Custom Unmanned Aerial Vehicle for Photography based Terrain Reconstruction

Jernej Kranjec[*]
Supervised by: Borut Žalik[†]

University of Maribor
Faculty of Electrical Engineering and Computer Science
Laboratory for Geometric Modelling and Multimedia Algorithms
Smetanova ulica 17, SI-2000 Maribor / Slovenia

Abstract

Consumer electronics have become considerably powerful in terms of hardware features, which can be expanded beyond their original design with custom software.

This paper serves as a summary of a student's experience of acquiring suitable aerial images for terrain reconstruction. The paper covers the utilization of consumer components like a cell phone with on-board sensors, point-and-shoot cameras and a prefabricated model airplane, combined with easily accessible electronics. This was to create an inexpensive platform for high definition aerial photography, as needed for terrain reconstruction. It describes the challenges of building such a platform and presents an overview of the results.

Keywords: unmanned aerial vehicle, aerial photography, terrain reconstruction

1 Introduction

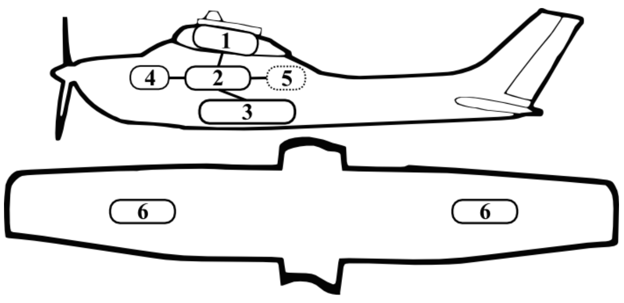
Today consumer devices are more powerful and flexible than ever. By considering the increased popularity of easily accessible hobby-grade remote-operated models, we decided to combine these within a customized unmanned aerial vehicle, adapted for carrying a customized stereo camera rig for aerial photography. Our focus on consumer devices was driven primarily by their prices and availability. By utilizing our skills we turned them into a platform capable of performing features normally found within professional kits.

The current iteration is based on a scaled foam model of a Cessna airplane, which offers a lot of space for necessary modifications and payload. The proposed control unit is split into an Android phone, which is used for sensors, computing and communications, and a control module from custom electronics for controlling the servo-motors used in the model. Two Canon point-and-shoot budget cameras were used for stereo camera mounting. The cameras were chosen due to their support of customized third-party firmware.

This enabled the acquiring of high-resolution aerial images of a desired area with high overlap, which were then used for terrain reconstruction.

2 Model setup

A scaled-down model of a Cessna 182 airplane was chosen made of durable Elapor foam, with a wingspan of 1400mm. This provided a lightweight durable base that could be easily modified, with enough room for additional gear. The model was equipped with a KORA 10-15 brushless electrical engine with a HobbyWing 40A speed regulator driving an APC 11x5.5in propeller, powered by a 2200mAh LiPo battery. The model was controlled by a FrSky 8 channel X8R Remote Control receiver with telemetry feedback, which operates in the 2.4GHz frequency band, allowing over 1km line of sight operational range. 6 standard 9g servo motors were used to operate the control surfaces of the model. A structural diagram of the modified model can be seen in Figure 1.



1. cell phone with panorama attachment
2. control module
3. battery
4. RC-receiver
5. servo-motors
6. point-and-shoot cameras

Figure 1: Aircraft model structural diagram

The model was set up in the following configuration: channel 1 servo operates the vertical stabilizer, channel 2

[*]jernej.kranjec@gmail.com
[†]borut.zalik@um.si

servo operates the horizontal stabilizer, channel 3 controls the engine output, channels 4 and 5 servos operates the ailerons, channels 6 and 7 servos supervise the flaps and channel 8 the spare servo-channel used to signal who is in control.

Modifications performed on the model include mounting the stereo camera rig (see Figure 3) through the fuselage to preserve the model's center of gravity, securing it onto the wings with custom 3D printed holders (see Figure 2) in order to prevent the camera rig from being damaged during takeoffs and landings.

