VR Quadcopter Telepresence Proposal

Brandon Benton
Department of Computer Science
Cornell University
bnb32@cornell.edu

Figure 1: *Flying quadcopter and stereo-view rendering*

Abstract

Virtual reality (VR) lets us experience things we can't do in the real world. Demos at SIGGRAPH have shown tightrope walking above a city, hanging out with dragons, and even flying like a bird. But what if VR could let us truly experience flying through the real world? In this work we propose to develop a tightly integrated low-latency system to enable a quadcopter to be piloted using a virtual reality interface, letting the user immersively inhabit the vehicles point of view. Full-surround visibility is provided by orienting the vehicle to track the users head, and stabilization and latency hiding are provided by a wide-field-of-view camera array coupled to a high-performance view synthesis engine that decouples the view shown to the user from the pose of the vehicle. Challenges include latency compensation; artifact-free view synthesis from depth and video; stabilizing video with imperfect motion tracking; and using video from fast-moving cameras. Surmounting these challenges and achieving an authentic immersive experience will require going beyond combining available hardware components and existing algorithms. Rather, we will take a minimalistic approach to hardware selection and primarily focus on maximizing performance of low-cost platforms. Taking advantage of vehicle and head pose data streams, motion prediction and image warping of sparse image sequences will be exploited to optimize required data transfer and onboard hardware needed for immersive environment reconstruction on the ground.

1 Introduction

Over the past decade there has been a surge in production of low-cost single board computers (SBCs) and small sensor packages. This has opened up a world of possibilities for designing various compact systems capable of performing sophisticated vision/learning/robotics tasks. Exploring the range of such systems is increasingly important to fully recognize limitations of individual components and neccessary improvements to software pipelines. This technological evolution has been a boon to both the development of high-quality VR systems, such as the Oculus Rift (figure 2), and small multi-rotor vehicles. Multi-rotor development has been ongoing for decades but only now have low cost multicopter systems relying on efficient onboard computingbecome possible. This new climate is marked by widespread multi-copter use in aerial photography and mapping [Siebert and Teizer 2014], vision-based aerial inspection [Bodla et al. 2014; Kuo et al. 2013], autonomous high-altitude navigation [Rengarajan and Anitha 2013; Raj and Katiyar 2014], and even autonomous navigaion in low altitude at close proximity to obstacles [Scaramuzza et al. 2013; Meier et al. 2011a; Agrawal et al. 2007].

Although research on quadcopter navigation and flight has been significant the focus on integrated flight viewing systems has been limited. Attempts to create immersive flight experiences do exist within the growing RC/DIY/Maker community [Benchoff 2013; Mikkelsen 2013], but have not achieved low-latency, mostly use analog video streams for limited real-time flight viewing, use servo actuated cameras for view direction changes, and rely mainly on a bolt-together approach providing a low-fidelity experience with restricted domain of comfortable use. The goal of this project is to harness minimal commodity hardware letting a user pilot a quadcopter via a VR interface that immerses them in the real world with the point of view of the vehicle. In contrast to approaches in the hobbyist community, we will mainly employ techniques on the software side to provide an optimally integrated immersive viewing system. Multi-rotors are agile vehicles which can reach high speeds, are capable of performing complicated maneuvers, and can cover a wide range of perspectives during a single flight. Special focus on this wide range of motion we be leveraged to povide more direct human-vehicle integation. Development of a more extensive environment sensing system than previous platforms for mobile control and remote view reconstruction [Kelly et al. 2011] will also be required.

Our main contribution here is a systems solution (hardware, software, and algorithms). The system includes an autonomous quadcopter outfitted with an array of cameras, a low-bandwidth radio link to the ground, and a VR application that runs on a high-end GPU, using a real-time view interpolation engine to generate stabilized views that track the user's head motion at VR rates. The user will be free to look around the vehicle's environment, with the view interpolation system absorbing the lag between head motion and vehicle motion. This is an ambitious engineering project which will bring together several technologies: view interpolation, video stabilization, and flight control will have to be tightly orchestrated, with careful attention to latencies and bandwidths in every part of the system. Simply bolting together existing methods for these three subsystems will not suffice to meet the extreme performance demands of VR and vehicle control. Novel research will

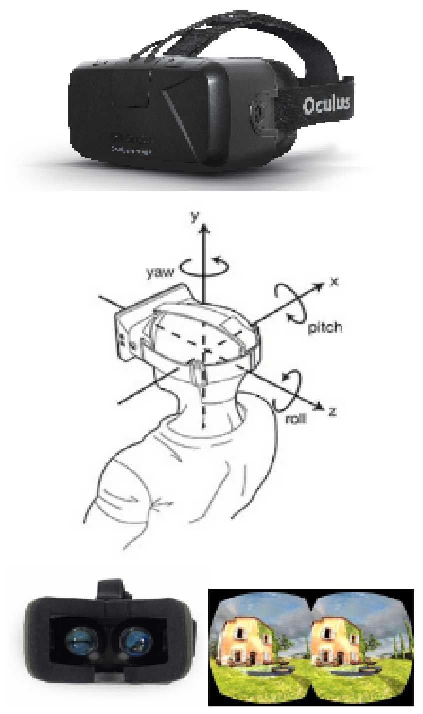

Figure 2: *Oculus Rift Headset, Coordinate System, and Tuscany Demo Example*

be required to build an integrated system that streams selected pixels through a highly constrained communication link, uses them to asynchronously update a model sufficient for view synthesis, and synthesizes views for the user's two eyes in a tight, low-latency loop.

2 Prior Work

Some of the key technologies required for this project are head-tracked stereo display, real-time stereo, real-time view synthesis, environment reconstruction, video stabilization, and aerial vehicle control. While there has been much work focused on these areas in isolation, there has been far less attention on optimal integration for real-time application.

2.1 Head-Tracked Stereo Display

Head-tracked stereo display will be handled by existing technology, and the Oculus Rift VR package is a prime candidate for this part of our pipeline (figures 2). The Oculus system uses low-cost MEMS sensors for maintaining human head orientation, with gyroscope integration and compensation of dead reckoning, tilt, and yaw errors using gravity and magnetic fields of the Earth. Predictive tracking methods are used to dramatically reduce latency and improve user experience. Predictive tracking employs constant acceleration kinematics for up to 50ms forecasting with imperceptible error, using only a few milliseconds of past data [LaValle et al. 2014]. Lower fidelity options also exist, such as Google Cardboard, which uses an Android-based smartphone to perform head-tracked stereo rendering. As with the Oculus, low-cost sensors, which are widely used in smartphones, are harnessed for head-pose tracking. A more compact system, integrating a smartphone-based rendering engine, is worth exploring for future work.

2.2 Real-Time Stereo

A great deal of work has been done on real-time stereo reconstruction; the plane-sweep algorithm is a popular method in real-time applications [Pollefeys et al. 2007], known for its simplicity and ease of parallelizability. The primary operation of the algorithm, rendering images onto planes, is an operation at which the GPU is particularly adept. Recent work demonstrates a multi-view plane-sweep-based stereo algorithm which correctly handles slanted surfaces and runs in real-time on the GPU [Gallup et al. 2014]. This involves determining the scene's principal plane orientations, estimating depth by performing plane-sweep for each direction, and combining multi-directional results. Other interesting work is shown in [Yang and Pollefeys 2003], which combines GPU acceleration and multi-resolution stereo to enhance quality of pixel matching and reduce artifacts from depth discontinuities. The algorithm described runs in real-time and would be well suited for our GPU equipped Jetson SBC. Stemming from easy extension to multi-baseline stereo this method would also scale well with additional cameras. Similarly, following a plane-sweep approach extended by truncated summed-squared-difference (SSD) scores, shiftable windows, and best camera selection, real-time pixel-wise depth value estimation has been demonstrated [Woetzel and Koch 2004]. This work relies on multi-stereo input, and through minimizing aggregated per-pixel matching cost for different views maximally probable depth values for a given cameras projection center are computed. Truncation of the SSD scores is used to limit the influence of outliers due to image noise and non-diffuse surfaces. This is accelerated on the GPU by exploiting independence of pixel intensities and achieves rates of 20Hz.

