Author:
Boyle, Stephen J

Title:
Drone cinematography and the generation of environment models for flight planning

Drone Cinematography and the Generation of Environment Models for Flight Planning

STEPHEN JOHN BOYLE

Department of Electrical and Electronic Engineering

UNIVERSITY OF BRISTOL

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Doctor of Philosophy in the Faculty of Engineering.

April 2022

Word Count: 41000

Abstract

The use of drones (UAV's) as a camera platform has become widespread in media production, finding application in a diverse range of genres such as natural history, sport, news and movies. This thesis examines various aspects of their use for the coverage of live events such as sports.

The techniques and heuristics used in standard cinematography that are still applicable when filming from a drone are reviewed. Current drone cinematography practice is examined and its limitations for live filming are discussed. A novel shot hierarchy and taxonomy suitable for directing and controlling a multiple drone platform to film sports is proposed. A methodology to determine suitable drone camera shot parameters (e.g. drone height or speed), using the subjective testing of simulated camera shots (created using Unreal Engine), is discussed and the results obtained from testing a representative set of the proposed shot types are given.

For successful filming, preparation using a flight planning or training application is an important requirement and for this to be effective the software should incorporate a realistic model of the environment at the filming location. A method of creating these 3D models using photogrammetry, with image data obtained from existing resources such as Google Earth, is investigated. The theory behind the photogrammetry reconstruction process is discussed. In cases where images for photogrammetry are not available, they can be generated using a drone scan of the filming location. An outline of current state of the art research in the optimization of image capture scans for photogrammetry is given. A system built using the Python development environment in Blender, designed to optimize scanning parameters given a basic 3D model of an environment, is described. The system produces a metric value for the expected quality of photogrammetric reconstruction from a scan by calculating the accumulated coverage of surface points over all images. The approach is evaluated to determine the variation in the reconstruction metric with scan parameters and the results compared to those obtained using simulated scans created in Unreal Engine. The optimum height and camera angle calculated using the system were established to be largely independent of other parameters such as the focal length and the track separation distance. Using a methodology to adjust the parameters of the coverage model to take into account photogrammetry image overlap requirements, it has been found possible to calibrate the system so that the calculated optimum height and camera angles were comparable with the actual results obtained through photogrammetry (using scans simulated in Unreal Engine). The developed system provides significant benefits for the optimization of photogrammetry for 3D environment model creation.

Finally, conclusions are given on the benefits of simulation for camera shot design, and the use and optimization of photogrammetry for 3D environment model creation.

Author's Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

Acknowledgements

I would like to thank my academic supervisor, Professor David Bull for his help, support and advice throughout the PhD. I worked closely with David and Dr Fan Zhang as part of the team on the MultiDrone project and they have contributed to many aspects of the work described in this thesis. I would also like to thank Matt Newton, Tao Xu and Yingfei Yu whose project work has produced important insights into the optimization of drone scans for photogrammetry and has greatly informed my own work in this area.

Contents

List of Tables

List of Figures

Acronyms

2AFC Two Alternative Forced Choice

ACR Absolute Category Rating

AFOV Angular Field of View

CUDA Compute Unified Device Architecture

DEM Digital Elevation Model

DMOS Difference Mean Opinion Score

DSCQS Double Stimulus Continuous Quality Scale

DSIS Double Stimulus Impairment Scale

FSM Finite State Machine

GPS Global Positioning System

GSD Ground Sampling Distance

HDR High Dynamic Range

IMU Inertial Measurement Unit

LIDAR Light Detection and Ranging

LOD Level of Detail

MOS Mean Opinion Score

MPC Model Predictive Control

NIR Near-Infrared

PTX Parallel Thread Execution

SIFT Scale Invariant Feature Transform

SLAM Simultaneous Localization and Mapping

SRTM Shuttle Radar Topography Mission

SSCQS Single Stimulus Continuous Quality Scale

SSDQS Single Stimulus Discrete Quality Scale

SWIR Short-wave Infrared

UAV Unmanned Aerial Vehicle

USGS United States Geological Survey

1 Introduction

The use of Unmanned Aerial Vehicles (UAVs), more commonly known as drones, is becoming increasingly widespread in the film and television industries. They can be employed as a cost effective replacement for helicopters and cranes when filming aerial shots and their flexibility makes them ideal for filming live, dynamically changing events. The use of drones also supports the production of innovative types of shot which would not be possible using conventional techniques.

1.1 Context

The movie industry was an early pioneer in the use of drones for filming. The opening scene of *Skyfall* (2012), which took place on the rooftops of a bazaar in Istanbul, was partially shot using drones. This was the first time they had been used in a movie to film close-up action scenes rather than wider shots such as landscapes. In the pool party scene of *The Wolf of Wall Street* (2013) a drone was used to film a continuous shot beginning with a wide view of the beach and ending with close-ups of the guests. This would have been impossible to shoot using a helicopter because of the required close proximity of the actors to the camera in the final stages of the shot.

Since their adoption for movie production, drones are now commonly employed for filming in a diverse range of genres such as sports, natural history, archaeology, news and travelogues. The flexibility of drones to provide a wide coverage without the need for fixed infrastructure and the unobtrusiveness of small drones has made them particularly useful for the filming of sporting events and natural history. Notable examples include the 2014 Winter Olympics in Sochi [1] [2], the 2016 Olympics in Rio de Janeiro [3] and the BBC natural history documentary *One Planet, Seven Worlds* [4]. In news production drones have been used to provide coverage in dangerous or inaccessible areas, e.g. in the reporting of the aftermath of Hurricane Matthew in Haiti by NBC [5].

1.2 Filming using Autonomous and Multiple Drone Systems

The majority of filming from a drone is currently performed using one or a small number of drones under manual control. A number of systems, such as MultiDrone [6] and Red Dot [7], have recently been developed for semi-autonomous filming using multiple drones. The use of autonomous and multiple drone filming platforms can provide many benefits when filming live action, especially for events taking place over a wide area.

The MultiDrone consortium [6] developed an innovative multiple drone platform for use in filming outdoor sporting events such as cycling races and rowing regattas. Autonomous features including intelligent shooting and target tracking allows the system to be operated by a small team, making it cost effective for many filming requirements. The system incorporates many safety features such as crowd detection/avoidance and has built-in automated emergency landing procedures. The use of MultiDrone for filming sports events can give many advantages over traditional techniques:

- Individual drones can be assigned to track specific targets so that if unexpected events occur the video stream can be switched to the appropriate drone (e.g. when a rider crashes in a cycling race).
- No complex fixed filming infrastructure is needed (e.g. cranes, booms etc).
- Aerial shots are much more cost effective compared to filming from helicopters.
- High level control of the drone swarm. Directors can concentrate on specifying targets to film and shot types to use rather than spending time communicating instructions to multiple camera operators and pilots.
- Reduced crew size and operating costs.
- Drones needing recharging can be automatically replaced with standby drones without any loss of continuity.
- New and innovative shot types can be employed to improve the viewing experience (e.g. 360 degree panoramas and fly-through shots). Drone camera shots can give an increased sense of movement and dynamism compared with those created using conventional camera techniques, allowing the viewer to feel more immersed in the action.

The MultiDrone consortium consisted of a number of partner institutions and companies across Europe including:

- The Aristotle University of Thessaloniki :- responsible for the development of target detection and tracking, crowd detection and intelligent shooting capabilities.
- The University of Seville :- design of the high level decisional architecture.
- Thales :- responsible for LTE communication infrastructure and integration.

- Deutsche Welle and RAI television companies.
- The University of Bristol :- Specifications for MultiDrone camera shots and the drone operating envelope. Development of a flight planning and training package.

1.3 Research Motivations

The research detailed in this thesis concerns various problems relating to the filming of live action such as sports with drones and the modelling of environments for drone flight simulations.

Some of the problems regarding shot types and shooting that were identified included:

- There is currently no universal standard for the naming and specification of camera shot types and those in use are often complex, open to ambiguity and require a technical grammar.
- Many of the shot types of traditional cinematography do not lend themselves to filming events such as sports, being most useful for prescribed motions of camera and subject.
- For live filming the task of directing multiple drones to shoot multiple targets, or a single target from multiple angles, will be difficult. This is especially true for the fast moving and unpredictable action involved with sports.

It is evident that directing fast moving action, with multiple drones, using current standard shot types would present many difficulties. Defining shots in terms of the relative motion of the camera with respect to the target and using a simple terminology for shot nomenclature will help to reduce shot decision and selection time, easing the task of directing multiple cameras to shoot multiple targets.

The control of a multi-drone filming system will also be simplified by incorporating autonomous features such as target tracking, collision avoidance and automatic shooting. Implementation of an automatic shooting capability will require each shot type to be fully parameterized, enabling the drone control system to calculate the desired drone camera position and angle throughout the duration of the shot. Suitable default values and operational ranges for these parameters will need to determined, taking into account possible effects on the quality of the viewing experience. The perceived quality is highly subjective in nature, varying according to each viewer's personal preferences and for some viewers certain parameter values may give an unpleasant experience.

Flight planning and training applications are an important requirement for any drone filming platform. Shots and shot sequences can be evaluated for feasibility and viewing experience, and the response to various scenarios practiced. Detailed flight planning and training for a shoot will be supported by an ability to integrate accurate real world models into a simulation. There are no readily available data sources for 3D environment models that have complete world coverage and in many cases a model of the shooting location will need to be created using techniques such as photogrammetry or laser scanning. Models produced using such methods will often need editing to remove unwanted artefacts or repair distortions.

1.4 Aims and Contribution

The aims of the research were as follows:

- Specify a library of shot types for integration into an automatic shooting system such as MultiDrone that is suitable for filming live events such as sports.
- Develop a methodology to subjectively test the proposed shot types and determine appropriate default parameters and operating envelopes.
- Assess the suitability of existing sources of environment data such as Google Earth for the production of world models in a drone flight planning and training program.
- Develop a system to help optimize a scan used to capture images for environment model production with photogrammetry.

The contributions produced as a result of the research include:

- A library of shot types, fulfilling many of the shooting requirements for filming sports events such as races, has been defined using a hierarchical taxonomy in terms of the relative motion between camera and target. The shots can be specified with a simple grammar comprising of commonly understood terms. The use of the proposed library can simplify an automatic shooting control interface and help to ease the task of the director by reducing the time to select a shot and communicate instructions to members of the drone control team. The proposed shot library will be especially beneficial for the filming of live events such as sports with multiple drone cameras.
- Parametric definitions for the drone trajectories and camera gimbal rotation angles of each shot have been produced, to facilitate their integration in the control system of an autonomous filming platform.

- A method to assess the quality of the viewing experience for shots and determine optimum shot parameters, by subjectively testing videos of shots simulated using Unreal Engine, has been developed. A selection of the proposed shot types for filming sports races have been studied using this method and it has been successful at identifying a statistically significant range for parameters (e.g. drone height) that gives a preferential viewing experience.

  The technique can be used to configure the default values and operating ranges of camera shot parameters in an automatic shooting system. Only a limited number of single drone shots (for the scenario of a race) have been tested and, for the full development of such a system, shot libraries for all required types of scenario will need to be defined and tested. The method could be extended to a case involving multiple drones, for example, optimizing the transition between shots from two drones.

- A methodology, using photogrammetry, to create 3D environment models using images captured from data sources such as Google Earth has been evaluated and techniques to optimize the procedure established.

- Procedures using the Blender 3D modelling program to repair common problems found in models produced using photogrammetry (e.g. distortions and artefacts) have been detailed.

- Previous research on the optimization of image capture for photogrammetry has been surveyed, including work to determine the effect of increasing the number of input images and the effect of the overlap between scan lines.

- A Blender Python program has been developed to estimate the model quality resulting from a photogrammetric reconstruction using images captured during a scan. The system can be used to optimize image capture scans employing a fixed drone height and camera angle, given a simple model of the scan area. This novel approach to optimizing scans can potentially reduce scan times and simplify drone control when compared with techniques that attempt to optimize a 3D drone trajectory with continually varying camera angles.

Papers published as a result of this research or containing elements of this research are:

- S. Boyle, F. Zhang, and D. R. Bull, "A Subjective Study of the Viewing Experience for Drone Videos," in *2019 IEEE International Conference on Image Processing (ICIP)*, (Taipei, Taiwan, Sep. 22–25, 2019), IEEE, 2019, pp. 1034–1038. DOI: 10.1109/ICIP.2019.8803747

- S. Boyle, M. Newton, F. Zhang, *et al.*, "Environment Capture and Simulation for UAV Cinematograhy Planning and Training," in *European Signal Processing Conference, Satellite Workshop: Signal Processing, Computer Vision and Deep Learning for Autonomous Systems*, 2019. [Online]. Available: https://research-information.bris.ac.uk/ws/portalfiles/portal/199953761/Environment_Capture_and_Simulation_for_UAV_Cinematography_Planning_and_Training.pdf

- F. Zhang, D. Hall, T. Xu, *et al.*, "A Simulation Environment for Drone Cinematography," *IBC Technical Papers*, 2020. [Online]. Available: https://www.ibc.org/technical-papers/a-simulation-environment-for-drone-cinematography/6747.article

1.5 Structure of Thesis

The remainder of the thesis is structured as follows:

Chapter 2 provides an overview of the standard cinematographic techniques and heuristics that are also relevant when filming using drones. Previous research in drone cinematography is then examined, including work on systems to design and preview drone drone camera shots, the optimization of drone trajectories and the autonomous filming of target subjects following cinematographic principles. Details of other related work in fields such as automatic camera control and problems related to multiple camera images (e.g. camera calibration) are then given.