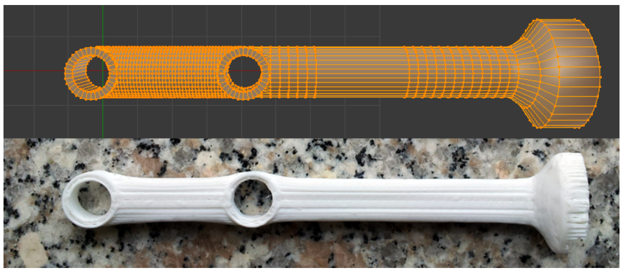


Figure 2: Customized holder, designed in Blender and printed on RepRap 3D printer

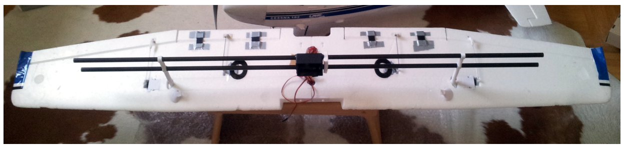


Figure 3: Camera rig mounted onto the wings

The battery bay was extended to accommodate a bigger battery pack and fine tune the model's center of gravity by shifting its position as necessary. Stock landing gear was replaced with one made out of carbon fiber mounted on flexible FR-4 fiber glass strip (see Figure 4), in order to better absorb and withstand the increased weight of the model.

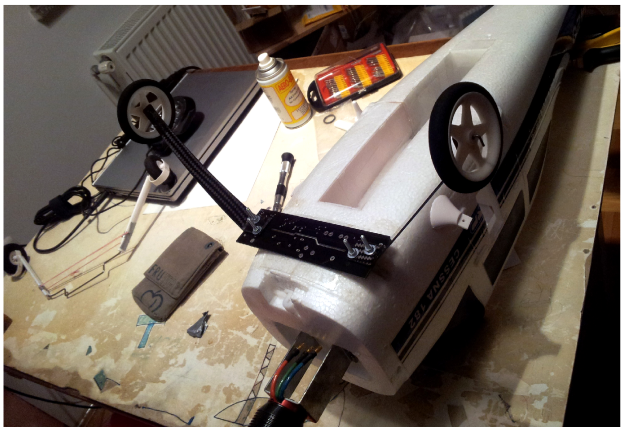


Figure 4: Carbon fiber landing gear with extended battery bay

Lastly, attaching a customized cell phone case and holder carved into the wing on the top of the fuselage with a modified Kogeto DOT 360° panoramic lens attachment, for capturing video of the model in flight for later review and visualization.

The complete setup (see Figure 5), including the cameras weights 1.9kg, provides 5 minutes of flying time, out of which 2-3 minutes are at the minimum desired altitude difference of 200m for taking aerial photographs. The total traveled distance of the model using that configuration is about 4.5km at an average ground speed of around 50km/h.

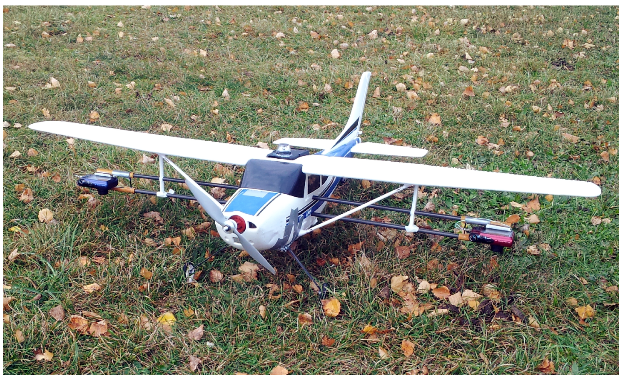


Figure 5: Photo of the model ready for take-off

3 Electronics

First part of the control unit was based around the Samsung Galaxy S2 cell phone. The device was chosen because of the available on-board sensors containing a triple-axis accelerometer, triple-axis gyroscope and a triple-axis compass, which were used as an Inertial Measurement Unit in the autopilot implementation in order to determine the relative orientation of the model. A GPS receiver for locational logging and navigation and a GPRS modem for communication with the ground computer and real-time visualization of the flight. It also contained an 8MP camera with video capabilities used for recording in-flight video of the model, as well as providing a 1GHz dual core computer with 1GB of RAM within a programmable-friendly environment.