2.3 Real-time View Synthesis

Approaches for determining a new view for a virtual camera are quite varied, although warping, or transforming, cached frames is a common component. Accounting for arbitrary trasformations can be done with image-based rendering (IBR) techniques [Zitnick et al. 2014], projective geometry methods [Saito et al. 2002], using dense image correspondences [Lipski et al. 2011], with disparity maps [Naemura et al. 2002; Chen and Williams 1994], or by exploiting a few linear correspondences [Avidan and Shashua 1998]. As previously demonstrated [Chen and Williams 1994], known camera transformations and range data can be used to automatically establish pixel correspondences within overlapping camera views and subsequently generate associated pixel offset vectors. Linearly interpolating these offset vectors and mapping pixels in accordance with these offsets provides an approximate intermediate view. This approach, referred to as ”forward mapping”, is simple to implement and allows for immediate interpolation once disparity maps are calculated, but can result in significant holes around depth discontinuities. On the other hand, ”backward mapping”, as described in [Martin and Roy 2008; Laveau and Faugeras 2004], could prove more successful particularly in applications involving live camera feeds. In a similar vein, [Rodriquez et al. 2006] describes a video compression algorithm where camera homographies are derived from known vehicle motion and used to construct plausible future camera images. Early work, focused on virtual scene reconstruction [Szeliski and Kang 1995], describes a simple image stitching approach for calibrated cameras. This also relies on 2D image transformations and minimization of differences in pixel intensities for alignments, and a bilinear weighting function for blending images together. This work also describes a method for recovering projective depth from structure from motion. Both of these contributions will prove helpful in performing model construction and view interpolation.

2.4 Environment Reconstruction

In our proposed system view synthesis and environment reconstruction are closely coupled, with much overlap in the literature. Environment reconstruction can be achieved by simply generating a coarse 3D model, which can exploit depth data from stereo imaging. However, avoidance of artifacts in areas with unreliable depth is better developed in offline systems [Chaurasia et al. 2013]. In this previous work an approach using oversegmentation of input images, creation of superpixels of homogeneous color content, and depth-synthesis from traversal of the superpixel graph structure, performs well for poorly reconstructed regions. Additionally, there has been some work demonstrating parallel scene reconstruction and view synthesis using the plane-sweep approach [Yang et al. 2002]. However, this relied on calibrated reference images and distributed computing with extensive hardware requirements. 3D reconstruction with free-viewpoint synthesis can also be performed using the image-based hulls method [Matusik et al. 2000] combined with inter-frame prediction to perform viewing updates [Wurmlin et al. 2003]. This previous work exploits spatio-temporal coherence of multiple cameras for acceleration and error correction. Reconstruction with integrated treatment of depth artifacts has also been demonstrated [Merrell et al. 2007], wherein depth estimates for each pixel are selected which minimize violations of visibility constraints. Some quite promising work has recently performed volumetric 3D mapping in real-time on a dual-core CPU [Steinbrucker et al. 2014]. Here an octree data structure is used, which allows representation of the scene at multiple scales and growth of the reconstruction volume dynamically. This work also reduces memory footprint and accelerates data fusion. The work performed in [Kelly et al. 2011] describes a practical implementation for scene reconstruction using some of the aforementioned methods. Here range data from a LIDAR sensor is triangulated to construct coarse scene geometry and realistic textures captured from an onboard camera are then projected onto the scene. However, LIDAR provides reduced depth and temporal resolution which may be suited for a slowly moving ground vehicle but inadequate for our platform.

2.5 Video Stabilization

A particularly interesting work on video stabilization is featured in [Liu and Jin 2014]. This uses structure from motion to construct a 3D camera path and least-squares optimization to compute a spatially varying warp to match cached images and avoid deforming content within reference images. Being that our pipeline will include processes that provide knowledge of camera trajectory, and view synthesis will rely on reference image warping, a similar method would integrate well. Similarly, a robust motion estimation algorithm is used to construct geometric transformations which ensure video stability on a micro aerial vehicle in other recent work [Aguilar and Angulo 2013]. Additionally, work described in [Stupich 2014], could also prove helpful. Here an algorithm for modelling and correcting video artifacts caused by movements of rolling shutter cameras is described. Affine transformations are used to model full frame camera movements, equivalent to a warping approach, and sinusoids model high frequency camera movements and vibrations.

2.6 Vehicle Control

Low-cost flight control hardware packages are also seeing widespread development, similarly harnessing sensor systems used in smartphones. A prime example here is the Pixhawk flight management unit (FMU), an opensource system developed with primary focus on tight integration of vision processing within the vehicle control loop. This FMU has demonstrated the ability for parallel image processing along with interial measurement information fusion to perform tasks such as localization, pattern recognition, and obstacle avoidance [Meier et al. 2011a; Meier et al. 2011b]. This existing hardware is selected to manage flight control of our vehicle. There has also been some development on remote-control interfaces for driving robots, including systems that provide latency-compensated first-person views, using an approach generally similar to the proposed system [Kelly et al. 2011]. However, as this used a large slow-moving ground rover little attention to weight considerations was required. We propose to provide an immersive experience for the operator, rather than simply a first-person view on a monitor. Our faster-responding vehicle will require smaller deviations between virtual and real viewpoints, enabling the system to work with lightweight sensors that can fly on a small vehicle and to achieve the demanding performance needed to create the illusion of immersion.

3 System Overview

A block diagram of our system is shown in figure 3. Physically, it has two parts: a UAV carries a camera array, an onboard computer, and a flight controller; on the ground there is a high-end graphics workstation connected to an Oculus VR system. The two parts are connected by two radio links: a video downlink and a low-bandwidth bidirectional link for control data. The hardware options for handling communication links and for the onboard SBC, as discussed previously, have a wide price range. Additionally, cameras in the viewing system are not limited to the standard RGB variety, and could include the increasingly more prevalent and compact depth cameras. The viewing system needs to provide environmental information neccessary for model construction on the ground, but the combination of components able to provide this data is not unique. Narrowing in on a particular configuration with focus on optimal weight, power, and bandwidth requirements will be one of our challenges.

Each part of our system has a fast loop that handles low-latency interaction with the real world. The UAV runs a 500 Hz flight control loop in separate hardware that communicates with the onboard computer via a serial link; the GPU-accelerated view synthesis loop feeding the VR headset runs at 75Hz. The quoted rates are from the Pixhawk flight controller and Oculus Rift VR system but subject to variation with different hardware selection. The two fast loops are connected by a slower asynchronous system that has the job of carrying video data from air to ground while absorbing the latency between them. The onboard computer reads video data from a subset of cameras, selects the most relevant parts subject to bandwidth constraints, and transmits the data to the ground. On the ground side, our scene modeling system uses incoming video and vehicle pose to build a weak 3D model of the scene and caches the images for use by the view synthesis system. The ground system also uses head-tracking data from the VR system to determine requested vehicle motion and transmits these commands back to the UAV.