Chapter 3 gives an outline of current drone filming techniques and examines the application and limitations of this practice when filming live action using single and multiple drone platforms. A new shot hierarchy and taxonomy suitable for filming fast moving live action such as sports races is presented. The parametric equations describing the drone trajectory and camera gimbal rotation for each of the proposed shot types are given.

Chapter 4 describes a methodology employing the subjective testing of simulated drone camera shots to determine optimum shot parameters and operating envelopes. Results for studies with various shots in cycling and car racing scenarios are given and compared with results obtained using real footage. The procedure for converting shot parameter values for use with different cameras is described.

Chapter 5 examines methods to create real-world models suitable for incorporation into drone planning and training systems. A review of currently available software for flight planning and training is given. The prevalent techniques for capturing data (such as height-maps), which can be used to create 3D environment models, are described. Details of some useful, publicly available resources for existing data are given. The use of the *Land-*

scape object to model terrain in Unreal Engine is discussed and a method to import height-map data into a *Landscape* model is detailed. The application of photogrammetry to generate environment models is then examined. An introduction to the theory behind photogrammetry is given. A workflow to produce environment models from Google Earth images using 3DF Zephyr photogrammetry software is described. A drone simulation package for training and flight planning, which has been developed in Unreal Engine and uses data from Google Earth to produce a realistic world environment, is described.

Chapter 6 examines techniques to optimize photogrammetry reconstructions. A review of current research regarding the optimization of trajectories and camera angles for drones being used to capture images for photogrammetry is given. Studies looking at the effect of the number images used for a photogrammetry reconstruction and the drone track overlap distance (when scanning an area for image capture) on the quality of the final reconstructed model are described. The development of a system (built in Blender using the Python language) to estimate the quality of photogrammetric reconstruction, given parameters defining a scan used for image capture and a simple model of the environment, is detailed. The results from a number of simulations, to determine the qualitative variation in the reconstruction quality predicted by the program with changes to scan parameters, are discussed. The prediction of the Blender system, for the quantitative variation in reconstruction quality with scan parameter values, is evaluated against the results from actual reconstructions created using simulated scans in Unreal Engine. The configuration of the program to match the results of real photogrammetry software is discussed. Possible improvements to the program, regarding the method used to calculate the metric estimating scan coverage, are given.

Chapter 7 details the results and conclusions obtained from the research and areas in which further investigation could be of benefit.

2 Drone Cinematography

2.1 Standard Cinematography Techniques

The terminology, techniques and heuristics used in standard cinematography are also largely applicable when filming from a drone.

2.1.1 Standard shot types

Film-makers commonly use shots in which the framing conforms to conventions known to give a pleasing appearance. Examples of such shot types include:

- Wide Shot: The entire subject is visible and takes up nearly the whole frame. A small safety gap at the top and bottom is used to prevent the head and feet being 'cut off'.
- Very Wide Shot: The subject appears just large enough to be visible. It is often used as an establishing shot to help give the context of the scene.
- Extreme Wide Shot (Extreme Long Shot): Often used as an establishing shot to depict the wider area in which the action of the scene takes place with the main actors too small to be seen.
- Medium Shot (Mid Shot): Most of the frame is filled by one or more actors with each having their head and much of their body above the knee visible. This is the most common type of shot used in film and television since it allows the facial expressions and gestures of the actors to be easily seen [11]. The shot can be combined with a camera pan to follow any action.
- Close Up: A particular feature of the subject (e.g. a persons face) fills the entire frame.
- Medium Close Up: In-between the Close Up and Medium shots, typically showing the face and upper body of the actor.
- Extreme Close Up: A very high zoom is used so that a small area of the subject or a particular detail fills the frame. Filming part of the face such as the eyes can be used for dramatic effect (e.g. to convey emotion).

For examples of the above and other shot-types see the *Film Riot* YouTube channel [12].

2.1.2 Camera Angles

Film-makers often select camera angles to achieve a particular cinematographic effect. Mascelli [11] gives guidelines on how the purpose of a shot and its desired effect on the viewer can be used to select a suitable camera angle and framing for the subject:

- Eye Level: The most commonly used camera angle which gives a natural view of the subject.
- Low Angle: The camera points upwards towards the subject from a low position. This angle can be used to make the subject appear more powerful or prominent.
- High Angle: The camera looks down on the subject from above. In a dramatic scene this angle can be used to make a person appear less powerful or more submissive. Mascelli [11] states that a high angle will make a moving subject appear slower (especially with large camera distances) and so it should be used with care if filming fast-moving action such as a race.
- Birds Eye: The camera is placed at a long distance directly above the subjects. It is often used to give an overall perspective of a scene showing the relative positions of the subjects or to to give a dramatic effect emphasizing a sense of height.

For examples of the above camera angles see the *Film Riot* YouTube channel [13].

2.1.3 Rules and Heuristics of Cinematography

There are a number of rules and heuristics that film-makers often follow to produce visually pleasing shots and shot sequences that convey a storyline without confusing the viewer. Arijon [14] provides a comprehensive overview of such standard cinematographic techniques, giving suitable shot sequences for various scenarios such as a dialogue between two people standing next to each other, a conversation on the telephone and dialogues involving differing numbers of people (e.g. groups of three or more than four people).

For a scene involving two principal actors in conversation the 'line of interest' is given by the gaze direction of the participants [14]. The camera positions for each shot are often placed at the corners of a triangle located on one side of the line of interest, which has its base parallel to the line. By following this *Triangle Principle* the film-maker keeps camera positions on the same side of the line of interest which will result in each actor always appearing on the same side of a frame. This ensures that the viewer will not be confused at shot transitions by an actor appearing to change sides

with the other actor and that any motion of an actor will not appear to reverse in direction. The exact locations and directions of the cameras on the triangle will determine the type of shot. For an 'external reverse angle' shot (see Figure 1) the base-line cameras are located to the outside of the players and pointed inwards towards them to give a shot of an actor from over the shoulder of the other actor. For an 'internal reverse angle' shot (see Figure 2) the base-line cameras are positioned between the actors and point outwards towards a particular actor. A camera positioned at the apex of the triangle is used for a shot showing the faces of both actors.

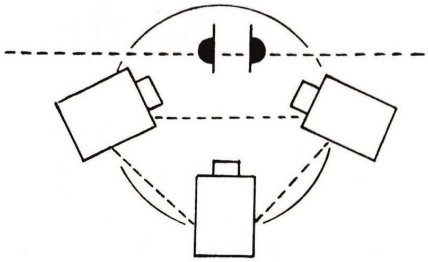


Figure 1: External Reverse Angle Shot [14].

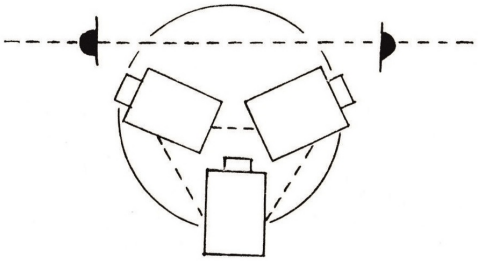


Figure 2: Internal Reverse Angle Shot [14].

The *Triangle Principle* can be extended to a more general scenario (e.g. in which the principal subjects are two football players and a football) by defining the line of interest to be the line or composite line joining the principals or the trajectory of a single moving subject of interest.

The knowledge embodied in the *Triangle Principle* is also expressed in the *180 Degree Rule* which states that once an establishing shot for a scene has defined which side of the line of interest the camera is placed, the camera should not move to the other side of this line in subsequent shots (see Figure 3 and *Film Riot* YouTube channel [15]).

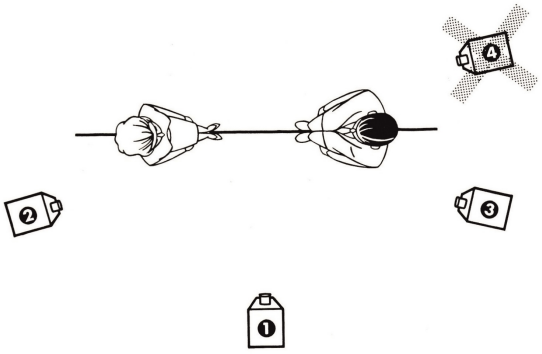


Figure 3: The 180° Rule [11].

Composing a shot using the *Rule of Thirds* has been found to give visually pleasing results. The key elements of the shot are placed at positions on lines which divide the frame into thirds, both horizontally and vertically (as shown in Figure 4). For a person looking towards a point off-screen or a subject in motion the placement should be such that the person looks towards or the subject moves towards the larger two-thirds portion of the screen. Placing a subject on the top horizontal third line is also often used to avoid 'dead space' above the subject. For examples of the use of the *Rule of Thirds* see the *Learn Online Video* YouTube channel [16].

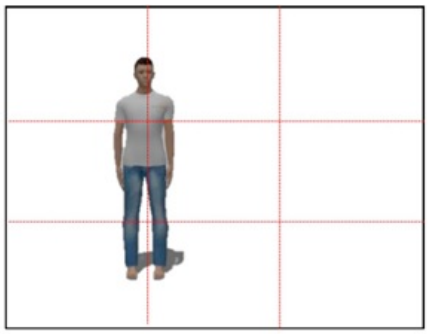


Figure 4: The Rule of Thirds [17].

Gebhardt *et al.* [18] states that the most pleasing shots are obtained by using either a static camera, panning with a tripod mounted camera or using a camera on a dolly. Transitions between shots should be jerk-free or made using a cut. He *et al.* [19] also states a number of heuristics commonly used by film-makers to give pleasing shots and a good scene development:

- An actor should initiate all movement with the camera following and coming to rest before the actor.

- Establishing shots should be used for new developments and before any close-up shots.
- A scene with motion should be broken in to two shots with the actor appearing to move from one side of the screen to the other.
- Jump cuts from one scene to a similarly composed scene (regarding number and placement of actors) should be avoided to prevent a jarring transition.

2.2 Current Filming Practice for Drone Cinematography

Although many of the techniques and heuristics of standard cinematography are also used when filming with a drone, there are cases where the conventional rules do not apply or are less restrictive. One such example is the *Triangle Principle*, which is applicable when using fixed camera positions to prevent a shot that is jarring or confusing due to an abrupt change of view from one side of the line of action to the other, but is not applicable when using a continuous (i.e. smooth) drone camera transition.

Currently, most drone shoots are filmed using one, or a small number of drones, under manual control. Typical shots include 'Tracking' shots of moving subjects (e.g. of cars) or 'Establishing' shots used to show the context for a scene [20]. Previously, using traditional techniques, these would have been filmed from a moving vehicle (e.g. a car or motorbike) for close range, low angle shots or from a helicopter for long range and high angle shots. Aerial shooting using a drone rather than a helicopter can significantly reduce filming costs, from around $25,000 a day to $5000 a day [21]. Drones can be used to produce shot sequences which would previously have been impossible, such as uncut 'Flythrough' shots going through a window into a building or along a narrow passage-way. Lightweight, low noise drones are also commonly used in sports and nature filming since they can follow target subjects and provide close-up shots without causing disturbance.

2.3 Previous Work on Drone Cinematography

Research in drone cinematography has largely focused on problems involving semi-autonomous and autonomous drone control, with early work often concerned with filming a number of predetermined stationary targets.

Joubert *et al.* [22] developed a system to allow drone users to visually design and preview camera shots whilst ensuring adherence to constraints such as the maximum thrust and velocity of the drone. The camera was assumed to be positioned at the centre of mass of the drone and trajectories were designed by specifying the look-from and look-at positions at key-frames, with the system interpolating a smooth C^4 continuous trajectory between

these points. A model simulating the actual dynamics of the drone calculated velocities, thrusts and gimbal angles along the path and any unfeasible sections were highlighted for remediation (e.g. by modification of the easing curves defining the timing of the drone along the trajectory). The control input for the drone $\mathbf{u}$ was calculated using an equation governing drone dynamics, relating the state of the drone $\mathbf{q}$ to $\mathbf{u}$, given by:

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{B}(\mathbf{q})\mathbf{u} \tag{1}$$

In the above equation $\mathbf{H}$ models inertia, $\mathbf{C}$ models velocity dependent forces (e.g. drag), $\mathbf{G}$ models the effect of gravity and $\mathbf{B}$ models the effect of the control inputs. To find $\mathbf{q}$ and its derivatives is was necessary to calculate the position, velocity, acceleration and orientation of the drone from the camera trajectory. The acceleration of the drone (calculated using a finite difference method) was used to determine the required thrust $\mathbf{f_t}$. The drone y-axis direction $\mathbf{y_q}$ was given by the thrust direction $\frac{\mathbf{f_t}}{\|\mathbf{f_t}\|}$ and assuming the drone x-axis direction was equal to the camera look-at direction $\mathbf{x_c}$ the drone z-axis direction was calculated from $\mathbf{z_q} = \mathbf{y_q} \times \mathbf{x_c}$, which fully determined the orientation. The required drone input $\mathbf{u}$ was then calculated by substituting values for $\mathbf{q}$ and its derivatives into Equation 1 and using a pseudo-inverse for $\mathbf{B}$. It was thus possible to determine if the trajectory was feasible from the required drone inputs given by $\mathbf{u}$. Virtual previews of the camera view along the trajectory were generated from a 3D model of the environment to give the user an insight in to the expected cinematographic quality of the filmed sequence. The system could also generate control software to execute the flight autonomously.