For the second part a control module was constructed from custom electronics, which took the control inputs from either the phone or the RC-receiver and performed the actual control of the servos on the model. This allowed for controlling of the servos from the phone as well as the remote control.

The customized electronics consisted of an Atmega128 microcontroller from Atmel, an FT230x USB-serial bridge from FTDI, and some passive components. The Atmega128 was chosen because of its hardware support for driving a large number of servo-motors using its timer modules. It also provided pin change interrupts, which were used to read the outputs from the RC-receiver. The signals generated and read by the microcontroller are standard servo control signals with a 50Hz period and a duty cycle of 1-2ms, where 1ms represents -45 servo rotation from center and 2ms corresponds to 45 rotation. The On-The-Go USB capabilities of the phone were used to connect the control module to the phone, which allows connection of the device to the phone. We utilized the serial interface of the microcontroller to connect the phone to the control module using a USB-serial bridge. We used the serial connection simplified programming of the module as it negated the need for any high-level abstraction requested by the USB protocol. The electronics block diagram can be seen in Figure 6.

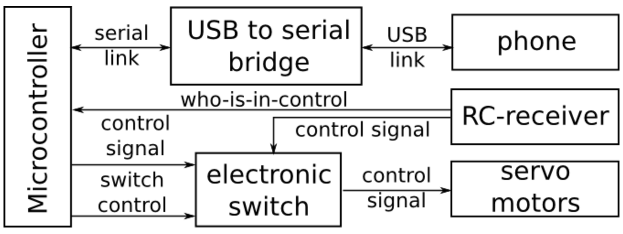


Figure 6: Control module block diagram

4 Camera setup

Two Canon A2200 cameras were chosen for this setup as they were inexpensive, light, and allowed us to extend their functionalities using third party firmware. Our first step was porting the open-source firmware CHDK on to the camera. This allowed the usage of the camera's parameters and settings, which are usually hidden within the consumer firmware, like manual focus and exposure control. We also acquired new features, among others the ability to save raw 12bit images, run custom scripts, and synchronize the shutter release across multiple cameras.

The following method is employed to test the camera's ability to capture synchronized stereo images. A rotating platform was used, constructed of a small electric motor and a CD-ROM holder, which held a CD-ROM, onto which an LED with a battery was glued and balanced. We took the first picture with a known exposure time of 20ms. Using the light trail left on the image, it was possible to calculate the rotational speed of the setup, which was 10.36Hz. Knowing this, pictures of the rotating platform with both cameras shutters synchronized were taken and the light trails on both pictures took into account the start and end-points, and its length was then compared. Using multiple measurements the deduction was made that the synchronization between cameras were less than 0.1ms apart. Assuming a speed of 50km/h for the model, the cameras would trigger in less than 1.4mm of travel between each other, making the perspective distortion minimal. Testing the cameras can be seen in Figure 7.

