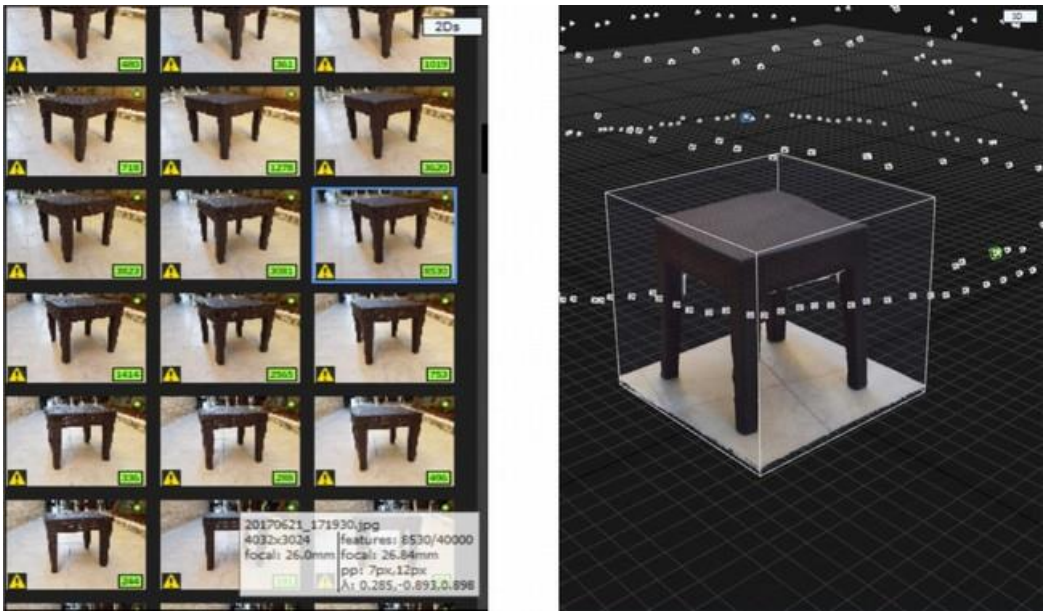
A SAR UGV with high visual inspection capabilities and relatively high speed was designed and implemented. The main contribution of the work dwells in the design of a separate user-interface unit constituting of a 3-axis rotating platform with three cameras: a stereoscopic camera and a Pi camera. The unit has two control modes, one is automatic through a smart phone-based VR-headset and the other is manual through a gamepad attached to the operator-station laptop. Video feeds from the unit are transmitted over a wireless link to both the VR-headset to provide visual feedback and to the operator-station laptop for: object detection – in the images captured by the stereoscopic camera – using the SIFT algorithm and 3-D model creation from 2-D images captured by the Pi camera.

Multiple computers can connect simultaneously to the unit to receive video feeds and since the wireless connection is realized by means of managed Wi-Fi, a team of operators can cooperate using a shared video conferencing session while viewing the video feed from the unit on their computers.



(a) Operator-station laptop video feed receiving and viewing, b) SIFT workflow thread

Figure 15. Flowcharts of Threads.

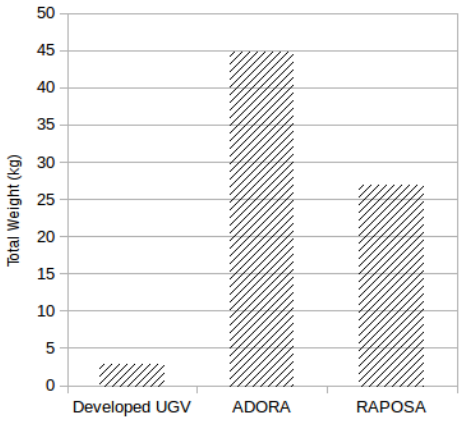


(a) 2-D pictures taken by the Pi camera (b) 3-D model created by the software

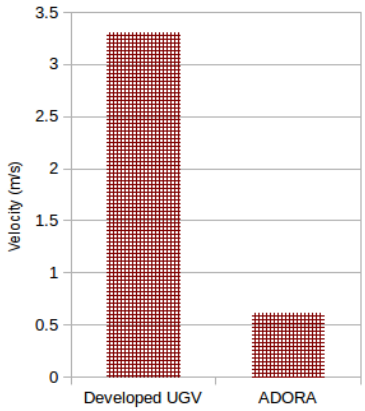
Figure 16. 3-D model creation.