Depending on the carrier signal and equipment used for video transmission, streaming image sequences can suffer air-to-ground time delays ranging from near zero (sub-millisecond) up to around a hundred milliseconds. This upper limit of around 10Hz is in contrast to most available small cameras, which have minimum capture rates of around 30Hz. With a vehicle speed around 10m/s and delay of 100ms, we require data transmitted to ground be sufficient for reconstructing an environment model with a minimal radius of one meter. Thus, our camera array will have to capture views at multiple angles and provide depth data as well. On the other hand, our low-bandwidth radio link provides vehicle pose data at 30Hz.

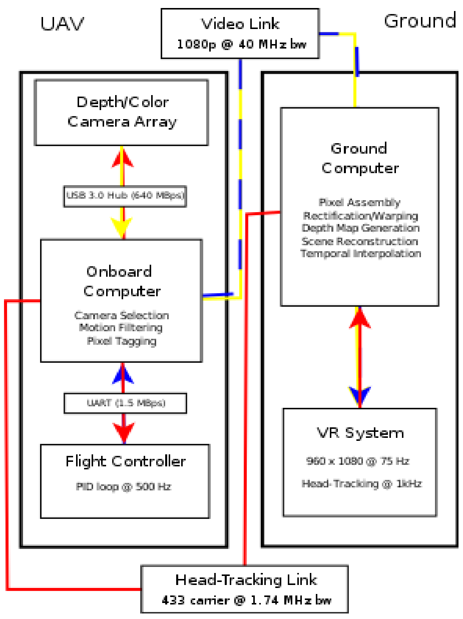

Figure 3: *System block diagram: Video (Yellow), Head-Tracking (Red), Vehicle Pose (Blue)*

Along with control input transmission, based on head-tracking data, vehicle pose streamed to ground enables derivation of view transformations using orientation and position changes relative to previously cached states. Our onboard SBC will have to select appropriate cameras based on head-tracking data, cache images and apply time-stamps, and possibly provide depth data to ground as well. Whether depth map construction is performed on the ground or onboard depends on results of future experimentation. The ground system will process this data, which will need correction for artifacts from gaps in environmental information and rolling-shutter effects from fast-moving cameras. Additionally, model construction, view synthesis, and possible depth map generation will need to be performed within a time interval on the order of 100ms. Stabilization of video, despite imperfect motion tracking, will also need to be performed. Filtering out motion using vehicle pose data will provide a coarse initial treatment but likely need further refinement within our view synthesis engine. High-performance graphics hardware has been shown to accelerate depth-map generation and scene reconstruction, acheiving combined times around 40ms [Merrell et al. 2007].

4 System Hardware

Extensive research on available hardware is required to achieve desired rendering rates. While finding a combination of hardware able to provide an immersive flight experience of reasonable quality may be possible with extensive budget, we opt for low-cost commodity components. The main objective here on is achieving the best performance possible from a limited budget and guidance from theoretical considerations.

Flight Control Unit: The flight controller is the quadcopter brain, in charge of running a PID loop to correct vehicle attitude based on deviations of sensor determined state from control inputs. The price range and compute power available in these units varies significantly, with quite limited controllers, in the $20-$50 range, dedicated strictly to flight control for hobbyist use and others, in the

Components	Specifications
Microprocessor	32-bit STM32F427 Cortex M4 core with FPU
	168 MHz/256 KB RAM/2 MB Flash
	32-bit STM32F103 failsafe co-processor
Sensors	ST Micro L3GD20 3-axis 16-bit gyro
	ST Micro LSM303D 3-axis 14-bit accel/mag
	Invensense MPU 6000 3-axis accel/gyro
	MEAS MS5611 barometer
Interfaces	5x UART (serial ports)
	2x CAN
	Spektrum DSM/DSM2/DSM-X
	Futaba S.BUS compatible I/O
	PPM sum signal
	RSSI (PWM or voltage) input
	I2C
	SPI
	3.3 and 6.6V ADC inputs
	External microUSB port
Power System	Ideal diode controller with automatic failover
	Servo rail high-power (7V) and high-current ready
Weight & Dimensions	Weight: 38g
	Width: 50 mm
	Thickness: 15.5 mm
	Length: 81.5 mm

Table 1: *Pixhawk Details*

$500-$1000 range, employing multiple inertial navigation systems for redundancy and highly tuned for video stability. Our needs include extra computing power for our custom software, an open-source package, and interfaces for additional onboard hardware. We select the $200 Pixhawk flight management unit (FMU) [Meier et al. 2011a; Meier et al. 2011b], intially developed with primary focus on tight vision sensor integration. The FMU runs a PID loop at 500Hz with sensor fusion performed using an Extended Kalman Filter. Onboard sensors include a barometer, two accelerometers, and magnetometer, which provide data for vehicle pose estimation. A 1.5 MBps bandwidth UART interface on the FMU enables sending vehicle pose data to the onboard computer at up to 60Hz. To our knowledge, this FMU provides the most versatile package for relatively low cost. Refer to table 1 for more interface and processing details.

Camera Array: Determining the required components for our viewing system will be closely coupled to the development of our view interpolation and environment reconstruction pipeline, and optimizing for minimal required components is one of the challenges of this project. Following previously mentioned back of the envelope calculations, we are on the line of where depth cameras can be useful. The maximum range of these devices is typically on the order of a few meters. Small depth cameras are a relatively new, still developing, technology and hold promise for providing dedicated components for capturing depth data need for environment reconstruction. Due to their more limited range of utility we will start with a single stereo camera (ZED), each sensor fitted with a fisheye lens, and scale up from there depending on results. However, depth cameras will be explored further for higher resolution reconstruction at close range. The ZED camera boasts up to 20m range due to its 0.1m baseline. Operation on USB 3.0, which is true of increasingly more commodity cameras, provides a 10x data transfer speedup over legacy USB 2.0.

Onboard Computer: Our onboard computer will need to handle moderate image processing, control over the camera array, timing for synchronization, and provide interfaces for other onboard electronics. To take advantage of the increasing availability of USB 3.0 compatible cameras, providing higher speed and bandwidth, this interface is desired over USB 2.0. CUDA compatibility is desired for image processing, along with its inherent versatility for other computation. We select the Jetson TK1 board, providing superior performance among single board computers (SBCs) for the low cost of $200. This SBC is equipped with a USB 3.0 port, 2.5GHz quad-core ARM15 CPU, and 192 CUDA core Kepler GPU with a maximum memory clock rate of 924MHz. Considering the speeds on the SBC processing units, the TK1 should provide ample compute power for pixel selection and vehicle pose tagging at the desired rates. The ZED camera SDK includes depth-map generation previously demonstrated in real-time on the Jetson TK1 [Azzam 2015]. Depending on the specific pixel sampling approach this SBC should also leave room for additional in air image processing, possibly for partial motion blurr elimination, before streaming pixels to the ground.