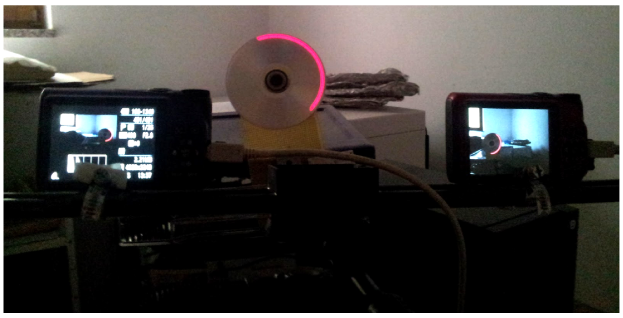


Figure 7: Camera synchronization testing rig

In order to construct the stereo camera rig, 1m long prefabricated 1cm diameter hollow carbon fiber tubes were used. The carbon fiber tubes were perfectly aligned parallel to each other, placed 40mm apart and glued onto a supporting plastic platform, which operated as a camera stand mount. In order to attach and align the cameras, a cradle (see Figure 8) was made out of larger tubes, which held a standard 3/4in camera mount screw.

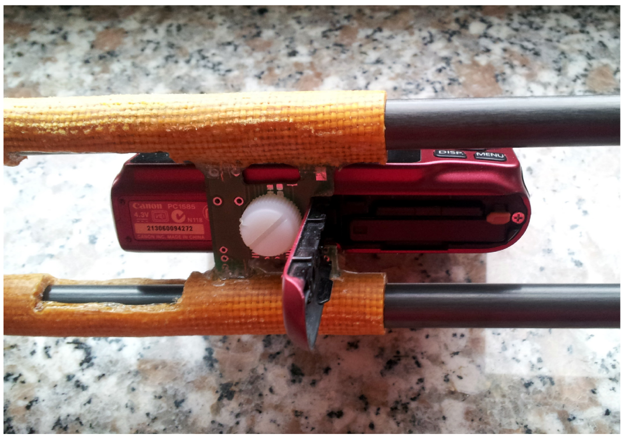


Figure 8: Camera cradle for the stereo rig

5 Software

Our autopilot with ground communications and sensor logging was implemented as an Android application. This allowed usage of the underlying Android APIs, making the development easier.

Using Android's Motion Sensors API, which performed sensor fusion on the on-board sensors, the orientation of the device was obtained according to the device's coordinate system. By strategically placing the phone on the model, the relative orientation and direction of the model was made in the form of angles around the device's axis.

The GPS data, location, altitude, and speed of the device were collated using the Android's Location API. The API also allowed navigation tasks to be performed like calculating the distance or heading to a specified point, which the autopilot had to reach.

The implementation of the autopilot was designed around a Proportional Integral Derivative feedback loop or a PID controller, which takes the device's current orientation and desired orientation as input, and provides servo rotation as output. A separate PID controller for each of the models control axes was used, thus translating the device's coordinate system to yaw, pitch and roll as well as throttle.

The autopilot was developed and tested using a FlightGear flight simulator, which can simulate the sensors found in the phone. In order to simulate the flight, FlightGear's two-way network capabilities for sensor input and servo output was used, as required and produced by the autopilot application. Simulated flights were conducted using the FlightGear's Rascal 110 RC model. This provided a crash-proof environment for testing and tuning. A testing session with live visualization over the network from the autopilot application can be seen in Figure 9.

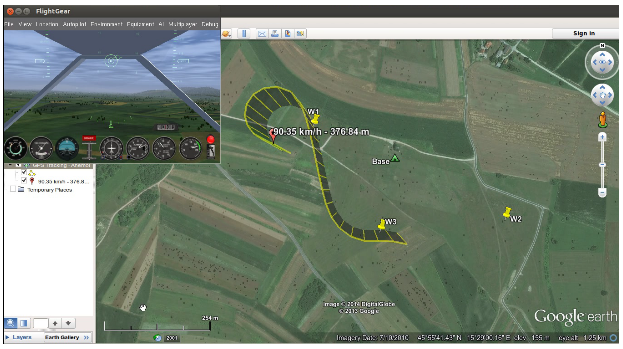


Figure 9: Autopilot testing with live visualization

The visualization was done using custom software made in Python. When using it, the logged GPS data from the autopilot is aggregated. The output is a Google Earth compatible Keyhole Markup Language structured file, containing a flight path, ground speed, altitude, GPS resolution and way points used by the autopilot for navigation. All of the sensor data was rendered into separate videos, which were later combined and synchronized with the panoramic video from the phone's camera and visualization playback from Google Earth (see Figure 10).