Later research often focused on the optimization of drone camera trajectories. Gebhardt *et al.* [18] developed a tool for the generation of feasible drone trajectories from sketched key-frames, each defining a time and the corresponding desired drone position. The key-frame positions were used as soft constraints in the trajectory optimization, allowing a trade off between drone position and feasibility. High level cinematographic constraints (e.g. to minimize perspective distortion or to ensure smooth camera motion) and collision avoidance constraints could also be specified. Although the optimization algorithm required knowledge of the location of the targets to be filmed, it was implemented using an iterative quadratic programming method allowing it to be used in real-time. Hence the control program could continually update its calculation for the optimum trajectory using the current position of moving targets obtained via tracking. For validation of the trajectory the tool could generate a virtual preview showing the drone's camera image over its flight. To calculate the optimized trajectory a simplified linear model relating the drone inputs $\mathbf{u}$ (consisting of a thrust $\mathbf{F}$ and torque about the drone body z-axis M_ψ) to its dynamics was used:

$$\mathbf{u} = \begin{bmatrix} \mathbf{F}, & M_\psi \end{bmatrix}^T$$

$$m\ddot{\mathbf{r}} = \mathbf{F} + m\mathbf{g} \tag{2}$$

$$I_\psi \ddot{\psi} = M_\psi \tag{3}$$

In the above equations $\mathbf{r}$ is the position vector of the centre of mass for the drone, ψ is the yaw angle and I_ψ is the moment of inertia about the body z-axis. The state of the drone is calculated at discrete time intervals of Δt using:

$$\mathbf{x}_i = \left[\mathbf{r}, \psi, \dot{\mathbf{r}}, \dot{\psi}\right]^T$$

$$\mathbf{x}_{i+1} = A_d \mathbf{x}_i + B_d \mathbf{u}_i + c_d \tag{4}$$

In the above equation A_d describes the response of the system due to its inertia, B_d describes the response due to the drone inputs and c_d describes the effect of gravity. The variables $\mathbf{x}_i$ and $\mathbf{u}_i$ are optimized over N time steps (i.e. $i = 1 \ldots N$) subject to constraints on $\mathbf{u}_i$ due to maximum values for force and torque inputs and minimization of the following cost functions:

1. E^k : measures the deviation of the drone position from the desired key-frame position.

2. E^d : measures the jaggedness of the trajectory calculated from the higher derivatives (using a finite difference approximation) of the drone position with respect to time.

3. E^c : measures the difference between the camera direction and the direction of specified targets at particular stages on the trajectory.

4. E^s : measures the skewness of the target images.

The systems developed by Joubert *et al.* [22] and Gebhardt *et al.* [18] require users to predefine drone trajectories. Recently the autonomous control of drone cameras without the need for such advance planning has become a common topic of research.

Nägeli *et al.* [17] implemented a Model Predictive Control (MPC) algorithm for autonomously controlling a drone in which control inputs were optimized in small time steps using the state of the drone at the beginning of the time step and a model of drone dynamics. Constraints were added to the optimization procedure to ensure collision avoidance, valid drone inputs (e.g. thrust in the body z-axis direction), a feasible drone state (e.g. vertical and horizontal velocities less than the maximum allowable) and the adherence to high level aesthetic requirements. The optimization algorithm minimized a cost involving terms measuring the deviation of the projection on the image plane of each target being filmed from its desired image (regarding position, size and view angle). For example, an image position cost c_{image} is given by:

$$\mathbf{r}_d^c = \left[\mathbf{m}_d, \quad 1\right]^T$$

$$c_{image} = \|\rho_m\| Q_m \quad \text{with} \quad \rho_m = \frac{\mathbf{r}^c_{ch}}{\|\mathbf{r}^c_{ch}\|} - \frac{\mathbf{r}^c_d}{\|\mathbf{r}^c_d\|} \tag{5}$$

In the above equation $\mathbf{r}^c_d$ is the direction vector pointing from the camera centre to the desired pixel location $\mathbf{m}_d$ in the image and $\mathbf{r}^c_{ch}$ is the vector from the camera to the target as measured in the camera frame. Camera intrinsics (focal point and focal length) were used to calculate $\mathbf{m}_d$ given the desired target location on the image plane. To implement collision avoidance soft constraints were added for each known object in the environment (having size Ω_s and at a distance from the camera of $\mathbf{r}_{ct}$), using a slack variable s_c to ensure the problem was solvable:

$$\mathbf{r}^T_{ct} \Omega_s \mathbf{r}_{ct} > 1 - s_c \tag{6}$$

The MPC problem was solved for N time steps of duration Δ_t with the positions of targets over this period ($N\Delta_t$) being estimated from the initial positions using a Kalman filter (based on a constant velocity Gaussian model). The calculations were repeated after each time step which resulted in a robust closed loop performance and enabled the successful filming of targets moving unpredictably.

Joubert *et al.* [23] developed a system able to autonomously film one or two human subjects using a predefined sequence of canonical shot types, following well established visual composition principles. The particular shot type, number of targets and target position(s) were used to calculate the camera look-from and look-at positions for each shot, ensuring that the resultant image adhered to principles such as the *Rule of Thirds* and the *Triangle Rule*. For example, if there were two subjects the average eye position would lie at the centre of the top horizontal third line and the drone was always positioned at the side of the line of action that initially showed more of the subject(s) faces (as determined by the gaze direction). The trajectory of the camera between shot positions was found by blending two easily generated basis trajectories as shown in Figure 5. For two targets A and B having position vectors $\mathbf{P}_A$ and $\mathbf{P}_B$ with the camera positions at the first and second shot being $\mathbf{C}_0$ and $\mathbf{C}_1$, the blend trajectories σ_A and σ_B are given by:

$$\sigma_i(u) = \mathbf{P}_i + d_i(u) \cdot \mathbf{v}_i(u) \quad \text{for} \quad u \in \Big[0, \quad 1\Big], \quad i = \Big\{A, \quad B\Big\} \tag{7}$$

In the above equation $d_i(u)$ linearly interpolates between the initial subject to camera distance (from subject i to $\mathbf{C}_0$) and the final distance (from subject i to $\mathbf{C}_1$). The function $\mathbf{v}_i(u)$ gives the spherical linear interpolation between the direction from subject i to $\mathbf{C}_0$ and the direction from subject i to $\mathbf{C}_1$. The two trajectories were blended together to form trajectory $\sigma(u)$ using a function $w(u)$ that was optimized to minimize the distance of the blend from the basis trajectories, enforce C^4 continuity and adhere to

minimum camera to target separation constraints:

$$\sigma(u) = w(u) \cdot \sigma_A(u) + (1 - w(u)) \cdot \sigma_B(u) \tag{8}$$

Finally an easing curve defining how time varies with u was used to generate a time dependent trajectory. Targets were tracked using RTK GPS and inertial measurement unit (IMU) sensors and the system was able to produce real-time updates to the calculated trajectory in response to target movements.

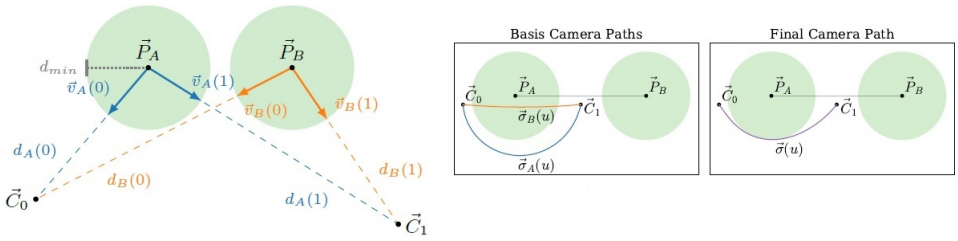


Figure 5: Blending Two Basis Trajectories [23].

Research on filming with drones has focused on trajectory planning and the use of semi-autonomous or autonomous control to film targets using standard shot types and composition. There has been little research on the optimization, with regards to viewer experience, of drone camera shots. The design of automatic shooting platforms will require existing shot types (or novel shot types) to be assessed with regard to their applicability to particular genres of filming. Further research is also needed to fully exploit the potential of filming using a multiple drone platform, including work on the development of novel shot types, shot transitions and heuristics.

2.4 Other Related Work

He *et al.* [19] created 'The Virtual Cinematographer' which provided real-time camera control for the filming of simulated 3D worlds. Each different filming scenario was encapsulated in a film 'idiom' (e.g. a '2Talk' idiom for two actors talking to each other) which was implemented in software as a hierarchical finite state machine (FSM). Camera modules which used cinematographic principles to determine subject framing were implemented for common types of shot such as apex, internal, external, pan and tracking. Each state of an idiom resulted in a particular camera module call such as external(A,B) for an external shot of actor A over the shoulder of actor B. Idioms changed state as a result of certain events such as a change in the actor talking or a shot exceeding a time limit, resulting in a new camera call. These changes of state were structured in such a way that the camera shot sequence adhered to good cinematographic practice.

Christie *et al.* [24] give a comprehensive account of techniques to solve various camera control problems in computer graphics. Topics covered include:

- Interactive control of virtual cameras.
- Assisted control methods to aid user navigation through 3D worlds such as obstacle avoidance (e.g. by maintaining the camera at a fixed distance to an object or by using repulsive potentials), road-map planners to optimize a route (e.g. by decomposing a scene into cells and building a network linking the cells) and the creation of natural camera transitions between a number of targets.
- Automated camera control methods. These techniques have many applications in computer graphics such as the solution to the problem of ensuring targets appear at given coordinates in a camera image and in camera planning for multi-media guided tours. Optimization based, constraint based and constrained optimization based approaches to solve such problems are described. Reactive methods which directly compute a response as a result of a change in an image (e.g. visual servoing to control a set of visual image features) are also discussed.
- Assurance of occlusion free views using approaches such as ray-casting, bounding volumes (around camera and target) and shadow volumes.

Hartley and Zisserman [25] give an extensive account of many techniques that have been applied to problems involving multiple camera images such as:

- Three-View computational methods using a trifocal tensor to calculate the intrinsic and extrinsic camera matrices (up to a common projective transformation) and the 2D projection of a world point on a plane given its projection on two other planes.
- Automatic camera calibration using the images obtained from a rotating camera.
- N-View computational methods (such as bundle adjustment) which when combined with automatic camera calibration enable the 3D scene to be reconstructed by matching corresponding points in the N 2D images.

2.5 Summary

There are many rules and heuristics governing composition, framing and transitions when filming using fixed cameras (with or without panning). Research on filming with moving camera platforms such as drones has focused on trajectory planning and the semi-autonomous or autonomous control of a drone to film targets using standard shot types and composition. There has been little research on the optimization, with regards to viewer experience, of drone camera shots and the development of new types of shot and techniques that can fully exploit the potential of drone cinematography.

The techniques and approaches employed when filming with a drone mounted camera will be dependent on the particular requirements of the genre involved. The following section examines the selection and definition of a library of shots that has been designed to ease the task of filming and directing live action such as sports using a single drone or a multiple drone system.

3 Shot Specifications for Filming Sports using a Drone Platform

A number of drone professionals from media production companies (e.g. Deutsche Welle and Radiotelevisione Italiana), including directors, line producers, camera operators and drone operators, were surveyed to gain insight into current drone practice and its limitations, and to elicit shooting requirements for the MultiDrone system [6]. The results from the survey revealed that most of the shot types currently employed in aerial shooting (e.g. from a drone or helicopter) were adapted from standard cinematography practice and few took advantage of the extra capabilities available when filming from a drone. The majority of drone filming was carried out using a single drone and the possibilities and advantages of filming using multiple drones were not widely exploited.

3.1 Limitations of Current Practice

When filming live events with a single drone it can be difficult for the director to anticipate what will happen next (especially for fast moving action such as sports) and hence be able to position the drone to give the best view of the action at all times. With a single drone it is difficult to provide uninterrupted coverage of prolonged events due to limitations in drone battery life, although this can be mitigated with the use of standby substitute drones. If broadcasting is deferred any breaks in coverage can be removed using post-production editing. Some of the limitations associated with filming using a single drone can be overcome in the case of offline (non-live) shooting with full post-production editing. For instance, if an aerial sequence requires shots from a variety of camera angles then different versions can be filmed in separate takes and the results edited together in the post-production phase. Disadvantages of this method include the difficulty in maintaining scene consistency across takes (especially when filming outdoors due to varying lighting conditions) and the extra time and cost involved in shooting multiple takes. Because of these shooting time and cost constraints a compromise in the number of shots may be needed.

The coordination of multiple manually controlled drones for a live shoot is difficult. It can also be costly due to the large crew size required, with each drone potentially requiring both a pilot and camera operator. Manual control is less of a problem for the case of a movie shoot since it is possible for the drone pilots to rehearse shots in advance.

3.2 Advantages and Potential of Drone Cinematography

The use of drone mounted cameras gives the film-maker great flexibility in shot design, with the movement of the drone relative to the target and the

camera tracking only limited by physical constraints on drone and camera motion (e.g. the maximum drone speed). This increases the potential for the creation of innovative shot types that can contribute to new and exciting viewing experiences. However this flexibility also increases the potential for inadvertently creating visually unappealing shots which confuse or disorient the viewer. Hence any new shot types will need careful testing to determine constraints on their use, and suitable ranges and optimum values for shot parameters (e.g. parameters defining the drone trajectory relative to the target and camera orientation or tracking).