Wireless Data Links: Equipment for video, vehicle state, control, and head-tracking data is required for our system. The video data link is primarily established using analog video transmission equipment, as dedicated digital video equipment comes at a large price. The analog approach, however, presents time delay challenges and low resolution imaging. The Connex HD Video Downlink package operating on 5.8GHz carrier signal provides a video link capable of 1080p resolution at 60fps with sub-millisecond latency, but comes with a $1500 price point. Without dedicated digital video equipment the time delay constitutes a possible botteneck in our pipeline, whether using analog or alternatively short-range WiFi. Initial approaches here will focus primarily on using vehicle and head motion prediction and subsequent future view estimation for latency compensation. WiFi and analog options are already available in our quadcopter build, so these will be explored in union with predictive view synthesis. However, from the previous considerations this part of our system will likely require experimentation with alternative hardware options. Bi-directional telemetry transmission is typically perfomed using radio modules operating around 915 MHz. For this component we would like update rates faster than video in order to facilitate vehicle trajectory interpolation for view-synthesis. We also need to accomodate head-tracking data transmission, but this presents only a small bandwidth requirement. We select the 3DR telemetry package, providing radio modules compatible with the Pixhawk FMU and capable of a 30Hz update rate.

Ground Station: On the ground we will need to construct our environment model, synthesize new views, correctly synchronize views with motion, and fuse data streams to predict virtual camera poses. Access to a high-performance workstation is already available, which is equipped with a Titan GPU, 64 GB RAM, and two Intel i7 CPUs providing 32 total virtual cores. Prior tests on this worksation show that rendering rates well above the maximum rates possible for the Oculus Rift Headset are easily achievable. Given this hardware platform, we will allocate separate cores and threads for simultaneous data fusion, timing and synchronization, model construction with asynchronous updating, and rendering to our VR headset.

5 System Software

There exist a number of software packages that will prove helpful in development of our complete system. We will have a number of separate processes, both in air and on ground, and data streams coming coming from wireless links and head-tracking. On ground data streams will need to be fused in order to predict virtual camera poses, correctly track quadcopter trajectory, and to calculate required head-pose based control inputs. Additionally, each separate software layer will need to be tightly integrated and all processes monitored. Opensource solutions are desired for customizability and portability.

Flight Control Firmware: Our flight controller requires appropriate firmware to run the PID loop and interface with our onboard SBC. We also want the ability to integrate custom software with firmware. The Pixhawk flight controller is compatible with a few different firmware options, the most notable being PX4 Autopilot [Meier 2012] and ArduCopter [Osborne 2012]. Both of these flight stacks are opensource and run ontop of the native NuttX RTOS, which provides a UNIX style minimal operating system and uses the MavLink protocol to communicate with ground control software. Currently the ArduCopter firmware has been selected, as it is compatible with our choice of ground control software the platform portable APMPlanner2.

Middleware Layer: Our onboard SBC requires a layer to interface with the flight controller and oversee the various processes composing our airborne subsystem. Likewise, onground processes need to be monitored and separate data streams fused to determine head-pose based control inputs. ROS (Robot Operating System) provides an extensive library and toolkit for developing robotic applications [Quigley et al. 2009]. In particular, ROS provides many tools for message-passing, simulation, and visualization which will prove useful for unifying pose data from sensors in separate parts of our system under a single framework. We will use ROS to run an onground PID loop, and fuse head and vehicle pose data, to track camera ego-motion and calculate necessary flight corrections based on head-tracking signals. Camera array calibration will also be simplified using ROS.

VR Software: We will need a software layer for the VR system to handle general head-tracking data, head-pose prediction, and perspective-correct rendering for 3D stereo viewing. There are two software candiates for handling stereo display and exposing head-tracking data. The Oculus SDK could certainly accomplish this, but it also presents limitation concerning platform portability. OpenHMD is an opensource alternative which is worth exploring. This is not as well tuned to the Oculus VR system but reduces restrictions on operating system selection for our ground workstation.

Rendering & Simulation: Model construction will need direct access to depth and image data acquired onboard as well as virtual camera orientation updates. The construction process will be made easier using available libraries for handling lighting, geometric primitives, and incorporating physics of our vehicle. Additonally, rather than relying completely on physical testing, simulation software which provides accurate flight physics and hardware-in-loop interfacing can be leveraged. Gazebo is an opensource robotics simulation framework, with an integrated physics engine specifically suited to our focus [Koenig and Howard 2004]. This framework, like ROS, is supported by the Open Source Robotics Community and the two can be seamlessly integrated. Rendering is handled by OGRE with particular focus on physically accurate simulation, in contrast to more perceptually focused game engines.

Components	Model
Flight Controller	Pixhawk
GPS Receiver	3DR NEO-7
Motors	Lumenier FM4006-13 740KV
Electronic Speed Controllers	Lumenier 30A SimonK Firmware 5V/3A BEC
Radio Receiver	FrSky X8R 8/16CH 2.4 GHz
Telemetry Module	3DR 433 MHz
Video Transmitter	Fatshark 5.8 GHz 600 mW
Propellers	APC 12” with 6” pitch
Analog Camera	Lumenier CMOS 720TVL
Frame	Lumenier QAV400 G10
Arms	Lumenier QAV520 G10
Sonar	MaxBotix XL-EZ4
Flight Battery	Lumenier LIPO 14.8V 3300mAh
Video Battery	Lumenier LIPO 12V 1250mAh
Computer	Jetson TK1

Table 2: *Quadcopter Components*

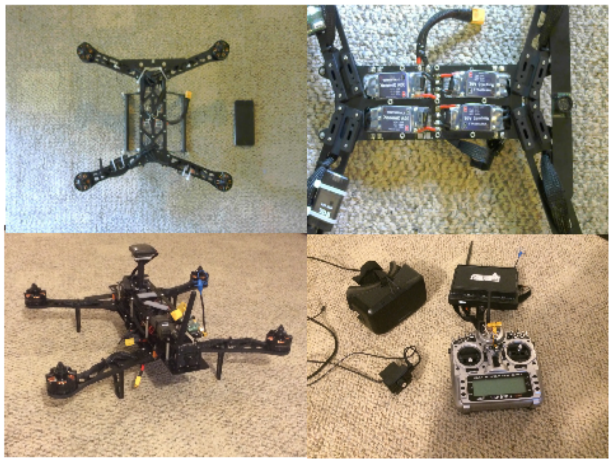

Figure 4: *Quadcopter Frame, Powertrain, Complete Build, and Ground Gear*

6 System Construction and Setup

For this work a consumer-grade small quadcopter has been built (figure 4, bottom-left). This used low-cost commodity powertrain components (motors/ESCs), flight control hardware, and video data transmission equipment. The frame (figure 4, top-left) consists of high-strength laser-cut G10 plastic, and a closer look at the powertrain wiring is shown top-right in figure 4. Using a small consumer GPS module, remote motor control inputs can be converted into position control inputs, enabling trajectory planning and fail-safe functionality such as return-to-launch mode. Combined cost of quadcopter components came to around $1000 and with build completition taking around a week. Refer to table 2 for build component details. Relatively simple components can be used for remote control and perspective-correct viewing, shown bottom-right in figure 4. Some additional soldering and electrical wiring was required for completing the powertrain. Motors are individually wired through Electronic Speed Controllers (ESCs) which are soldered to a central power distribution board. This board serves as the frame foundation, also taking power input from the flight and auxillary batteries and providing power to all onboard electronics. The Pixhawk flight controller provides a number of serial interfaces through which the onboard SBC can communicate. We have already tested this interface with currently owned SBCs, and connection is achieved through either USB connection, with an FTDI converter, or directly to a UART port on the SBC. This provies a 1.5 MBps link for direct sensor data transmission to the onboard computer to be used for further image processing and filtering. Cameras can be directly connected to the SBC USB ports, although scaling up the camera array will likely require a multi-port hub or switching interfaces. Synchronization is achieved either on the SBC through software or directly on the camera if stereo packages are used.