A wheel-type rover was designed to hold the separate unit; the relatively high-speed of the rover (3.3 m/s) six-wheel differential-drive chassis and spiked air-filled rubber tires gave the rover high manoeuvrability in open rough terrain compared to other SAR UGVs found in literature.

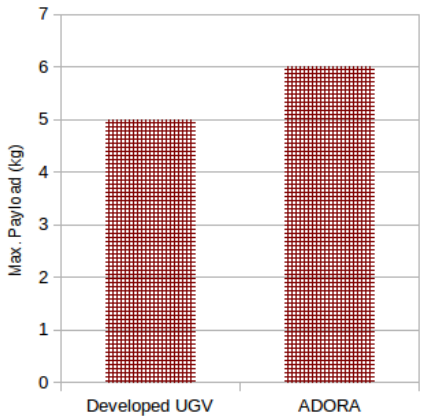
The UGV was designed and implemented in a laboratory environment. It was tested over soft and rough terrain and results showed its suitability for civil defence search and rescue applications, as well as for hazard areas (e.g. military) reconnaissance applications. Future work of this project goes in two directions: First, enhancing the night-vision capabilities of the UGV which can be done by connecting



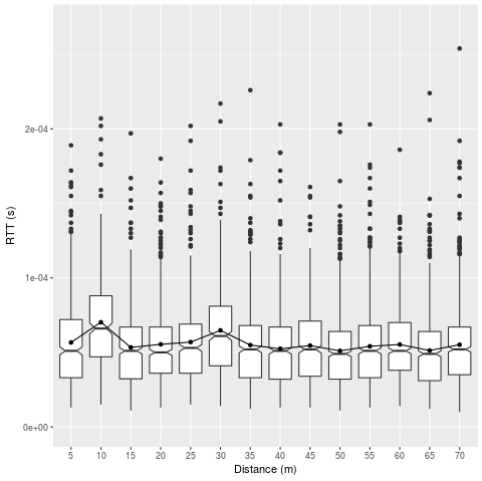
(a) Total Weight (kg)



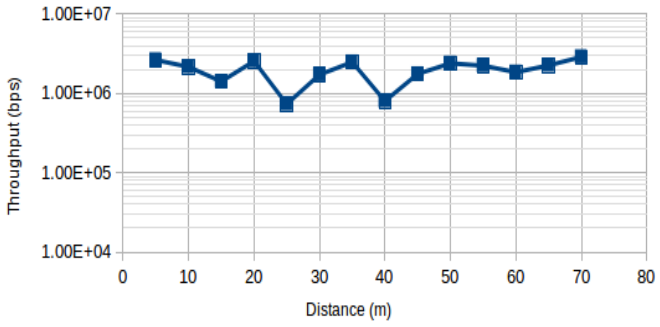
(b) Velocity (m/s)



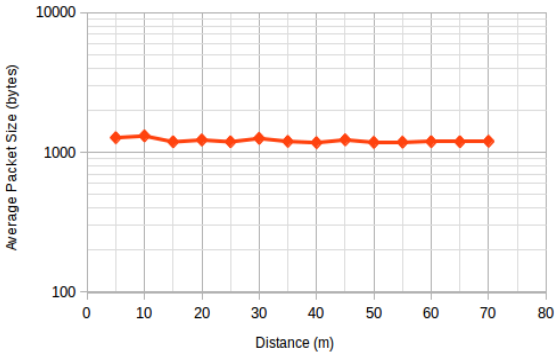
(c) Maximum Payload (kg)



(d) Round Trip Time (seconds)



(e) Throughput (bps)



(f) Average Packet Size (bytes)

Figure 17. Comparison of relevant UGV (a-c) and Performance measures of communication protocol (d-f).

an infra-red camera (Raspberry Pi Infra-red Night Vision Camera Module V2.1 IMX219 8-megapixel sensor) to #2 Pi using the on-board 15-pin CSI-2. Also, a LED torch can be mounted on top of the 3-axis platform which can be controlled by the operator through the gamepad in critical situations (e.g. in a dark pipe). Second, enhancing the exploratory capabilities of the UGV by implementing a general-purpose robotic arm. For instance, the arm can hold a special bucket to collect samples of soil, liquid, etc. Also, a separate sensor box can be added on the rover to sense temperature, humidity and gas. The data of the sensor box can be sent to a dedicated computer – other than the one used for displaying the video feed – for further analysis of the UGV's surrounding environment. Finally,

enhancing the processing power to make use of advanced software; for instance, using a dedicated graphical processing unit (GPU) such as for example NVidia Jetson development kit, which can increase the retrieved frame rate of camera, and allow to run software for real-time face recognition and autonomous driving.