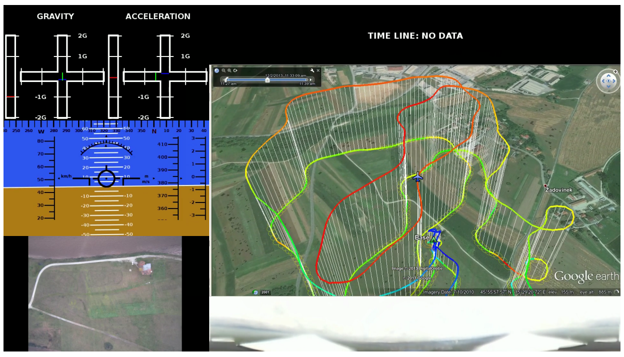


Figure 10: Visualization of flight data

Ground communication was done through the Internet using the built-in GPRS modem of the phone. GPS data from the autopilot was processed by the software and piped in KML format into Google Earth where it was rendered in real time.

Unrolling of the 360° panoramic images and video taken with the Kogeto DOT attachment was done using Log-Polar to Cartesian conversion with an added scaling factor to offset the distortion of the lens. The final image was further improved with B-spline interpolation [3]. The final results can be seen in Figure 11.

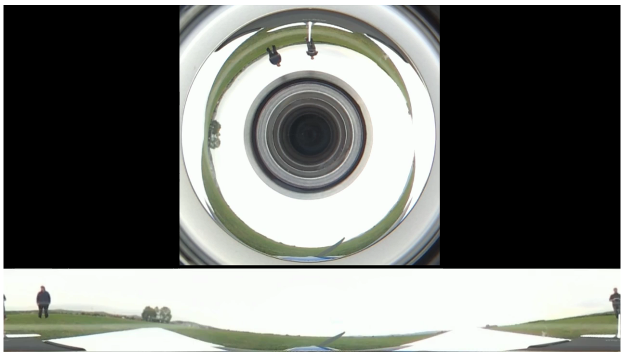


Figure 11: Raw and unrolled 360° panoramic image

6 Terrain reconstruction

From the camera rig we obtained 14MP stereo images, taken 5 seconds apart. In order to ensure adequate overlap between images, only those taken at a minimum altitude difference of 200m or more were used. The selected images were then processed to remove lens distortion detected by taking photos of calibration checkered board with each of the used cameras.

To test whether the aerial images are suitable for terrain reconstruction, the following tool chain was employed. First we used VisualSFM [1, 6] which implemented SiftGPU [5] for feature matching between images, calculated the camera positions within the world coordinate system and performed the sparse terrain reconstruction using the Structure from Motion method. That information was then used for two more applications; CMPMVS [2] and SURE [4]. In order to perform dense reconstruction, the CMPMVS application uses the augmented Labatut CGF 2009 method, while the SURE application applied an improved Semi-Global Matching (SGM) algorithm.

7 Results

A test flight covering around 170,000m^2 was performed, which generated 38 useful images (see Figure 12). After processing the images the result were obtained for the area seen in Figure 13.