The use of a multiple drone filming system provides a number of benefits when compared with a single drone system. For filming live events such as sports, cameras can be positioned so that the director can cut to alternative views of the main action and (especially for events such as cycling road races taking place over wide areas) respond quickly to evolving situations away from the main focus. For non-live filming a multiple drone system will allow sequences to be filmed from a variety of angles without having to use multiple takes. Multiple drone cinematography offers a great potential for innovative techniques which will enhance the viewing experience, improve coverage and aid narrative, but the technology is still in its infancy and there are few guidelines on how best to exploit it. Some of the possible shot types and effects which can be employed in multi-drone filming systems include:

- Single drone movement transitions: For example upright to inverted or a pass and then a lift to zoom out.
- Overtaking: Viewpoint moving from behind to in-front of subject (using a single drone or by transitions between multiple drone cameras).
- Seamless view transitions: Changing from the view of one drone to that of another. One possible strategy could be to use a method similar to that used by Joubert *et al.* [23] to ensure smooth transitions between camera shots, so that both of the trajectories of the two drones involved lie on a single smooth curve. Images from the two drones could also be combined to produce a synthesized image approximating the view mid-way between the drones.
- Panoramic shots: Transitions from one drone camera to the next in a sequence giving an unbroken view of the surroundings.
- Multi-view shots: Virtual views generated from multiple drone cameras (e.g. the views from multiple cameras could be stitched together to give a Bird's Eye View or 360° content).
- Dolly zoom (contra zoom): The zoom is adjusted as the camera moves towards or away from the subject so that the subject appears the same size but the background size changes, leading to perspective distortion.

- Hyperzoom: Continuous zoom using a combination of camera zoom and view transitions to other drone cameras at varying distances to the subject.
- Augmented reality: Methods to determine camera motion such as those described by Hartley and Zisserman [25] will allow the use of augmented reality video in which a computer generated 3D model is superimposed on the video from a drone in such as way that the motion of the model in relation to the background is consistent with that of a real object in the scene.
- Synthetic 3D views: Generated using scene reconstruction techniques (as described by Hartley and Zisserman [25]).
- New image formats and camera technologies: HDR (High Dynamic Range) and plenoptic (light field) cameras which allow images to be post-processed to give the required depth of field and focus.

There are a number of barriers which have limited the use of multiple drone filming systems, in particular the difficulty in coordinating the control of multiple drones for a filming task and the cost of operation. The implementation of semi-autonomous operation (e.g. target tracking and automatic shooting) can reduce the complexity of operation and crew size requirements for a multiple drone system and hence make such a system a viable proposition. However many of the shot types used with standard cinematography may not be appropriate when filming from a drone and hence automatic shooting systems will require suitable shot libraries (consisting of existing or novel shot types) to be defined for particular genres of filming.

3.3 Limitations of Standard Shot Types for Filming with Drones

There is no entirely consistent, standard language or grammar in use to specify drone operation and camera shot types. Terms used in shot descriptions are not always precisely defined and these ambiguities can lead to differences in their interpretation. There are a large number of shot type definitions with some being very similar (e.g. shots differing only in pan or tilt camera tracking). Complex shot specifications are needed to precisely define the required camera motion and tracking. Because of the similarity between certain shots there is a degree of redundancy and in many shooting situations a single shot type with a camera motion or tracking relative to a target could replace multiple shot types with prescribed motion. When filming sports many of the required camera shots have a camera motion that is defined relative to the motion of a target subject, for example following or overtaking the subject, and using shot specifications with prescribed

motions does not fit well with this type of shoot. A complex naming terminology is necessary to identify each shot from all the categories available along with its particular camera motion and tracking. The complexity of the shot hierarchy makes the task of directing fast moving action such as sports difficult because of the time needed to decide which of the many shots to use and communicate this to the production team. Using these traditional shot types in an automated shooting system such as MultiDrone would also require a complex shot selection interface.

It is evident that with the increasing sophistication of drone operations and the use of multiple drone systems, a simpler and unambiguous grammar will be needed (especially for the filming of fast moving live events such as sports). The creation of new customized shots types for particular well defined situations (e.g. the filming of races) could reduce the number of different types needed during filming. This, along with a more natural language to specify each shot (encapsulating the motion of the camera relative to the target), would ease the directors task and support the design of simple and easy to use command interfaces for drone control systems.

3.4 Revised Shot Types and Grammar for Filming using a Single Drone

The full list of shot types required for shooting sports events with MultiDrone, elicited from the survey of drone professionals [6], is given in Appendix A. Based on these requirements a number of types of shot class (given in Table 1) were defined within a hierarchical structure [8]. The different shots provided most of the types of camera transition (relative to the target subject) that were thought necessary for filming sporting events such as races. The Chase, Lead and FlyBy sub-classes are typical of the shots used in the filming of cycling races and are currently often accomplished using a camera on a motorbike. The Static shots such as Fixed Cam are often used when filming races to give a view from the perspective of a spectator. The Orbit shot is commonly used in drone cinematography to highlight a subject whilst giving an awareness of it's place in the environment [26]. The Establishing shot is also widely used in cinematography to set the context for a scene [20].

The shot classes were defined as follows:

- FREE-FLYING [Scene/target-oriented]: This is the master category in which the drone follows an arbitrary trajectory. The shot can be scene or target orientated with the drone trajectory programmed in absolute coordinates or relative to a tracked target. Each of the 6 degrees of freedom of the drone and camera can be controlled independently using an arbitrary function. This class can also be used

Table 1: The Revised Shot Types for MultiDrone

S: FREE-FLYING	S0: STATIC	S0.0: FIXED CAM
		S0.1: ROTATE CAM
		S0.2: TRACK CAM
	S1: FLY-THROUGH	S1.0: FIXED-CAM
		S1.1: ROTATE-CAM
	S2: APPROACH	S2.0: ESTABLISH
		S2.1: CHASE
		S2.2: LEAD
		S2.3: FLYBY
	S3: ELEVATOR	S3.0: ASCENT
		S3.1: DESCENT
	S4: ORBIT	S4.0: FULL ORBIT
		S4.1: PARTIAL ORBIT

to combine multiple shots of its sub-class types. The FREE-FLYING category is at the root of the class hierarchy and all other shots are are defined as sub-classes of this super-class as follows:

- STATIC [Scene/target-oriented]: This is the simplest class of shot in which the drone remains stationary and the camera can be stationary or moving. Three sub-types are defined for this class:
  - FIXED-CAM: There is no camera movement.
  - ROTATE-CAM: The camera gimbal rotates (typically pans or tilts) at a pre-programmed angle and speed.
  - TRACK-CAM: The camera gimbal rotates to maintain focus on and track a moving target.
- FLY-THROUGH [Scene-oriented]: The drone flies through the scene, typically with a constant speed following a pre-programmed path. Two sub-types are defined for this class:
  - FIXED-CAM: There is no camera movement.
  - ROTATE-CAM: The camera gimbal rotates (typically pans or tilts) at a pre-programmed angle and speed.
- APPROACH [Target-oriented]: The drone moves linearly closer to, away from, or maintains a fixed distance to the target. The approach can be from the front or the back and the drone may be at the same or a different height to the target. Four sub-types are defined:
  - ESTABLISH: The drone typically moves closer to the target from the front at a steadily decreasing altitude.

  - CHASE: The drone chases the target from behind, with the distance between them decreasing or remaining constant.
  - LEAD: The drone leads the target with the distance between them decreasing or remaining constant.
  - FLYBY: The drone flies past the target (offset from the target trajectory) in a straight line, normally overtaking the target, with camera tracking as it does so.

- ELEVATOR [Scene/target-oriented]: The drone moves vertically straight up or down, normally tracking a target or with the camera fixed (e.g. facing vertically downwards). Two sub-types are included for this class:
  - ASCENT: The drone moves up.
  - DESCENT: The drone moves down.

- ORBIT [Target-oriented]: In the orbit class, the drone flies around the target in a part or full circle, centred at the target. Two sub-types are defined for this class:
  - FULL ORBIT: The drone flies around the target in a full circle, centred on the target.
  - PARTIAL ORIBIT: The drone flies around the target following a defined segment of a circle.

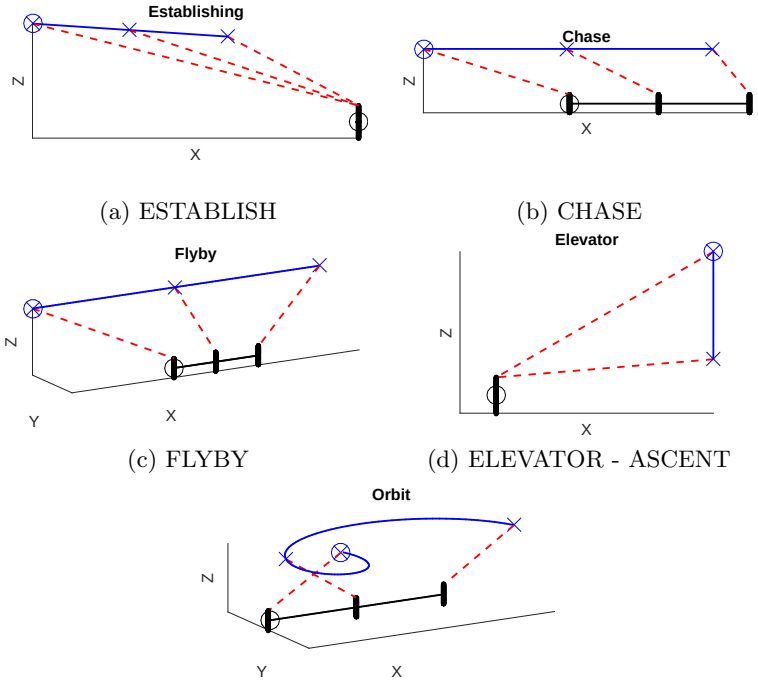
Examples of these shot types are illustrated in Figure 6.

3.5 Revised Grammar for Multiple Drone Shots

Current drone cinematography is mainly based on the use of single platforms. The use of multiple drones in a cooperating swarm can significantly enhance the cinematographic experience by providing multiple viewing angles and framing styles, innovative shot transitions and the elimination of dead time due to travel between shot locations.

The shot types described in Section 3.4 can be combined as complex shot types when multiple drones are operated independently. Different shot types can also be defined if one drone is considered as a MASTER drone following one of the trajectories defined previously, and the other drone(s) are operated as SLAVES, moving relative to the MASTER. These two modes are illustrated in see Figure 7 and summarised below:

- **M1: MASTER/SLAVE**: The MASTER drone follows one of the trajectories defined for single drone shot types, with the SLAVE drone(s) flying along a pre-programmed path offset from the trajectory of the



(a) ESTABLISH

(b) CHASE

(c) FLYBY

(d) ELEVATOR - ASCENT

(e) FULL-ORBIT

Figure 6: Five Typical Target-based Shot Types. Blue curves represent drone movement and black curves represent the target object movement. The red dotted lines indicate the gimbal rotation angles. The starting points of the drone and target are denoted with circles.

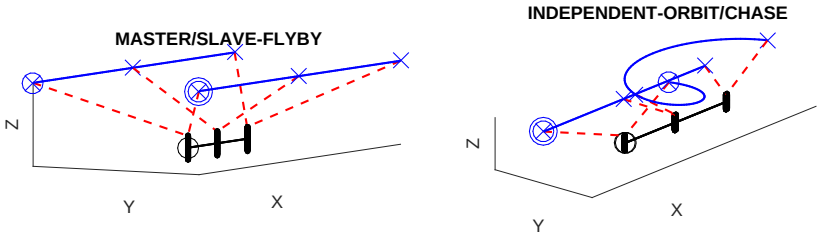
MASTER with their camera tracking the same target as the MASTER but from a different viewpoint.

- **M2: INDEPENDENT**: Multiple drones fly independently following different trajectories (as defined for a single drone shot), typically following or tracking a common target.

The use of multiple drones allows a more complex shot specification. Each drone can fly independently (e.g. each executing one of the previously defined single drone shot types) or the drones can be operated in a MASTER/SLAVE configuration.

3.6 Parametric Shot Definitions

Integration of a camera shot in a system having an automatic shooting capability requires that the shot be fully parameterized, enabling the control system to calculate the position and orientation of the drone and the gimbal rotation of the camera throughout the shot.



(a) MASTER/SLAVE (b) Independent

Figure 7: Example MultiDrone Shot Types [6]. Blue curves represent drone motion and black curves represent target motion. Red dotted lines indicate the gimbal rotation angles. Starting points of the Master drone and target are denoted with a single circle, starting point of the SLAVE with 2 circles.

3.6.1 Establishing Shot

An **Establishing** shot type (illustrated in Figure 6(a)) is used to present scene context by showing the audience the wider area in which the action will take place and can also help to set the mood. In this shot the drone moves linearly towards the target from the front (or the back in some cases), typically at a steadily decreasing altitude. During the shot the camera gimbal normally rotates so that the target is correctly framed. Assuming a fixed target with movement $\mathbf{T}(t)$ (a function of time t) defined by:

$$\mathbf{T}(t) = (0, 0, 0) \tag{9}$$

The drone movement $\mathbf{D}(t)$ relative to the target for the establishing shot is given by:

$$\mathbf{D}(t) = \left(\frac{t}{t_0}(x_e - x_s) + x_s, 0, \frac{t}{t_0}(z_e - z_s) + z_s\right) \tag{10}$$

In the above equation t_0 represents the shot duration and x_s, x_e, z_s and z_e are parameters which determine the drone start and end positions (relative to the target) on the x and z axis respectively.
The roll, pitch and yaw angles (with a yaw of 0° corresponding to the direction of drone travel) defining the gimbal motion $\mathbf{G}(t)$ relative to the drone are given by:

$$\mathbf{G}(t) = \left(0^\circ, \arctan\left(\frac{t(z_e - z_s) + t_0 z_s}{t(x_e - x_s) + t_0 x_s}\right), 0^\circ\right) \tag{11}$$

3.6.2 Chase/Lead Shot

In the **Chase/Lead** shot (illustrated in Figure 6(b)) the drone follows (or leads) the target from behind (or from the front) with the gimbal keeping the camera focused on the target. The distance between the drone and

the target can be either constant, for a follow shot, or decrease over time. The position of the target $\mathbf{T}(t)$ (a function of time t), assuming a linear movement along the x axis, is given by:

$$\mathbf{T}(t) = \left(\frac{t}{t_0}d_0, 0, 0\right) \tag{12}$$

The drone movement relative to the target $\mathbf{D}(t)$ can be defined by:

$$\mathbf{D}(t) = \left(\frac{t}{t_0}(x_e - x_s) + x_s, 0, z_0\right) \tag{13}$$

In the above equation x_s and x_e are parameters which represent the start and end positions (relative to the target) of the drone along the x axis. Parameter z_0 represents the difference between the height of the drone and that of the target object (measured along the z axis).