6.1 Quadcopter Tuning

Powertrain components were selected to keep the vehicle on the smaller side while maximizing lift capability and hover time. 740KV brushless motors and 30A ESCs, with built in 5V/3A battery eliminator circuits, from Lumenier were selected and frame arms were extended to accomodate 12 inch propellers. With 6 inch propeller pitch this platform is capable of lifting approximately 3kg with a 15 minute hover time. These values are determined from analysis similar to that seen in [Fay 2001; Gibiansky 2012], with the help of the online tool ”eCalc” [Meuller 2004]. Throttle curves are

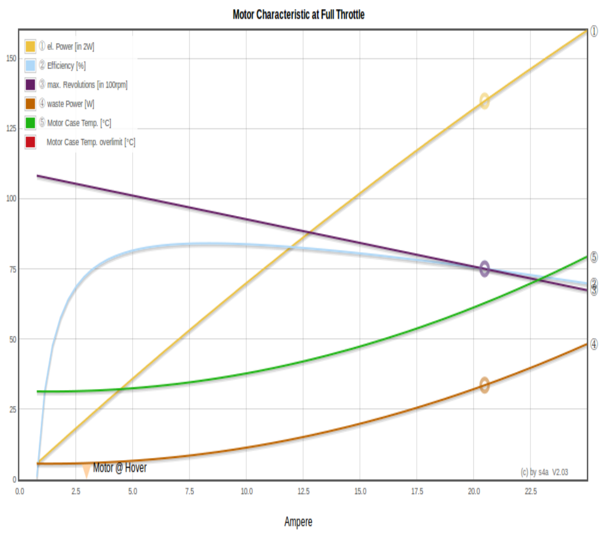

Figure 5: *Throttle curve plot*

shown in figure 5. For stable flight our quadcopter runs a PID loop on the Pixhawk FMU. The PID parameters require precise tuning, and baseline values were estimated using automatic tuning functionality within Ardupilot firmware. In a controlled environment, various vehicle orientations are explored and tuning parameters are corrected based on deviation from sensor determined pose.

6.2 Multi-camera Calibration

Although initially we will start with a single stereo camera (ZED), providing a full range of view will require additional monocular cameras in our array. As more components are incorporated into our viewing system precise calibration will become increasingly important. There has been much work focused on multi-camera calibration [Heng et al. 2014; Pless 2003; Li et al. 2013; Clipp et al. 2014], and a set of dedicated tools developed in [Heng et al. 2014] have been made available through ROS. Calibration using these tools is straight forward, relying on detection of a checkboard pattern, calculating camera instrinsics based on observed distortion in straight lines, and subsequent rectification of images. The ROS middleware provides a communication layer for handling image processing and pose estimation, which we will use to unify our image and both vehicle and head pose data acquisition processes. Furthermore, to maximize the FOV of our camera array we would like to minimize the amount of overlap between them. This is an understanable objective in order to take advantage of the limited real-estate and lift capacity on the quadrotor. The work done in [Clipp et al. 2014] demonstrates a minimum solution to relative stereo camera poses with small FOV overlap. This is achieved using limited spatial correspondances between different camera views with multiple 2D temporal correspondances. This work also demonstrates utility in determining structure from motion in real-time, and thus could prove helpful in optimizing the configuration of our camera setup as well as correcting for incomplete depth information.

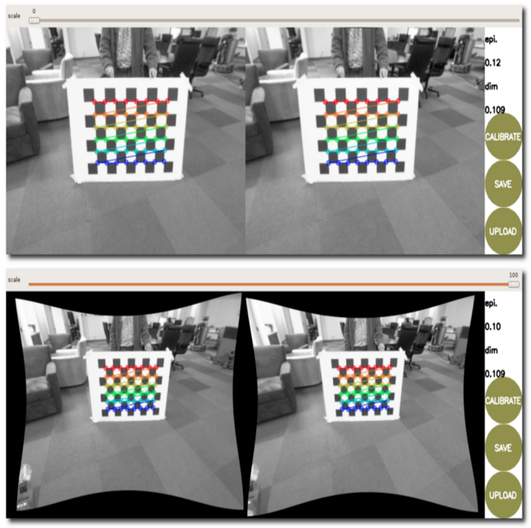

Figure 6: *ROS Camera Calibration*

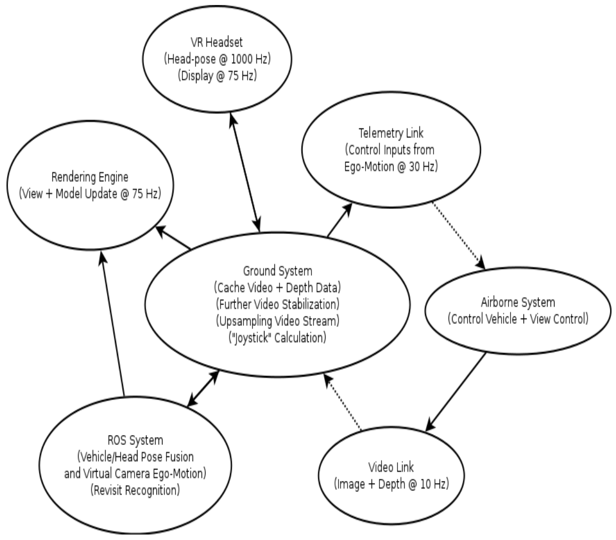
VR Headset
(Head-pose @ 1000 Hz)
(Display @ 75 Hz)
Telemetry Link
(Control Inputs from Ego-Motion @ 30 Hz)
Rendering Engine
(View + Model Update @ 75 Hz)
Ground System
(Cache Video + Depth Data)
(Further Video Stabilization)
(Upsampling Video Stream)
("Joystick" Calculation)
Airborne System
(Control Vehicle + View Control)
ROS System
(Vehicle/Head Pose Fusion and Virtual Camera Ego-Motion)
(Revisit Recognition)
Video Link
(Image + Depth @ 10 Hz)

Figure 7: *Diagram of Ground Subsystem*

7 View Interpolation Loop and Ground Subsystem

7.1 Ground Subsystem

The ground subsystem will manage a number of separate processes which are closely coupled to view interpolation. These include model construction from depth and images, pose prediction with view synchronization, and asynchronous scene updates. Additional processes running on the ground station including head-pose prediction, calculation and transmission of head-tracked control inputs, and video stabilization are discussed in subsequent sections. Data transmitted from the vehicle will be cached with associated time, vehicle-pose, and head-pose, for asynchronous processing. In this section we assume depth data is generated in the air although this could be shited to the ground without problem.

Model Construction: We pursue 3D model construction, as opposed to purely image-based construction, to aid in latency absorption from transmission delays. Predictive tracking of camera motion through a 3D scene will enable intermediate view synthesis at desired VR rendering rates. Exploiting depth data from stereo imaging will provide means to at least a weak geometric representation of our scene. As mentioned in section 2.4, the quality of our reconstruction can be improved through fusion of multiple depth maps, by exploiting coherence between sequences of frames, or leveraging coherence between overlapping images with different viewpoints. Choice of depth data structure is paramount and needs to allow for adaptive refinement, progressive updates with additional data, and low bandwidth requirement. These considerations suggest following the approach in [Steinbrucker et al. 2014], using an signed distance function and octree data structure. This approach would accomodate areas with unreliable depth information, by treating these sections with simple texture projection onto cubes. The simplicity and reasonable quality of this treatment is also shown in [Kelly et al. 2011]. We can confidently rely on in-air depth information updates at around 10Hz, due to latency of our video link, although this is a conservative estimate.