REFERENCES

[1] K. Cavallin and P. Svensson, "Semi-autonomous, Teleoperated Search and Rescue Robot," 2009.

[2] J. L. Drury, J. Scholtz and H. A. Yanco, "Awareness in Human-robot Interactions," IEEE International Conference on Systems, Man and Cybernetics, vol. 1, pp. 912-918, Oct. 2003.

[3] N. Michael, S. Shen, K. Mohta, Y. Mulgaonkar, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, Yoshida et al., "Collaborative Mapping of an Earthquake-damaged Building via Ground and Aerial Robots," Journal of Field Robotics, vol. 29, no. 5, pp. 832-841, 2012.

[4] R. R. Murphy, J. Kravitz, S. L. Stover and R. Shoureshi, "Mobile Robots in Mine Rescue and Recovery," IEEE Robotics & Automation Magazine, vol. 16, no. 2, 2009.

[5] Y. Baudoin, D. Doroftei, G. De Cubber, S. A. Berrabah, C. Pinzon, F. Warlet, J. Gancet, E. Motard, M. Ilzkovitz, Nalpantidis et al., "View- finder: Robotics Assistance to Re-fighting Services and Crisis Management," IEEE International Workshop on Safety, Security & Rescue Robotics (SSRR), pp. 1-6, 2009.

[6] D. Doroftei, G. De Cubber, E. Colon and Y. Baudoin, "Behavior Based Control for an Outdoor Crisis Man-agement Robot," Proceedings of the IARP International Workshop on robotics for Risky Interventions and Environmental Surveillance, pp. 12-14, 2009.

[7] G. De Cubber, D. Doroftei, H. Sahli and Y. Baudoin, "Outdoor Terrain Traversability Analysis for Robot Navigation Using a Time-of- light Camera," Proc. RGB-D Workshop on 3D Perception in robotics, Vasteras, Sweden, 2011.

[8] C. Marques, J. Cristvo, P. Alvito, P. Lima, J. Frazo, I. Ribeiro and R. Ventura, "A Search and Rescue Robot with Teleoperated Tether Docking System," Industrial Robot: An International Journal, vol. 34, no. 4, pp. 332-338, 2007, [Online], Available: https://doi.org/10.1108/01439910710749663.

[9] N. Enayati and F. Najafi, "Design and Manufacturing of a Teleoperative Rescue UGV with a Novel Track Arrangement," Industrial Robot: An International Journal, vol. 38, no. 5, pp. 476-485, 2011, [Online], Available: https://doi.org/10.1108/01439911111154045.

[10] K. Berns, A. Nezhadfard, M. Tosa, H. Balta and G. De Cubber, "Unmanned Ground Robots for Rescue Tasks," Search and Rescue Robotics-From Theory to Practice, InTech, 2017.

[11] H. Martins and R. Ventura, "Immersive 3-D Teleoperation of a Search and Rescue Robot Using a Head-mounted Display," IEEE Conference on Emerging Technologies Factory Automation, pp. 1-8, Sept. 2009.

[12] "Pololu: Robotics & Electronics," [Online], Available: http://www.pololu.com/, 2017, accessed: 3-October-2017.

[13] M. Pedley, "Tilt Sensing Using a Three-axis Accelerometer," Freescale Semiconductor Application Note, vol. 1, pp. 2012-2013, 2013.

[14] D. G. Lowe, "Distinctive Image Features from Scale-invariant Key Points," International Journal of Computer Vision, vol. 60, no. 2, pp. 91-110, 2004.

[15] C. M. Johnson and S.-J. Hsieh, "Board# 143: Maker: Urban Search and Rescue UGV: Visual localization and Navigation," ASEE Annual Conference & Exposition, 2017.

[16] Z. Wang and H. Gu, "A Review of Locomotion Mechanisms of Urban Search and Rescue Robot," Industrial Robot: An International Journal, vol. 34, no. 5, pp. 400-411, 2007, [Online], Available: https://doi.org/10.1108/01439910710774403.