The gimbal motion $\mathbf{G}(t)$ relative to the drone can also be described by:

$$\mathbf{G}(t) = \left(0°, \arctan\left(\frac{t_0 z_0}{t(x_e - x_s) + t_0 x_s}\right), 0°\right) \tag{14}$$

3.6.3 Flyby Shot

In the **Flyby** shot (illustrated in Figure 6(c)) the drone flies past the target in a straight line with the camera gimbal rotating to keep the target object in frame.
The position of the target $\mathbf{T}(t)$ (a function of time t), assuming a linear movement along the x axis, is given by:

$$\mathbf{T}(t) = \left(\frac{t}{t_0}d_0, 0, 0\right) \tag{15}$$

The drone movement relative to the target $\mathbf{D}(t)$ can be defined by:

$$\mathbf{D}(t) = \left(\frac{t}{t_0}(x_e - x_s) + x_s, y_0, z_0\right) \tag{16}$$

In the above equation x_s and x_e are parameters which represent the start and end positions (relative to the target) of the drone along the x axis. Parameters y_0 and z_0 are the distances between the drone and the target object along the y and z axes respectively.

The gimbal motion $\mathbf{G}(t)$ relative to the drone is given by:

$$\mathbf{G}(t) = \left(0°, \arctan\left(\frac{t_0 z_0}{t(x_e - x_s) + t_0 x_s}\right), -\arctan\left(\frac{t_0 y_0}{t(x_e - x_s) + t_0 x_s}\right)\right) \tag{17}$$

3.6.4 Elevator/Ascent and Descent Shot

In an **Ascent** shot (illustrated in Figure 6(d)) the drone moves vertically upwards whilst in the **Descent** shot the drone moves vertically downwards. The camera gimbal will normally rotate so that the target remains correctly framed at all times. The position of the target $\mathbf{T}(t)$, assuming it is stationary, is given by:

$$\mathbf{T}(t) = (0, 0, 0) \tag{18}$$

The drone movement relative to the target $\mathbf{D}(t)$ can be defined by:

$$\mathbf{D}(t) = \left(x_0, 0, \frac{t}{t_0}(z_e - z_s) + z_s \right) \tag{19}$$

Here z_s and z_e are pre-determined parameters which represent the start and end positions of the drone (relative to the target) along the z axis. If $z_e > z_s$ the shot is an Ascent whilst it is a Descent for $z_e < z_s$. The Parameter x_0 represents the distance (along the x axis) between the drone and the target object, which remains constant throughout the shot.

The gimbal motion $\mathbf{G}(t)$ relative to the drone can be defined by:

$$\mathbf{G}(t) = \left(0^\circ, \arctan\left(\frac{t(z_e - z_s) + t_0 z_s}{t_0 x_0} \right), 0^\circ \right) \tag{20}$$

3.6.5 Orbit Shot

In an **Orbit** shot the drone flies around the target in a circle (Full Orbit) or in a circular arc (Partial Orbit), whilst simultaneously following it's trajectory (if any). The camera gimbal slowly rotates so that the target is always correctly framed. A Full Orbit Shot of a linearly moving target is illustrated in Figure 6(e). The position of the target $\mathbf{T}(t)$, assuming a linear movement along the x axis, is given by:

$$\mathbf{T}(t) = \left(\frac{t}{t_0} d_0, 0, 0 \right) \tag{21}$$

In the above equation parameter d_0 represents the end position of the target object along the x axis.

Assuming the drone is initially located at an angle of 0° relative to the position of the target (in the direction of the x axis) and travelling in the $-y$ axis direction, the drone motion relative to the target $\mathbf{D}(t)$ can be defined by:

$$\mathbf{D}(t) = \left(r_0 \cos\left(\frac{2\pi t}{t_0} \right), -r_0 \sin\left(\frac{2\pi t}{t_0} \right), z_0 \right) \tag{22}$$

In the above equation parameters r_0 and z_0 represent the radius of the orbit and the height of the drone respectively. At time t_0 the drone will have

completed one full orbit. To define a partial orbit (or multiple orbits) the factor 2π is changed to the angle subtended by the orbit.

The gimbal motion $\mathbf{G}(t)$ relative to the drone can be defined by:

$$\mathbf{G}(t) = \left(0^\circ, \arctan\left(\frac{z_0}{r_0}\right), -90^\circ\right) \tag{23}$$

3.7 Summary

A rationalized library of camera shot types, with camera motions typically defined relative to that of the target, incorporating a simple taxonomy and terminology, will be of benefit for filming fast moving live action such as sports. Such an optimized library can help to ease the task of the director and simplify the control system of a semi-autonomous filming platform.

Integration of camera shots into an automatic shooting system requires that default values and operating ranges for each parameter are defined. The quality of the viewing experience for a shot will be subjective in nature and in many cases will be highly dependent on the particular parameter values used. The next section examines a subjective testing methodology that can be used to determine parameter values for a shot that will give a good experience for the majority of viewers.

4 The Determination of Optimum Shot Parameters using Subjective Testing

Some of this work has previously been published in the following paper:

S. Boyle, F. Zhang, and D. R. Bull, "A Subjective Study of the Viewing Experience for Drone Videos," in *2019 IEEE International Conference on Image Processing (ICIP)*, (Taipei, Taiwan, Sep. 22–25, 2019), IEEE, 2019, pp. 1034–1038. DOI: 10.1109/ICIP.2019.8803747

New types of camera shot developed for drone and multi-drone filming platforms will need to be tested to ensure the quality of the viewing experience. The range of values for each shot parameter that gives a pleasant or exciting result and the ranges that give an unpleasant experience need to be evaluated. To determine the optimum value and operating envelope for a particular parameter of a camera shot it is necessary to subjectively test a series of video sequences of the shot, each varying in the value of that parameter, whilst keeping other aspects of the shot as similar as possible. Creating test footage with a real drone is both time consuming and costly. A great deal of preparatory work, such as gaining necessary permissions, procuring resources and flight planning, needs to be carried out before the actual filming. The resources (people, venues and vehicles etc.) needed to recreate typical filming scenarios such as a cycling or car race are expensive. The production of video sequences which are identical apart from a precise parameter variation is also problematic due to the difficulty in accurately setting and maintaining drone parameters throughout a shot and the variable lighting conditions encountered if filming outdoors.

The use of computer simulated drone camera shots for the subjective tests can potentially give a number of benefits. The drone camera and subject objects can be precisely controlled and the model can be quickly modified to change parameter values such as drone height. There are also many realistic 3D models of environments and objects available for download from websites which can be used to quickly build a scenario. The assumption that the subjective testing of simulations can give comparable results to that obtained with real footage will require that the models are sufficiently realistic and the video renderings free from distracting artifacts.

A series of experiments were performed to determine if the optimization of camera shot parameters (for use in automatic shooting systems such as MultiDrone) could be achieved through the subjective testing of shot simulations. The shots used for the tests were selected from the shot library defined for the filming of sports races given in Section 3.4 and Unreal Engine 4 game engine software produced by Epic Games was chosen as the simulation environment.

4.1 Simulation of Shot Scenarios using Unreal Engine

The use of simulations rather than real footage for the subjective testing of camera shots can lower time and resource costs. It enables precise control over shot parameters and simulation models can be easily modified to produce different shot versions (e.g. with changes in a single parameter).

Unreal Engine 4 was found to fulfil all the requirements necessary for shot simulation. It has a number of advantages over other systems, one of the most important for subjective testing being the production of a very realistic rendered output which is of much higher quality than that produced from simulation packages such as the Gazebo robotics simulator. Unreal Engine is available without subscription cost, payment only being needed if the developed system is monetized and gross revenue exceeds a particular value. In addition to games development, Unreal has been used for the production of special effects in the film industry, training simulations, architectural walk-throughs and in design validation. Some of the main features provided by Unreal [27] include:

- Models can be built quickly using off the shelf components such as environments, vehicles and riders.
- Cameras and targets can be animated in the *Matinee* and *Sequence* editors, using keypoints to define motion.
- Character animation (e.g. for wheels turning, cyclists pedalling).
- Advanced lighting model using global illumination with effects such as dynamic shadows, diffuse inter-reflection and ambient occlusion.
- NVIDIA PhysX physics engine enables objects to be given a realistic physical behaviour (e.g. to model collisions, forces, dynamics and gravitational effects).
- Advanced model behaviour can be programmed using the *Blueprints* visual scripting or C++ languages.
- Materials are created using visual scripting with *Material Expression* nodes to produce High Level Shading Language. A large number of effects such as bump offsets, refraction, layered materials and detail texturing can be implemented. The UV coordinates of a texture can be animated to create moving materials (e.g. to model fire or water).
- Import of models from other systems such as 3ds Max using FBX (Filmbox) and OBJ (Wavefront Advanced Visualiser) formats.
- Real-time high quality shaded animation enables rapid prototyping of models.

- Very high quality rendering to video (e.g. 1920x1080 at 60 fps).
- Supported on a wide range platforms including Windows, Linux, Android, iOS, Xbox and PlayStation.

Unreal Engine also provides the 'Cine Camera Actor' camera object which has many configurable properties allowing it to simulate real-world cameras:

- Filmback Settings - e.g. 16:9 Film and IMAX 70mm.
- Lens Settings - e.g. Universal Zoom and 50mm Prime f/1.8.
- Focal Length.
- Focus Distance.
- Aperture.
- Horizontal FOV (Field of View).
- Rendering Features - e.g. Motion Blur.

There are are inherent problems from using videos of simulated camera shots in subjective testing. Computer generated images will have artefacts such as jagged edges due to limitations in the rendering and the computer screen resolution. It can be difficult and time consuming to achieve a convincing motion for moving objects in an animation. A very smooth object motion is often unrealistic, but attempting to overcome this by adding extra key-points to the animation track can result in abrupt transitions in the motion. The effect of sound is difficult to replicate, for example extra excitement can be added to a close-up shot of a racing car by adding the noise of an engine revving. To be confident in the results obtained through subjective testing it will be necessary for the simulation to have sufficient realism for the viewers to be able to concentrate on the particular issue being tested and not in the overall quality of the model.

4.1.1 Modelling Scenarios and Camera Shots using Unreal Engine

The simulations of shooting scenarios can be created using the *Matinee* or *Sequence* editors of Unreal Engine. These multi-track editors use key-framing to animate objects and can be used for the production of cinematic quality video sequences. For the case of the *Matinee* editor, moving target objects, such as cars and cycles, can be animated by selecting the object and then using the *Add New Empty Group* option. A *Movement Track*, used to define the object's motion, is then be added to the group. Key-points are added at particular times on the *Movement Track* and with a key-point selected, the orientation and position of the object is modified to that required

at the time corresponding to the key-point. To animate the drone camera a *Camera Actor* object can be added to the editor using the *Add New Camera Group* option and it's *Movement Track* defined using key-points. The value for the FOV (Field of View) property of the camera object can be adjusted to give appropriate framing. Rather than use a *Camera Actor* object, a *Cine Camera* object can be added to the animation using a similar method as that for adding target objects. The *Cine Camera* object has many more configurable properties than a *Camera Actor* object and it can be used to accurately model real-world cameras. To define camera cuts the *Add New Director Group* option is used to add a *Director* track, to which key-points are added for each cut. When using a single camera a single cut is made at the start of the *Director* track. A useful feature of the *Matinee* editor is the ability to preview the animation at particular speeds such as 50% or 100%. Using this feature the camera and target motions can be quickly refined to give the desired animation. Finally the *Movie* command can be used to generate high quality video rendered at full HD (e.g. 1920×1080 resolution at 60fps).

The simulation of vehicles such as cars or bicycles in a Game Mode of Unreal Engine is usually achieved by means of components such as *WheeledVehicleMovementComponent4W*, which define the vehicle setup and dynamic behaviour, and vehicle control blueprint classes such as *VehicleWheel* and *WheeledVehicle*, which use this setup to calculate vehicle dynamics (e.g. speed, acceleration and turn rate) in response to user input. Vehicles are typically modelled as a *Skeletal Mesh* object containing a hierarchy of bones connected via sockets. The vehicle control blueprints can be used to set appropriate rotations for bones (e.g. the wheel bone turn angle in response to a steering input) and control animation blueprints (e.g. to animate wheels spinning by modifying the wheel bone rotation). The vehicle control blueprints cannot easily be used to provide precise control of moving objects throughout a shot simulation and hence to accurately model camera shots the objects will need to be moved by specifying key-point positions (e.g. using a *Movement Track* in the *Matinee* editor). As a result, the facilities provided by the vehicle control blueprints for animating wheels will not be available and it may be necessary to create custom animations for wheel rotation, which can be added to an *Animation Track* applied to the object in the *Matinee* editor.