Pose Prediction & View Synchronization: Pose prediction will leverage the embedded Oculus sensors, some of which are identical to those used on the Pixhawk FMU, to provide data for fusion and subsequent pose estimation. The native predictive tracking loop will be exploited to reduce latency in vehicle response using forecasted head-pose. As mentioned in section 5, vehicle and head-tracking sensor data will be fused within ROS and PID loop outputs will provide head-based flight corrections. Ground views will need to be predicted sufficiently far in the future to offset video transmission latency. While zero latency would be ideal, we can confidently predict head-pose 50ms ahead using the Oculus system. Vehicle-pose will be predicted as well using a velocity driven dynamics model, which can similarly be run within the ROS framework. Control inputs calculated from the PID loop will be converted into joystick or radio controller inputs and sent through our telemetry link to the vehicle. Synchronization will be acheived through comparison of ground and air GPS time updates, and tracking time intervals between updates using system clocks. Precise prediction times will be determined using time-stamps on cached data and time since new view request.

Asynchronous Scene Updates: Asynchronous updates to our model will help us to meet the rendering rate requirements of our VR system and also provide a more immersive experience. Our suggested depth data structure is selected to facilitate this process. Here we will rely on analysis of cached image and depth data for progressive model refinement. Although we estimate a 10Hz transmission rate on our video link, depth-map generation and image capture can be done onboard at around 30Hz. Effects of bandwidth limitations here will need to be further explored but this could possibly provide enough cached data in a single transmission cycle from which to derive a more complete environment construction. Following a coarse construction from the most recent data, refinement can be performed using Structure from Motion (SfM) on longer data sequences. Regardless of bandwidth limitations, we will rely on this approach using available cached data. Additionally, model updates will be timed corresponding to synchronization considerations. Stored data which corresponds to previous locations or views should also be exploited. Here we could employ revisitation detection and scale invariant scene geometry matching. The revisiting problem of SLAM [Stewart et al. 2003] is that of recognizing perceptually that a vehicle has returned to a previously visited location. During outdoor flight we will solve this problem using GPS localization, although incorporating perception into this will be explored. A diagram of the ground system including use pose prediction is shown in figure 7.

7.2 View Interpolation

In addition to model construction, the bulk of our focus will be on the view interpolation system. New views will need to be synthesized using image data from multiple cameras, corresponding to yaw and roll of the user's head. Additionally, temporal view interpolation will need to be performed even as the quadrotor moves while maintaining constant orientation. Effects from fast vehicle motion and vibrations on video quality will also need to be corrected, although whether this will need to be done on both the ground and vehicle will be determined through future testing. To accomplish all this we will leverage many of the approaches described in section 2.

Motion Prediction: As mentioned in section 4, our video data link could be a potential bottleneck in our pipeline, unless expensive hardware is selected. We will approach this primarily on the software side, and attempt latency compensation using vehicle motion prediction. Provided a coarse geometric model, previous vehicle pose data and velocity driven prediction will enable interpolation of vehicle trajectory through our environment. Intermediate vehicle positions will be calculated in this way, within the ROS framework. With suitably long range depth data this approach will allow for almost complete latency reduction. However, here we are implicitly assuming a static scene during transmission delays along with a relatively open evironment. Occulusions and moving objects will have to be handled separately, although this may be reserved for future work. Head based motion prediction will also be employed to reduce latency from projection onto model geometry, camera selection, and vehicle control.

Image Stitching: To perform spatial view interpolation, that associated with changes in head-pose, we will need to stitch parts of separate camera images together. With a fully calibrated camera array, and pre-computed relative transformations, pixel correspondences can be found efficiently and the problem mainly reduces to seamlessly blending the separate images. The simple blending approach described in [Szeliski and Kang 1995], will be our starting point. The complete image will be mapped to a sphere and subsampled based on the recently predicted virtual camera to produce a perspective-correct image. This will, of course, require subsequent warping to render stereo view will correct parallax.

Image Warping: We have briefly reviewed much of the work involving reference image transformation, or warping. This general approach will be employed in multiple parts of our view interpolation pipeline. Backward and forward based mapping of images will be especially useful in cases with rotationally dominant view changes and when depth information is missing or unreliable. Artifact reduction will also benefit from the combined use of image warping and projection onto scene geometry. An optimization approach relying on comparison of views from these two approaches will likely be the most fruitfull. Composition of additional transformations could provide further video stabilization, as mentioned is section 2.5. Globally, transformations from motion will be derived from cached image sequences as well as the trajectory tracking within ROS.

8 Airborne Subsystem

The airborne subsystem will perform depth-map generation, camera selection and pixel tagging, and flight control. Although, future testing will be required to determine the optimal division of labor between the Jetson SBC and ground computer. Secondary processes will include encoding video data, possible decoding of head-tracking based control inputs, and encoding vehicle pose data transmitted to the ground.

Depth from Stereo: As discussed in section 2.2, we have a number of options for approaching this problem. We will follow the multi-resolution approach, relying on plane-sweep to perform stereo matching, which we will extend to multi-baseline stereo to leverage additional cameras. Multiple baselines will provide more robust depth-map construction, in turn reducing some of the concerns around artifacts from areas with unreliable depth information. This treatment of artifacts is similar to that with multi-view stereo and multi-direction plane sweeping. Simplicity is also a factor in pursuing this approach but we will also consider following the work on depth-synthesis described in section 2.4.

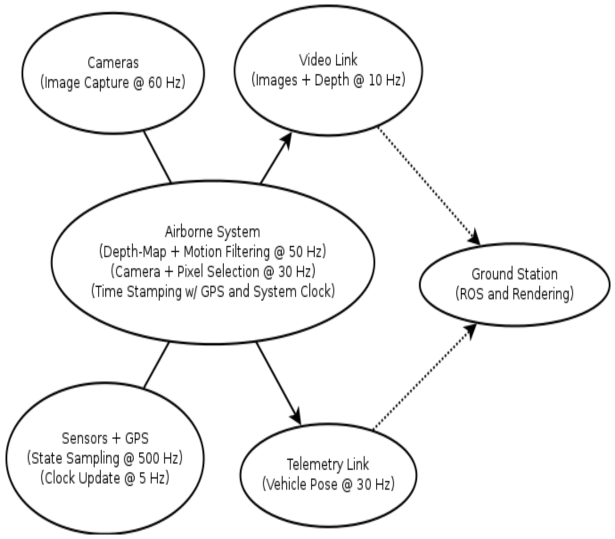

Figure 8: *Diagram of Airborne Subsystem*

Video Stabilization: With a direct serial link between the SBC and flight controller we will perform onboard warping of images based on vehicle motion. Here we will leverage the methods discussed in section 2.5, both for motion-based warping and treating possible artifacts from rolling shutter cameras. However, with our initial choice of the global shutter ZED stereo camera these particular issues can be bypassed. Onboard video stabilization will be a coarse initial treatment, with further refinment performed within the view interpolation loop.

Camera Selection & Flight Control: Virtual camera pose, predicted onground within ROS, will be used to select nearest cameras. This will need to be integrated with vehicle control to enable full range image capture. If the virtual camera pose lies outside of available view from current vehicle orientation yaw rotation will be executed. The required control inputs for this motion will be determined with the ROS-based PID loop running onground, as mentioned in section 7.1. A block diagram of the airborne system is shown in figure 8.

9 Technical Challenges & Considerations

While the technologies used in the proposed system have been developed before in isolation, achieving the performance required to make the system comfortable to use will require tight integration and innovation going well beyond the state of the art.