Figure 12: Pictures used in reconstruction

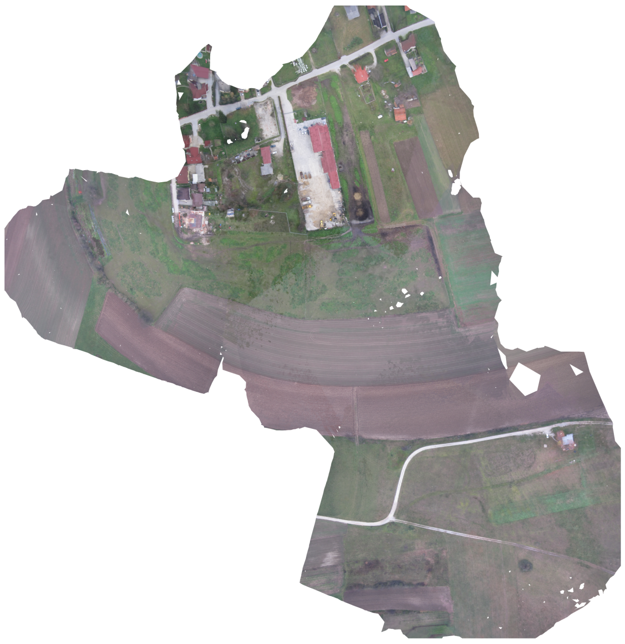


Figure 13: Ortho-photo of the reconstructed terrain

VisualSFM produced a point cloud consisting of 1 million points, resulting in a resolution of approximately 5.9 points per square meter. A section of the generated point cloud can be seen in Figure 14.

CMPMVS produced a point cloud of 2.3 million points, resulting in a resolution of approximately 13.7 points per square meter. A section of the generated point cloud can be seen in Figure 15.

SURE produced a point cloud for every image pair, which when combined form a point cloud with 43.7 million points, resulting in a resolution of approximately 257.3 points per square meter. A section of the generated point cloud can be seen in Figure 16.

8 Conclusions

The constructed unmanned aerial vehicle platform performed well enough to successfully perform initial testing, but there are of course some drawbacks. Namely, this configuration of the model requires a decent landing strip for takeoffs and landings. Battery life is an issue due to weight constraints. The current version of the autopilot would be good enough for simple flyovers and return-to-home, but not stable enough for precise maneuvers to acquire clear images.

Future work on this platform will concentrate on upgrading the used airplane with a bigger model wthin a pusher configuration where the propeller is mounted behind the engine, and the engine itself will be mounted on top of the fuselage. This will remove the need for a landing gear and landing strip, since it will be possible to launch the model by hand and land it anywhere without damage. Also an update of the customized electronics with a pass-through from receiver to the phone would allow a fly-by-wire type autopilot, whereby the autopilot would adjust the control surfaces of the airplane to maintain the position input by the ground controller.

References

[1] Brian Curless Changchang Wu, Sameer Agarwal and Steven M. Seitz. Multicore bundle adjustment. In *CVPR*, pages 3057–3064. IEEE, 2011.

[2] Michal Jancosek and Tomas Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR*, pages 3121–3128. IEEE, 2011.

[3] Jernej Kranjec and Božidar Potočnik. Razvijanje 360° panoramske slike iz leče s sferičnim zrcalom (in slovene - english title: Unrolling a 360° panoramic image from a spherical mirror lens). In *ROSUS*, pages 129–136. University of Maribor - Faculty of Electrical Engineering and Computer Science, 2013.

[4] Dieter Fritsch Mathias Rothermel, Konrad Wenzel and Norbert Haala. Sure: Photogrammetric surface reconstruction from imagery. In *Proceedings LC3D Workshop, Berlin*, 2012.

[5] Changchang Wu. Siftgpu: A gpu implementation of scale invaraint feature transform (sift), 2007.

[6] Changchang Wu. Visualsfm: A visual structure from motion system, 2011.