4.2 Subjective Testing of Camera Shots for the MultiDrone System

A number of the proposed shot types for filming sports races with the MultiDrone platform (see Section 3.4) were evaluated by subjectively testing videos of the camera shots simulated using Unreal Engine. Representative sporting event scenarios of bicycle and car races were used for the test con-

tent. A pilot study was used to determine the feasibility of the approach and after this was confirmed three additional phases of testing were carried out. In the first phase the optimum drone camera height for each shot under test was evaluated. In the second phase of testing the optimum drone speed for shots was evaluated (using the optimum drone heights found in the first phase). This approach assumed optimum drone height and speed were independent of each other. It was not feasible to test all combinations of height and speed due to the excessive time required for modelling and the probability of fatigue affecting the scores from the test participants. In a final phase of testing, used to verify the results found previously, real footage of a cycling race, using the camera shot types under investigation, was subjectively tested and the results compared with those from the simulations

4.2.1 Methodology

The perceived video quality in subjective tests can influenced by many factors including:

- The display properties such as size, resolution, contrast, brightness and dynamic range.
- The distance of the viewer from the screen.
- Environmental factors such as room lighting and noise.

These viewing conditions should be kept as consistent as possible throughout all executions of a particular test so that the effect on the test results is minimized. Test results obtained from each viewer will be dependent on factors such as level of interest, fatigue, visual acuity, expectations and the level of expertise in the subject being filmed and in video assessment. To minimize the effects of these factors there should be at least 15 assessors which have been screened to ensure they have normal colour vision. Test sessions should be limited to 45 minutes to prevent fatigue. Sufficiently accurate results can be obtained with test sequences lasting between 5 and 10 seconds [28].

Methodologies such as Double Stimulus Continuous Quality Scale (DSCQS) and Double Stimulus Impairment Scale (DSIS) , in which the image or video clip to be evaluated is compared against a benchmark, were not suitable for the testing of MultiDrone shots because there were no video sequences that could be used as a reference. For MultiDrone testing it was necessary to use a Single Stimulus methodology such as Single Stimulus Continuous Quality Scale (SSCQS) or Absolute Category Rating/Single Stimulus Discrete Quality Scale (ACR/SSDQS). In an SSCQS test viewers typically rate a video clip with a score from 0 to 100. In ACR viewers typically rate each clip using categories of bad, poor, fair, good and excellent while in SSDQS clips are typically rated at 5 for excellent, 4 for good etc. A

Forced Choice methodology such as Two Alternative Forced Choice (2AFC) in which the viewer chooses their preference between two sequences could also be used.

For the subjective testing of MultiDrone shots a single stimulus SSDQS methodology was chosen, in which clips were rated from 1 (bad) to 5 (excellent). A Mean Opinion Score (MOS) and a confidence interval could then be easily calculated from the results.

A total of 79 subjects (20 for the pilot study, 20 for the phase I height tests, 19 for the phase II speed tests and 20 for the validation tests) participated in the experiments, all of whom had been screened to ensure they had normal or corrected-to-normal vision. The experiments were undertaken in a environment resembling a living room with subdued lighting using a SONY KD-65ZD9 4K HDR television and a viewing distance of 254 cm (three times the screen height). The test sequences were displayed in a random order under the control of a Windows PC running Matlab Psychtoolbox. Each test session started with the participant viewing a 3s mid-level grey screen before three clips used for training and then the actual video clips under evaluation (in a randomly selected sequence). After each individual clip the subject was given an unlimited time to rate their viewing experience using an integer score from 1 to 5 (1=Bad, 2=Poor, 3=Fair, 4=Good, 5=Excellent) and this raw opinion score was recorded. After the session the participant was informally interviewed about their viewing experience, to ascertain the criteria they had used to rate the videos and what factors had made any sequences appear particularly good or bad. After all tests were completed the Mean Opinion Score (MOS) and standard error (SE) were calculated for each clip from the scores of all the participants. Outlier removal was not used because of the expected wide variation in the perceived quality of the viewing experience for a shot, since it will be highly dependent on each individual's personal preferences.

4.2.2 Pilot Study

A pilot study was used to evaluate if the proposed method of subjectively testing a simulated camera shot could be successful in determining any optimum parameters for the shot. Scenarios of sports cars racing in a city and a motorcycle being ridden around an industrial area were used for the tests. To build the simulations models of the environments, sports cars, motorcycle and rider were obtained from the Unreal Marketplace using assets *DownTown*, *Industrial City*, *RaceCourse* and *MotorCycles*. Table 2 summarises the shot type and parameter values tested for each scenario under investigation. In the 'Getting ready' phase of a scene the target objects are stationary (i.e. getting ready for the race). Note, the Reveal shot at the beginning of scenario 6 is similar to an Establishing shot, but the camera transitions from a close-up shot of the stationary target to a very wide shot

in order to reveal the scene. Sample frames from each scenario are shown in Figure 8 on page 37.

Figure 9 on page 39 shows the Mean Opinion Score (MOS) for each test video (each using a different value for the parameter under test) in the different scenarios of the subjective experiment. The 95% confidence interval for the MOS is represented by the bar at each of the plot points. A paired t-test was also conducted between the video having the highest average MOS and the other four test videos in a scenario. This t-test (using the MATLAB ttest function) tested the hypothesis that the mean for the difference between the scores for the particular shot version and the scores for the best version had a value of zero, the hypothesis being rejected at a significance level below 5%. The red dots/bars in each sub-figure indicate the test versions which have a significantly lower average MOS than the winning video for the scenario. A blue dot/bar indicates that there is at least a 5% probability that the MOS for the test video (when using a large number of trials) could be equal to that of the best video.

Table 2: Scenarios for the Subjective Tests of the Pilot Study.

Scenario	Target Object	Environment	Shot types	Scene	Duration	Test Parameters
1	Sports Car	Down Town	Establishing	Getting ready	10s	Speed: 2, 4 6, 8, 10 mph
2	Sports Car	Down Town	Chase	Racing	10s	Height: 1, 2 6, 10, 14 m
3	Sports Car	Down Town	Establishing + Elevator + Chase	Getting ready + Racing	30s	Speed: 8, 13 19, 24, 30 mph
4	Motor Cycles	Industrial City	Establishing	Getting reading	10s	Speed: 5, 11 16, 22, 27 mph
5	Motor Cycles	Industrial City	Orbit	Chasing	10s	Height: 25, 30 35, 40, 45 m
6	Motor Cycles	Industrial City	Reveal + Chase + Elevator	Getting ready + Chasing	30s	Height: 5, 10 15, 20, 25 m

The results show that for all of the scenarios a range of drone parameters which give a statistically significant viewing preference can be identified, although the preferred parameter values are highly content dependent and there is significant variability between the individuals tested. From the informal interviews after the tests it was found that assessors adopted very different criteria for rating the video clips and hence the results obtained are highly subjective. It was also apparent that certain drone shots can give a wow factor if used for a short time but when used for a longer duration

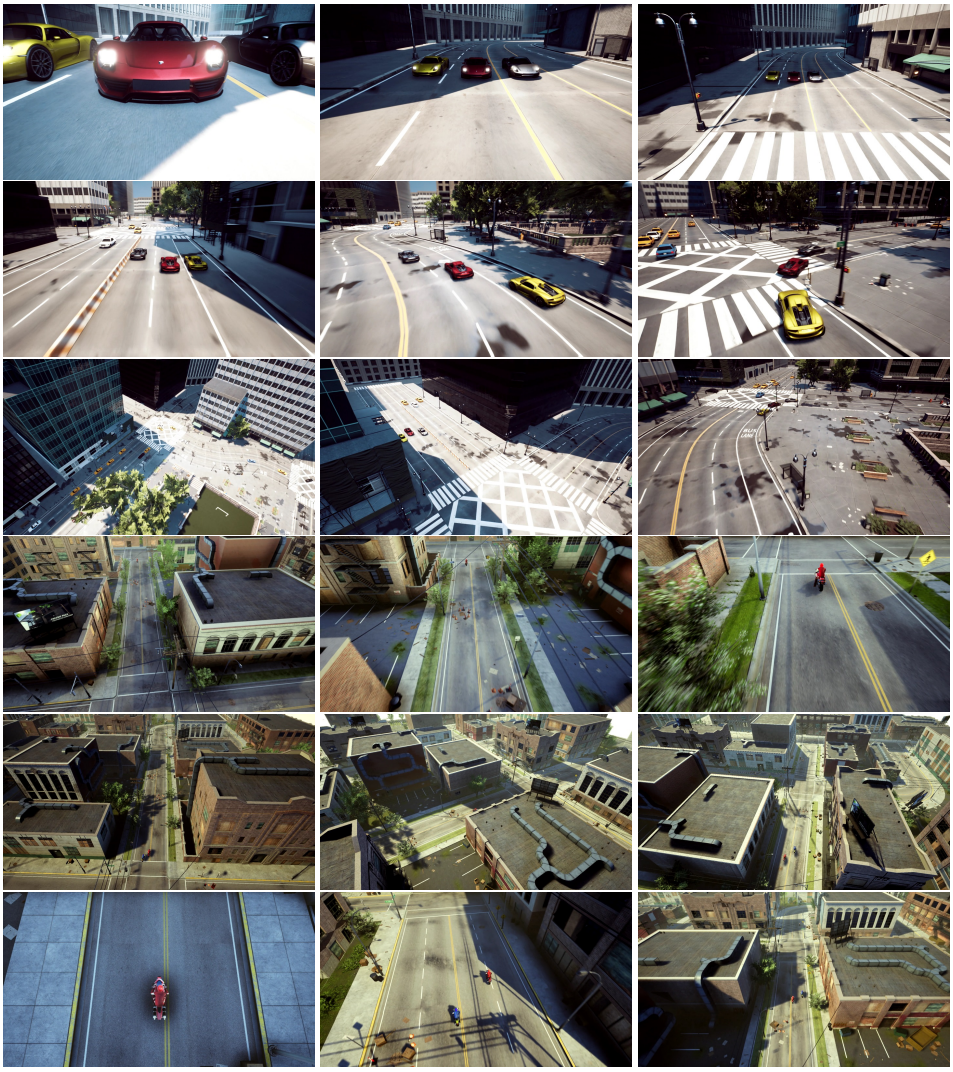


Figure 8: Sample Frames of Test Scenarios in the Pilot Study. Scenario 1 (Row1); Scenario 2 (Row 2); Scenario 3 (Row 3); Scenario 4 (Row 4); Scenario 5 (Row 5); Scenario 6 (Row 6).

they could have upsetting or disorientating effects.

The values for the optimal drone parameters clearly varied across the different test scenarios. This indicated that although an optimal parameter range can be identified for a particular test scenario, further investigation was required in order to confidently apply the results when capturing real footage. It was also noted that the optimal parameters are related to the motion (for the drone speed parameter) or the size (for the drone height parameter) of the target objects, and this dependence would also need further investigation.

From this pilot study it was concluded that subjective testing of simulated drone camera shots could provide an accurate and efficient method of assessing the quality of the viewing experience for MultiDrone camera shots. To try to reduce the sensitivity of the results on the particular context it was decided that in future tests the optimum parameters should be related to relative motion or size (e.g. speed of the drone relative to the target). It was also decided to simplify the scenarios, since for a complex shot combination the optimum parameter values could vary for each of the different shot stages. To reduce the effect of a 'wow factor' all videos used to test length parameters (e.g. drone height) should have the same duration. For parameters such as speed, a variable shot duration may be necessary to ensure each version has similar content (e.g. films identical target objects during a 'Flyby' shot).

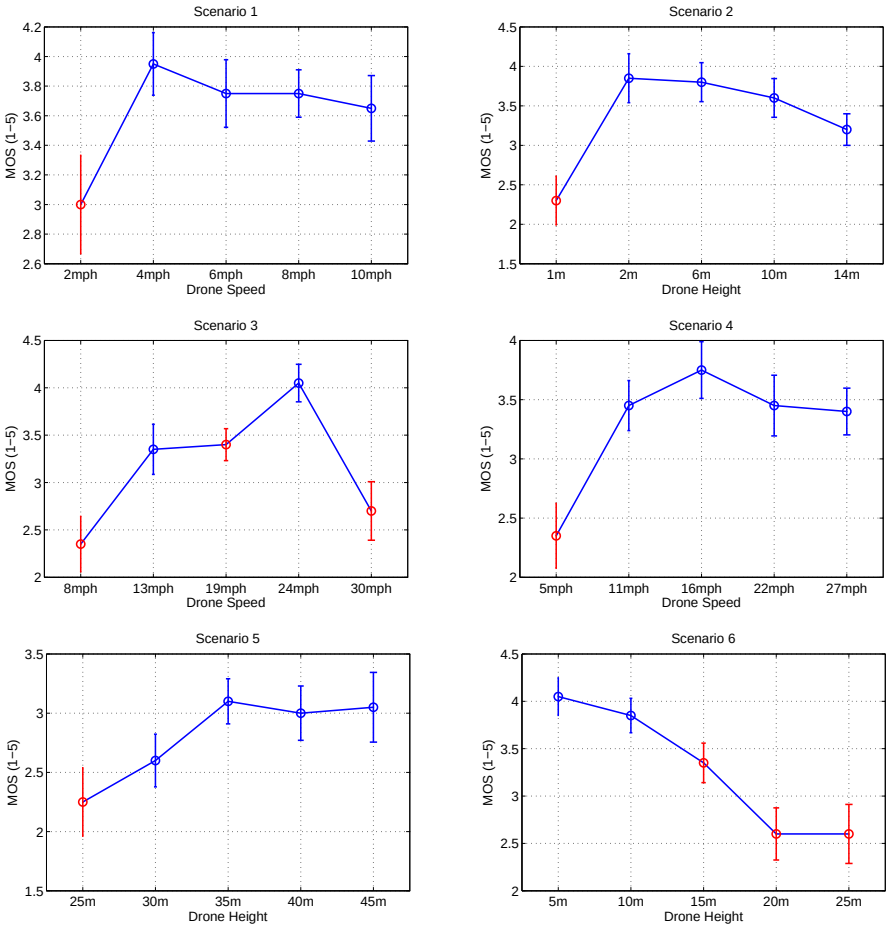


Figure 9: The Average MOS for the Test Versions of each Scenario in the Pilot Study. The error bar represents the standard error, with a red bar indicating that the MOS of the test parameter is significantly (through a paired t-test) lower than that for the best case.