Reprojection artifacts: Although the system is designed to keep the camera viewpoints quite close to the users head position, nearby objects will show artifacts wherever depth information is inaccurate. Hiding these errors so that they are not distracting will be important to the overall user experience. The image-based rendering and video stabilization literature contains many techniques that work well, but will need to be adapted for real-time performance. As mentioned previously, the work in [Woetzel and Koch 2004; Merrell et al. 2007] will serve as a starting point for dealing with these errors.

Stabilization: Under normal operation of the system, the users head may be steady or moving very little while the vehicle moves slightly around its target pose as the control system responds to small disturbances. Very accurate vehicle pose tracking is available in modern UAV controllers, but we expect it will still not be accurate enough to avoid an uncomfortable swimming sensation caused by residual errors in vehicle tracking. Video stabilization, though it is usually an offline process, is quite successful at creating stability in video from unsteady cameras, with the key property that the motion to be compensated comes from the video itself. We will need to import ideas from the video stabilization literature, and close the stabilization loop by using the video itself as an additional vehicle tracking input. The work described in [Liu and Jin 2014] could provide a suitable approach to this challenge, as composing additonal affine transformations would integrate well with our proposed view synthesis method.

Fast camera motion: The ill effects of camera motion can be reduced by using a relatively short exposure, which can also help reduce system latency. However, in global-shutter cameras the last pixel to be read out must still wait almost a full frame time before it is transmitted. Overall system latency can be reduced by using rolling-shutter cameras, in which the delay between exposure time and readout time is shorter and more consistent. However, rolling-shutter cameras complicate the interpretation of the video data, since every pixel has a different time and vehicle pose associated with it [Saurer et al. 2013]. Similar to handling stabilization, an image warping based method, as described in [Stupich 2014], will be our first line of attack here.

Latency Compensation: Using motion prediction for latency absorption is likely to produce some errors, similar to those encountered in dead-reckoning based inertial navigation. An approach to handling these errors is described in [Kelly et al. 2011], wherein the real vehicle is pushed to follow the virtual vehicle associated with predicted motion. However, due to the non-linear dynamics of the quadrotor additional efforts will be required to ensure vehicle stability. Additonally, as the number of individual hardware components increased further compression of video data, along with possible changes in data structure, might be required to ensure complete data tramission from air to ground.

10 Plans for Initial Stages

Our first objective for this project is building a simplified platform, with minimal hardware. Full camera array scaling, predictively sampling cameras and pixels, and eliminating projection and motion artifacts will be performed later in the project. A simple platform using a ZED stereo camera, a DIY project quadcopter, and an oculus headset will be assembled. Initially we will seek to expose the vehicle and head pose data, with head pose data from the Oculus software loop and vehicle pose data from either ground control software or ROS middleware. Using minimal time resolution, images will be captured onboard and sent to the ground. Depth-map generation onboard will be attempted, but further testing will be required to determine if maximum performance can be acheived this way. This construction process will be shifted to the ground if necessary. Cached camera images will be transformed according to predicted vehicle and head motion, using depth map extrapolation based on predicted vehicle trajectory. We hope to achieve a coarse demonstration of view interpolation through motion prediction by the end of the year. Fisheye lenses on quadcopter cameras will provide wider FOV than the oculus screen and enable testing on varying perspective integration. The camera array will eventually be scaled up so that a FOV covering more than the half-sphere is achieved. After this aforementioned preliminary demonstration we will focus more on acounting for sampling artifacts, both temporal and spatial, and eliminating undeirable effects from extraneous vehicle motion. Refer to table 3 for a more detailed, albeit tenative, timeline for project completion.

11 Summary

We have proposed an ambitious undertaking to develop a system which can enable a pilot to control a quadcopter using a VR interface. Wide and adjustable FOV will be accomodated using a combination of careful camera sensor selection and configuration along with a low-latency view synthesis and environment reconstruction engine. Development of this engine will be guided by optimal use of low-cost camera and communication equipment while exploiting motion prediction techniques and accelerated processing using commodity graphics hardware. We have given an overview of varying approaches to hardware selection and how these individual elements will be combined for our complete system. Specific areas which we expect to provide the biggest challenges have also been described, and we have suggested some initial approaches for surmounting these challenges. Ensuring a sufficiently realistic rendering, with responsive flexibility in perspective and fast refresh rates, will be achieved through novel approaches to latency compensation and enhancements to existing approaches for error correction and video stabilization. Our initial steps will consist of completing camera integration with our already built quadcopter and establishing an accesible data link for extracting video, head-tracking, and vehicle pose data. Preliminary tests will be performed on sparse image sampling interpolation using perspective transformations based on vehicle pose updates. These tests will initially rely on sparse depth sampling as well, possibly using only a single depth map for a given test environment.

12 Acknowledgements

Thanks to my advisor, Steve Marschner, for his instrumental role in the development of this proposal and his continued support and guidance on this project, even when it started as only a vague interest in quadcopter development.

References

AGRAWAL, M., KONOLIGE, K., AND BOLLES, R. C. 2007. Localization and mapping for autonomous navigation in outdoor terrains: A stereo vision approach. *International Journal of Robotics Research*.

AGUILAR, W. G., AND ANGULO, C. 2013. Rpbust video stabilization based on motion intention for low-cost micro aerial vehicles.

AVIDAN, S., AND SHASHUA, A. 1998. Novel view synthesis by cascading trilinear tensors. *IEEE Transactions on Visualization and Computer Graphics*.

AZZAM, E., 2015. Stereo vision for autonomous machines. `http://on-demand.gputechconf.com/gtc/2015/video/S5751.html`.

BENCHOFF, B., 2013. Fpv drones with an oculus rift. `http://hackaday.com/2013/07/16/fpv-drones-with-an-oculus-rift/`.

BODLA, M., AHMED, Z., NASIR, U., ZAIDI, A., AND SALEEM, M. 2014. Design and implementation of a uav for power sys-

Task	Timeline
Integrate ZED Camera & Jetson SBC	Nov. 15 - 21, 2015
Integrate Headset & Expose Pose Data	Nov. 22 - 28, 2015
Depth Map Calculation	Nov. 29 - Dec. 5, 2015
Latency & Platform Testing	Dec. 5 - 12, 2015
Data Caching & Coarse Modelling	Dec. 13 - 19, 2015
ROS Pose Fusion & Ground PID	Dec. 20 - 26, 2015
Head-Tracking View & Control	Dec. 27 - Jan. 2, 2016
Coarse Predictive Warping	Jan. 3 - Jan. 9, 2016
Video Stabilization	Jan. 10 - 16, 2016
Camera Array Scaling	Jan. 17 - 23, 2016
Spatial View Interpolation	Jan. 24 - 30, 2016
Predictive Pixel/Camera Sampling	Jan. 31 - Feb. 7, 2016
Multi-baseline Stereo	Feb. 8 - 14, 2016
Model Refinement	Feb. 15 - 20, 2016
Artifact Compensation	Feb. 21 - 27, 2016

Table 3: *Project Timetable*

tem utility inspection. *Power Electronics and Motion Control Conference and Exposition*.

CHAURASIA, G., DUCHENE, S., SORKINE-HORNUNG, O., AND DRETTAKIS, G. 2013. Depth synthesis and local warps for plausible image-based navigation. *TOG*.

CHEN, S. E., AND WILLIAMS, L. 1994. View interpolation for image synthesis.