[17] M. Neumann, T. Predki, L. Heckes and P. Labenda, "Snakelike, Tracked, Mobile Robot with Active Flippers for Urban Search and Rescue Tasks," Industrial Robot: An International Journal, vol. 40, no. 3, pp. 246-250, 2013, [Online], Available: https://doi.org/10.1108/01439911311309942.

[18] P. U. Lima, "Search and Rescue UGVs: The Civil Protection Teams of the Future," 3rd International Conference on Emerging Security Technologies, pp. 12-19, Sept. 2012.

" Unmanned Ground Vehicle with Virtual Reality Vision", M. Al-Khalil, R. Abu-Rhayyem, A. Hammoudeh and T. A. Edwan.

[19] M. W. Kadous, R. K.-M. Sheh and C. Sammut, "Caster: A Robot for Urban Search and Rescue," Proceedings of the 2005 Australasian Conference on Robotics and Automation, pp. 1-10, 2005.

[20] A. Kleiner, Mapping and Exploration for Search and Rescue with Humans and Mobile Robots, University of Freiburg, 2009.

[21] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh and V. Jacobson, "BBR: Congestion-Based Congestion Control," ACM Queue, vol. 14, pp. 20-53, 2016.

[22] Y. Wang, Z. Song, Y. Zhou and C. Zhou, "Design of A Tele-Operated Field Robot," Defense Science Research Conference and Expo (DSR), pp. 1-4, Aug. 2011.

[23] D. Sanders "Comparing Speed to Complete Progressively More Difficult Mobile Robot Paths Between Human Tele-operators and Humans," Assembly Automation, vol. 29, pp. 230-248, 2017.

[24] D. Sanders, "Analysis of the Effects of Time Delays on the Teleoperation of a Mobile Robot in Various Modes of Operation," Industrial Robot: An International Journal, vol. 36, pp. 570-584, 2009.

ملخص البحث:

تصف هذه الورقة البحثية تصميم و تنفيذ مركبة أرضية غير مأهولة ووحدة إظهار هاتف ذكي تركب على الرأس تسمح بادراك الوضع البصري، بحيث يتم إرسال صور الفيديو إلى حاسوب منفصل لكشف الأشياء و إنشاء نموذج ثلاثي الأبعاد لمحيط المركبة من الصور ثنائية الأبعاد للأشياء المحيطة بها.

وقد تم تحقيق التصميم المطلوب باستخدام وحدة إظهار متصلة بهاتف ذكي تلتقط حركات الرأس في الزمن الحقيقي وترسلها من ثم إلى ثلاثة محركات مركبة على عربة لتوفير الحركة حول ثلاثة محاور، بينما يعمل المشغل على التحكم بتلك المحركات إما عبر وحدة إظهار متصلة بهاتف ذكي أو عن طريق لوحة ألعاب. و تعمل ثلاث كاميرات محمولة على متن المركبة على التقاط الصور التي يتم إرسالها إلى وحدة الإظهار المتصلة بالهاتف الذكي و إلى حاسوب المشغل. و تقوم برمجية بتحويل الصور ثنائية الأبعاد الملتقطة لمحيط المركبة إلى نموذج ثلاثي الأبعاد.

لقد تم تنفيذ المركبة في بيئة مخبرية ومن ثم فحصها. وبينت النتائج التي جرى الحصول عليها أن المركبة المقترحة تمتلك خصائص أفضل للتفتيش البصري مقارنة بتصاميم أخرى لمركبات مماثلة. و قد منحتها تلك الخصائص – و أبرزها سرعة قصوى مقدارها 3.3 م/ث، و هيكل دفع تفاضلي بست عجلات، و إطارات مطاطية مملوءة بالهواء – قدرات عالية على المناورة، مما يجعلها خيارا ملائما لاستكشاف الكواكب و الأغراض العسكرية. و يمكن تركيب المحركات و الكاميرات بسهولة على وحدة منفصلة تتضمن هيكلا يستخدم آليات قطر مختلفة لزيادة فعالية عمل المركبة. و يمكن للتصميم المقترح أن يستخدم في بناء الخرائط، وإنشاء النماذج ثلاثية الأبعاد، وكشف الأشياء في الزمن الحقيقي، والتعرف على الوجوه في الزمن الحقيقي، والقيادة الذاتية.