4.2.3 Subjective Testing of Camera Shots to Determine Optimum Drone Height and Speed

After the verification of the subjective testing methodology in the pilot study, further experiments were carried out in two phases to determine optimum drone camera height and optimum drone speed for a selection of the shots proposed for filming races with MultiDrone (see Section 3.4). The 'Chase', 'Follow' and 'FlyBy' shots are typical of those used to film cycling races (often from a camera on a motorbike). The 'Orbit' and 'Establishing' shots are commonly used in drone cinematography to show the wider context of a scene [26]. The 'Static-Still' shot is often used when filming races to give a view from the perspective of a spectator. It can give the viewer a sense of anticipation as the racers approach and since the camera is fixed it helps to give a perception of the speed of the racers and their relative positions. The scenarios modelled for the subjective tests were a cycling race in the country-side and cars racing along a city street at night, which were representative of the types of sports events that could be filmed using the proposed shot types. In the first phase, simulations for each of the shot types in both scenarios, at various drone camera heights were subjectively tested. In the next phase, simulations for each applicable shot at various drone camera speeds were tested, with the drone height for a shot being set at the optimum value found in the first phase. The same methodology was used as for the pilot study.

The *Country side* and *Walking Street* assets from the Unreal Marketplace, which both contained long, straight sections of road suitable for modelling race stages and were of sufficient detail to produce a realistic rendering, were chosen to build the environments for the scenarios. The uniformity of the environments also helped to ensure that shots involving the same transition relative to a target object, but using different drone speeds relative to the target, would have a background of similar appearance at each stage. This would ensure that shots with differing drone speeds could be produced that were consistent in both the target and background aspects of their content.

Models of a bicycle with cyclist and sports cars, obtained from the *Cycling* and *RaceCourse* assets in the Unreal Marketplace were used in the simulations. These bicycle and car models were designed for game use and contained blueprints (e.g. *VehicleMovement*) that controlled wheel movements in response to player input. They did not include animations for wheel movement that could be incorporated into the *Matinee* editor. To enable rotation of wheels in the simulations, animations were created by editing each of the bicycle and car *Skeletal Mesh* objects and using the **Create Asset -> Create Animation -> Current Pose** command. The bone representing the wheel was selected and key-points added at four equal intervals of time. At each of these key-points the rotation of the wheel was modified by chang-

ing the value of the rotational pitch so that it varied from -180°to 180°, as shown in Figure 10. In the *Matinee* editor the animation was applied to the vehicle object by using an *Animation Track*. Key-points added to this track could then be configured to start a particular animation at a certain time and define settings such as play rate and play once or repeat. Animations were also created for the bicycle with both handlebars and wheels turning and with handlebars at a fixed rotation and wheels turning, (for use when the bicycle was changing direction), although it was not necessary to use these in the final simulations. In addition to the animation of the rotating wheels the cycling scenarios used animations of the cyclist pedalling and the pedals turning, which were available from the *Cycling* asset.

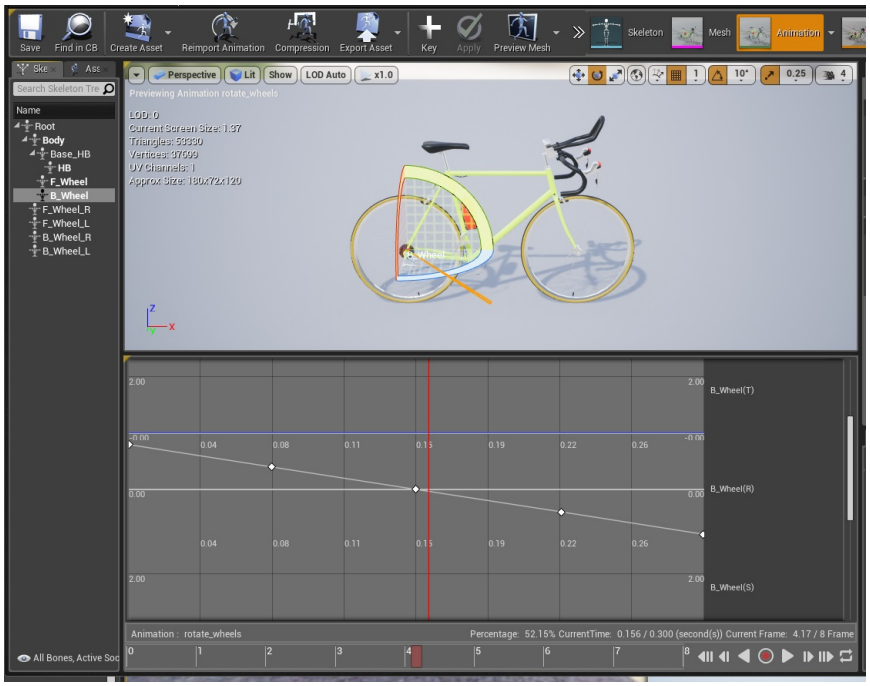


Figure 10: Animating the Bicycle Wheel.

Subjective Tests with Varying Drone Height

For the drone height tests the duration of the video clips was reduced from 10 seconds in the pilot study to 5 seconds. This duration was sufficient for the drone camera to pass from a position behind the group of target subjects to a position at the front of the group (as in the 'Flyby' shot) with a moderate speed (approximately 4 m/s relative to the target for the case of a 'Flyby' shot). This duration has also been shown to be sufficient for subjective tests [28]. The scenarios and shot types tested (given in Table 3) were modelled using the *Matinee* editor in a similar way as that for the pilot

test. In the 'Chase' shot the camera is moving closer to the target, while in the 'Follow' shot the camera is staying at a constant distance behind the target. Scenario 6 had been modelled in the pilot study (with a different but similar environment) and it was not modelled in this phase since the results would be similar. The modelling for the cycling scenarios was complicated by the undulating country-side landscape which resulted in substantially more key-points being needed on each bicycle *Movement Track* to ensure that they followed the terrain.

Examples of a cycling simulation ('Flyby' scenario) and a simulation of cars racing ('Chase' scenario) are shown in Figures 11 and 12.

Table 3: Scenarios and Parameters for the Phase I Drone Height Tests.

Scenario	Shot Type	Duration	Height m
Cycling CountrySide B1	ESTABLISHING	5s	1, 2, 3, 4 and 5
Cycling CountrySide B2	ORBIT	5s	1, 2, 3, 4 and 5
Cycling CountrySide B3	STATIC-STILL	5s	1, 2, 3, 4 and 5
Cycling CountrySide B4	FLYBY	5s	1, 2, 3, 4 and 5
Cycling CountrySide B5	CHASE	5s	1, 2, 3, 4 and 5
Cycling CountrySide B6	FOLLOW	5s	1, 2, 3, 4 and 5
Cars Night City C1	ESTABLISHING	5s	2, 4, 6, 8 and 10
Cars Night City C2	ORBIT	5s	2, 4, 6, 8 and 10
Cars Night City C3	STATIC-STILL	5s	2, 4, 6, 8 and 10
Cars Night City C4	FLYBY	5s	2, 4, 6, 8 and 10
Cars Night City C5	CHASE	5s	2, 4, 6, 8 and 10
Cars Night City C6	FOLLOW	5s	2, 4, 6, 8 and 10

The results of the subjective testing for the cycling race and cars street-racing scenarios are given in Figures 13 and 14 on pages 44 and 45. The

Figure 11: Flyby Scenario for Cycling Race.

Figure 12: Chase Scenario for Cars Racing.

results for scenario 6 ('Follow' shot) of the cars racing is not shown since this should give similar results to that of scenario 2 shown in Figure 9 on page 39.

For most of the scenarios there is a statistically significant range of drone heights that give a preferential viewing experience. In some scenarios (e.g. cycling scenario 1 and cars racing scenario 5) an optimum drone height can be clearly identified. There is less variation in preference for each scenario compared with the pilot study and this may be due to the tighter shot specifications used. In the pilot study some of the video sequences were of complex shot combinations and for these there may not be a single drone height that is preferable throughout the entire sequence. The preferences by gender for a selection of the scenarios are shown in Figure 15 on page 46. In general males scored the shots slightly higher than females and for certain shots (e.g. an 'Establishing' shot) males preferred a lower drone height.

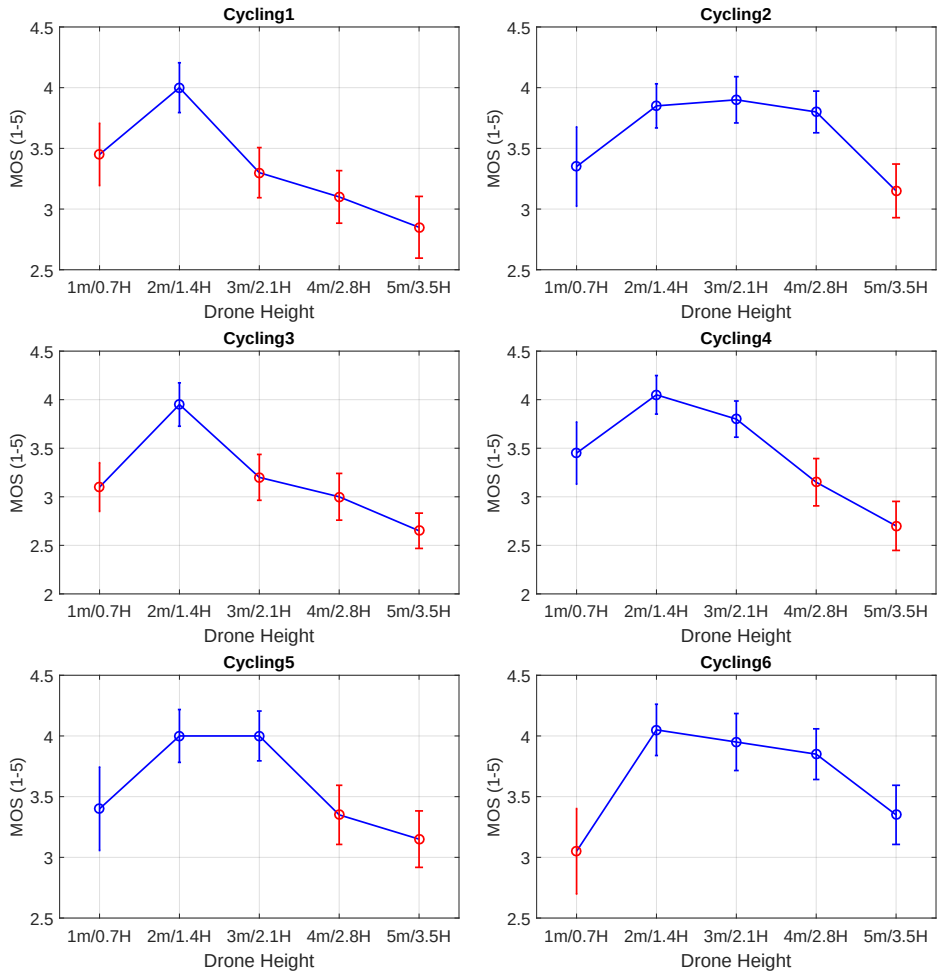


Figure 13: Results for Cycling Scenarios with Varying Drone Height. The error bar represents the standard error, with a red bar indicating that the MOS of the test parameter is significantly lower (through a paired t-test) than the best case.