CLIPP, B., ZACH, C., FRAHM, J.-M., AND POLLEFEYS, M. 2014. New minimal solution to the relative pose of a calibrated stereo camera with small field of view overlap. *CVPR*.

FAY, G. 2001. Derivation of the aerodynamic forces for the mesicopter simulation.

GALLUP, D., FRAHM, J.-M., MORDOHAI, P., YANG, Q., AND POLLEFEYS, M. 2014. Real-time plane-sweeping stereo with multiple sweeping directions. *IJCV*.

GIBIANSKY, A. 2012. Quadcopter dynamics, simulation, and control.

HENG, L., LEE, G. H., AND POLLEFEYS, M. 2014. Self-calibration and visual slam with a multi-camera system on a micro aerial vehicle.

KELLY, A., CHAN, N., HERMAN, H., HUBER, D., MEYERS, R., RANDER, P., WARNER, R., AND ZIGLAR, J. 2011. Real-time photorealistic virtualized reality interface for remote mobile robot control.

KOENIG, N., AND HOWARD, A. 2004. Design and use paradigms for gazebo, an open-source multi-robot simulator. *International Conference on Intelligent Robotics and Systems*.

KUO, C., KUO, C., LEBER, A., AND BOLLER, C. 2013. Vector thrust multi-rotor copter and its application for building inspection. *International Micro Air Vehicle Conference and Flight Competition*.

LAVALLE, S. M., YERSHOVA, A., KATSEV, M., AND ANTONOV, M. 2014. Head tracking for the oculus rift.

LAVEAU, S., AND FAUGERAS, O. 2004. 3-d scene representation as a collection of images. *CVPR*.

LI, B., HENG, L., KOSER, K., AND POLLEFEYS, M. 2013. Multiple-camera system calibration toolbox using a feature descriptor-based calibration pattern. *CVPR*.

LIPSKI, C., LINZ, C., BERGER, K., SELLENT, A., AND MAGNOR, M. 2011. Virtual video camera: Image-based viewpoint navigation through space and time. *SIGGRAPH*.

LIU, F., AND JIN, H. 2014. Content-preserving warps for 3d stabilization.

MARTIN, N., AND ROY, S. 2008. Fast view interpolation from stereo: Simpler can be better. *CVPR*.

MATUSIK, W., BUEHLER, C., RASKAR, R., GORTLER, S., AND MCMILLAN, L. 2000. Image-based visual hulls. *SIGGRAPH*.

MEIER, L., TANSKANEN, P., FRAUNDORFER, F., AND POLLEFEYS, M. 2011. Pixhawk: A system for autonomous flight using onboard computer vision. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*.

MEIER, L., TANSKANEN, P., FRAUNDORFER, F., AND POLLEFEYS, M. 2011. The pixhawk open-source computer vision framework for mavs. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*.

MEIER, L., 2012. Px4 autopilot. `https://pixhawk.org`.

MERRELL, P., AKBARZADEH, A., WANG, L., MORDOHAI, P., FAHM, J.-M., YANG, R., NISTER, D., AND POLLEFEYS, M. 2007. Real-time visibility-based fusion of depth maps. *CVPR*.

MEULLER, M., 2004. ecalc - xcoptercalc - multicopter calculator. `http://www.ecalc.ch/xcoptercalc.php?ecalc`.

MIKKELSEN, G., 2013. Tutorial: Low cost fpv setup for the rift (130ms latency). `https://forums.oculus.com/viewtopic.php?f=20&t=2358`.

NAEMURA, T., TAGO, J., AND HARASHIMA, H. 2002. Real-time video-based modeling and rendering of 3d scenes. *IEEE Computer Graphics and Applications*.

OSBORNE, M., 2012. Apm copter - dronecode. `https://copter.ardupilot.com`.

PLESS, R. 2003. Using many cameras as one.

POLLEFEYS, M., NISTR, D., FRAHM, J. M., AKBARZADEH, A., MORDOHAI, P., CLIPP, B., ENGELS, C., GALLUP, D., KIM, S. J., MERRELL, P., SALMI, C., SINHA, S., TALTON, B., WANG, L., YANG, Q., STEWNIUS, H., YANG, R., WELCH, G., AND TOWLES, H. 2007. Detailed real-time urban 3d reconstruction from video. *IJCV*.

QUIGLEY, M., GERKEY, B., CONLEY, K., FAUST, J., FOOTE, T., LEIBS, J., BERGER, E., WHEELER, R., AND NG, A. 2009. Ros: an open-source robot operating system.

RAJ, N., AND KATIYAR, R. 2014. Design and development of automated aero-terrestrial systems for persistent surveillance missions. *American International Journal of Research in Science, Technology, Engineering and Mathematics*.

RENGARAJAN, M., AND ANITHA, G. 2013. Algorithm development and testing of lowcost waypoint navigation system. *Engineering Science and Technology: An International Journal*.

RODRIQUEZ, A. F., READY, B. B., AND TAYLOR, C. N. 2006. Using telemetry data for video compression on unmanned air vehicles. *IAA Guidance, Navigation, and Control Conference and Exhibit*.

SAITO, H., KIMURA, M., TAGUCHI, S., AND INAMOTO, N. 2002. View interpolation of multiple cameras based on projective geometry. *SIGGRAPH*.

SAURER, O., KOSER, K., BOUGUET, J.-Y., AND POLLEFEYS, M. 2013. Rolling shutter stereo. *ICCV*.

SCARAMUZZA, D., ACHTELIK, M., DOITSIDIS, L., FRAUNDORFER, F., KOSMATOPOULOS, E., MARTINELLI, A., ACHTELIK, M., CHLI, M., CHATZICHRISTOFIS, S., KNEIP, L., GURDAN, D., HENG, L., LEE, G., LYNEN, S., MEIER, L., POLLEFEYS, M., RENZAGLIA, A., SIEGWART, R., STUMPF, J., TANSKANEN, P., TROIANI, C., AND WEISS, S. 2013. Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in gps-denied environments. *International Journal of Robotics Research*.

SIEBERT, S., AND TEIZER, J. 2014. Mobile 3d mapping for surveying earthwork projects using an unmanned aerial vehicle (uav) system. *Automation in Construction*.

STEINBRUCKER, F., STURM, J., AND CREMERS, D. 2014. Volumetric 3d mapping in real-time on a cpu. *ICRA*.

STEWART, B., KO, J., FORX, D., AND KONOLIGE, K. 2003. Revisiting problem in mobile robot map building: A hierarchical bayesian approach. *Conference on Uncertainty in Artificial Intelligence*.

STUPICH, N. 2014. Low power parallel rolling shutter artifact removal.

SZELISKI, R., AND KANG, S. B. 1995. Direct methods for visual scene reconstruction. *IEEE Workshop on Representations of Visual Scenes*.

WOETZEL, J., AND KOCH, R. 2004. Real-time multi-stereo depth estimation on gpu with approximate discontinuity handling.

WURMLIN, S., LAMBORAY, E., AND GROSS, M. 2003. 3d video fragments: dynamic point samples for real-time free-viewpoint video.

YANG, R., AND POLLEFEYS, M. 2003. Multi-resolution real-time stereo on commodity graphics hardware. *CVPR*.

YANG, R., WELCH, G., AND BISHOP, G. 2002. Real-time consensus-based scene reconstruction using commodity graphics hardware.

ZITNICK, C. L., KANG, S. B., UYTTENDAELE, M., WINDER, S., AND SZELISKI, R. 2014. High-quality video view interpolation using a layered representation. *SIGGRAPH*.