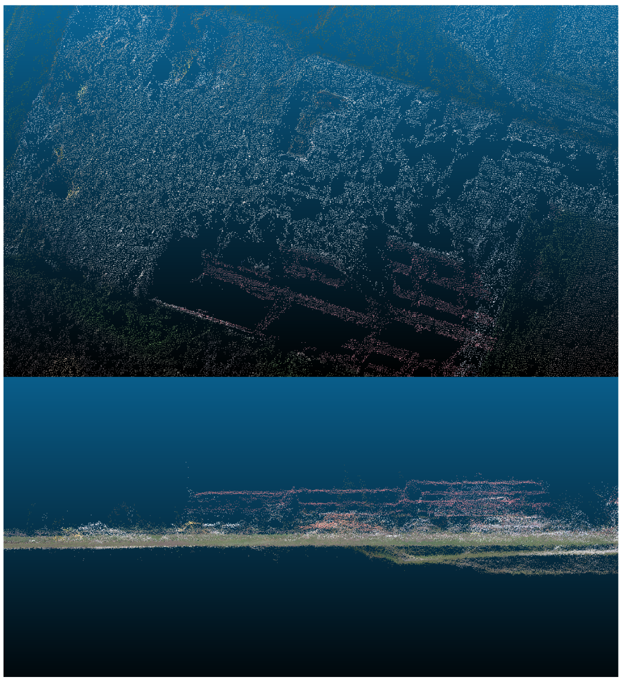


Figure 14: Point cloud generated by VisualSFM

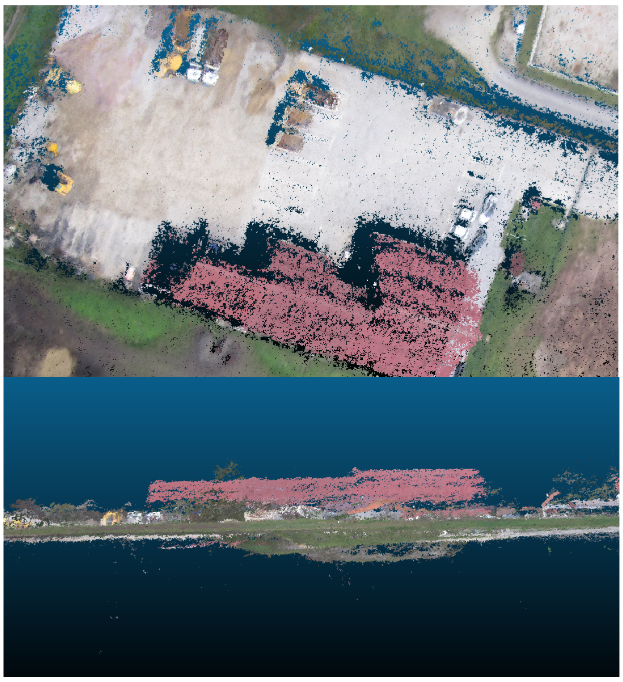


Figure 16: Point cloud generated by SURE

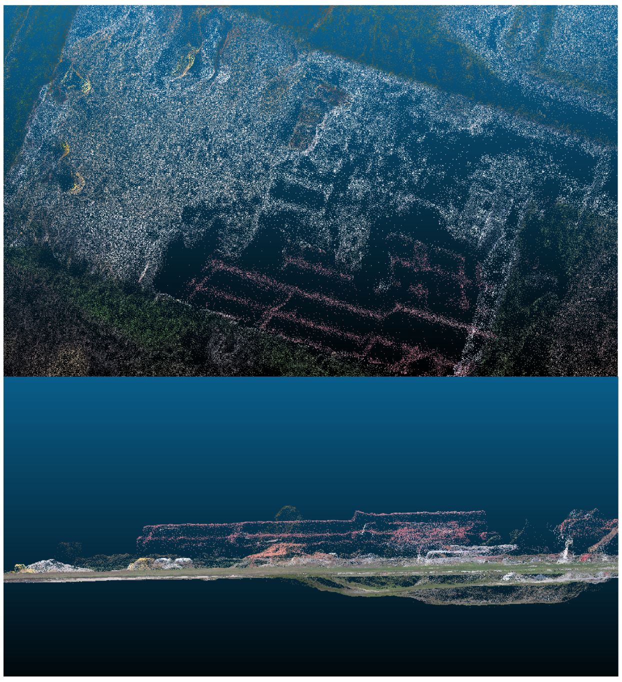


Figure 15: Point cloud generated by CMPMVS