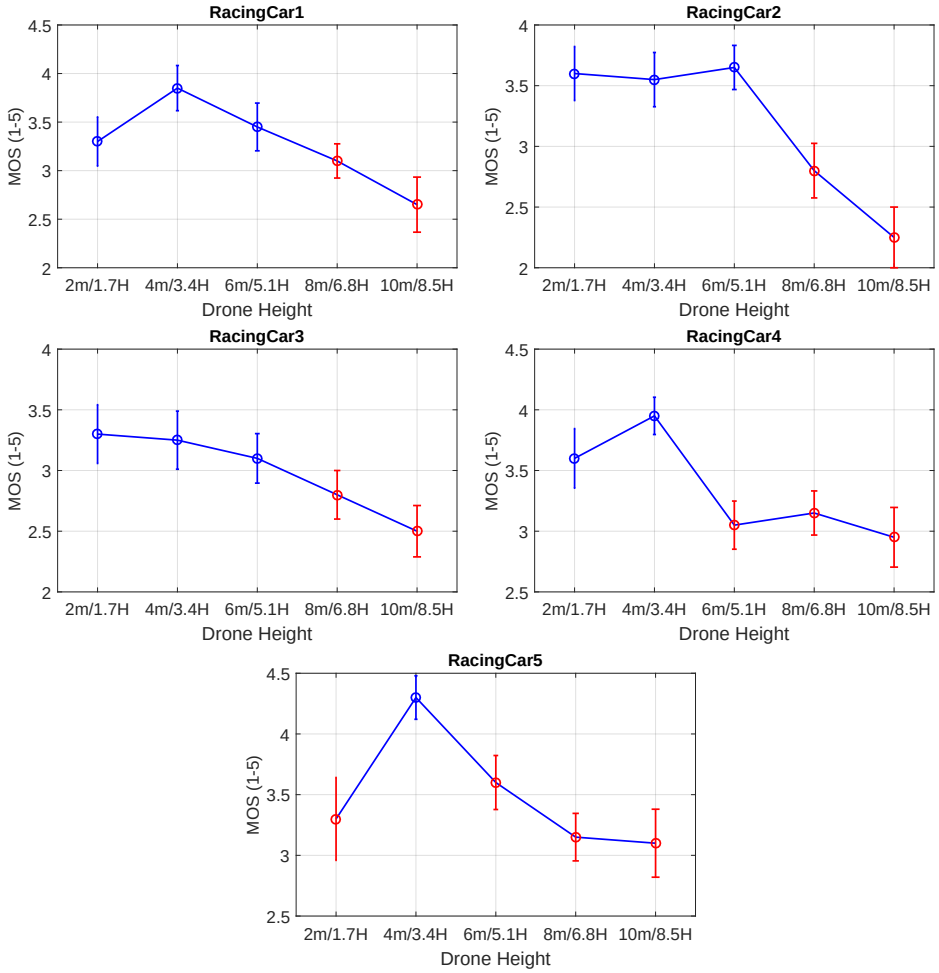


Figure 14: Results for Car Racing Scenarios with Varying Drone Height. The error bar represents the standard error, with a red bar indicating that the MOS of the test parameter is significantly lower (through a paired t-test) than the best case.

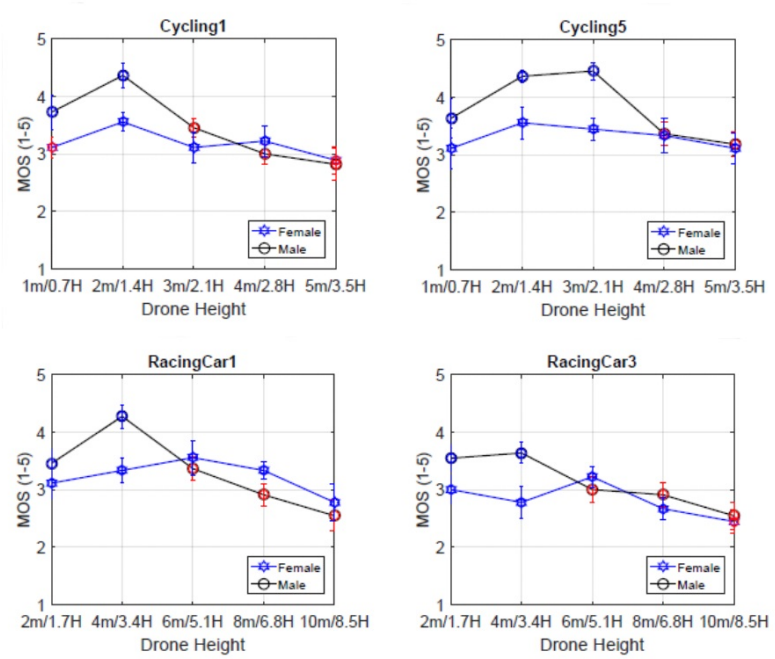


Figure 15: Gender Preferences for Drone Height with Selected Scenarios.

Subjective Tests with Varying Drone Speed

Those shot types used in the first phase of drone height tests that could be executed at different drone speeds (i.e. 'Establishing', 'Orbit', 'FlyBy' and 'Chase') were used in the second phase of testing to determine optimum speed. The shots were modelled at varying speeds for both cycling and street car racing scenarios, using the optimum drone height found from the corresponding scenario in the height tests. For each particular shot, the versions at different speed all used the same start and end positions for the camera relative to the target. This ensured that the content of each shot remained similar at each stage throughout the shot, but as a result the shot duration varied for each version. The shots types and parameters used for each test are given in Table 4.

Table 4: Scenarios and Parameters for the Phase II Drone Speed Tests.

Scenario	Shot Type	Target Speed	Speed Rel. Target m/s
Cycling CountrySide B11	ESTABLISHING	0 m/s	4, 5, 6, 7 and 10
Cycling CountrySide B12	ORBIT	8 m/s	-2, -3, -4, -5 and -6
Cycling CountrySide B13	FLYBY (Overtaking)	8 m/s	2, 2.5, 3, 4 and 6
Cycling CountrySide B14	CHASE (Closing)	8 m/s	2, 2.5, 3, 4 and 5
RacingCar Night City C11	ESTABLISHING	0 m/s	8, 9, 11, 15 and 20
RacingCar Night City C12	ORBIT	13.5 m/s	-5, -6, -7, -9 and -12
RacingCar Night City C13	FLYBY (Overtaking)	13.5 m/s	4, 5, 6, 8 and 11
RacingCar Night City C14	CHASE (Closing)	13.5 m/s	1.5, 1.8, 2.1, 2.7 and 3.8

The results from the subjective testing of the cycling race and cars street-racing simulations at varying drone speeds are given in Figures 16 and 17 on pages 48 and 49. The preference variation in the speed tests is much greater than that in the height tests and a statistically significant range for an optimum speed can only be identified for the 'Establishing' and 'Flyby' (Overtaking) shots. The viewing experience may have been affected by the variation in video duration with drone speed, but it would be difficult to overcome this problem without having a variation in the content of each sequence. The interviews conducted after each test also indicated significant

differences in preference according to gender, as shown in Figure 18 on page 49. Generally males were seen to prefer a higher drone speed than females.

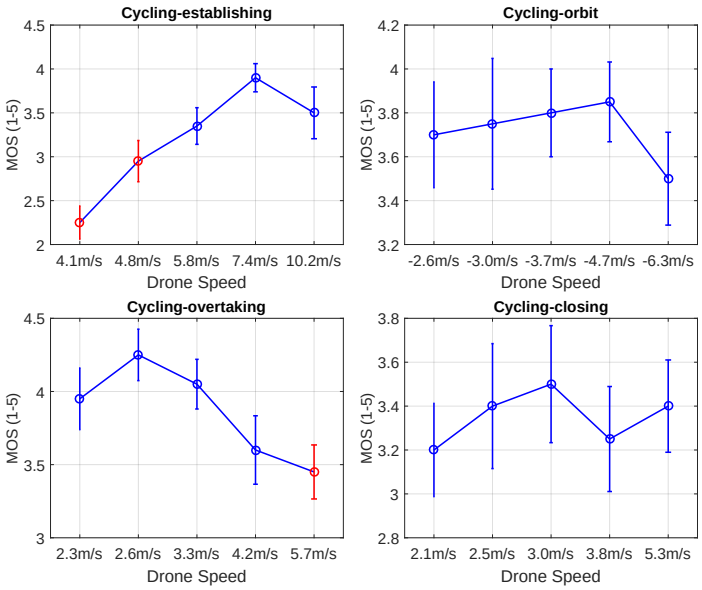


Figure 16: Results for Cycling Scenarios with Varying Drone Speed. The error bar represents the standard error, with a red bar indicating that the MOS of the test parameter is significantly lower (through a paired t-test) than the best case.

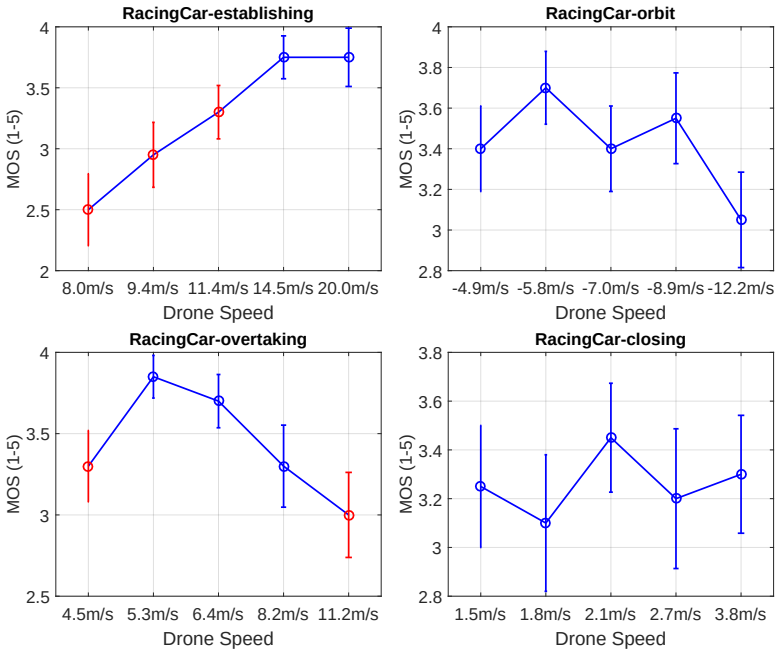


Figure 17: Results for Car Racing Scenarios with Varying Drone Speed. The error bar represents the standard error, with a red bar indicating that the MOS of the test parameter is significantly lower (through a paired t-test) than the best case.

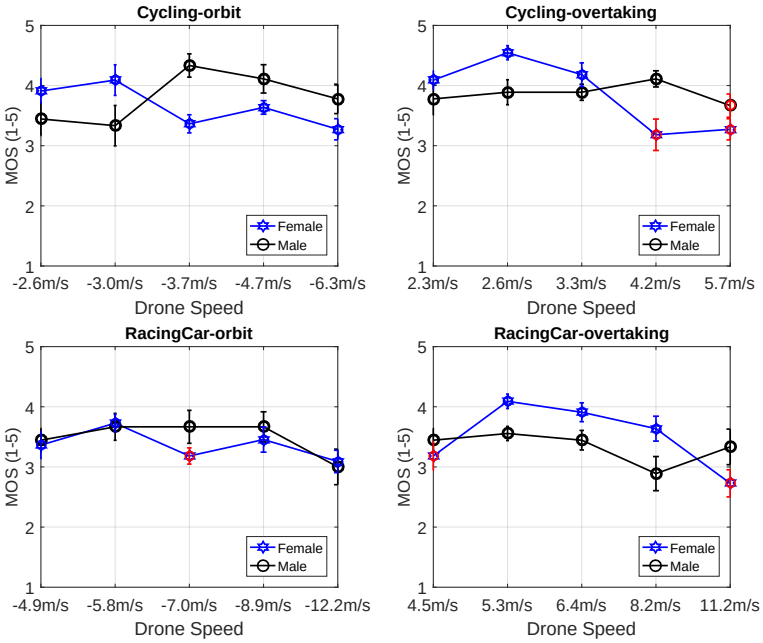


Figure 18: Gender Preferences for Drone Speed with Selected Scenarios.

4.2.4 Validation using Real Footage

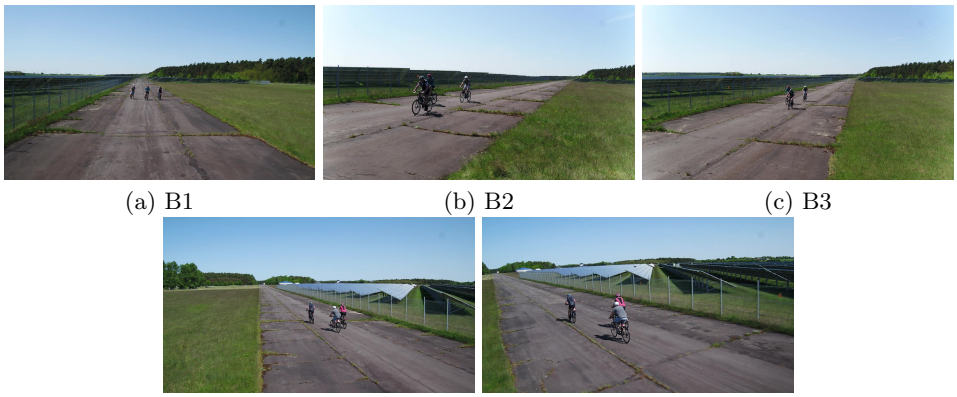
To validate the results obtained from the subjective testing of simulated footage, real video footage similar in content to that used in the cycling scenarios was shot at Müncheberg near Berlin (see Figure 20). For each of five scenarios, sequences were shot using five different drone heights, as given in Table 5. Filming employed a DJI Inspire with a Zenmuse X5S camera using a Micro 4/3 Sensor (having a crop factor of 2.0 over a 35mm full frame sensor). A variable shutter angle was used and the Panasonic zoom lens was fixed at 14mm. The effect of varying drone speed was not tested due to the difficulty in maintaining an accurate speed.

An experiment involving twenty participants (ten male and ten female), employing the same methodology as in the previous studies, was used to evaluate the optimum drone height. The results from the subjective tests, given in Figure 21, correlate well with those for the simulated videos (see Figure 13). In most cases the highest MOS occurs near or at the same height for both simulated and real videos. The differences with the 'Orbit' and 'Flyby' shots may be associated with the presence of an artefact in the simulated video (a non-smooth camera movement). The results support the proposal that simulated videos are suitable as a proxy for real footage in the subjective preference evaluation of camera shots.

Table 5: Scenarios and Parameters used for Filming Validation Sequences.

No.	Objects	Shot Type	Height
B1	Cyclist × 3	ESTABLISHING (Descent)	1, 2, 3, 4 and 5m
B2	Cyclist × 3	ORBIT	1, 2, 3, 4 and 5m
B3	Cyclist × 3	STATIC-STILL	1, 2, 3, 4 and 5m
B4	Cyclist × 3	CHASE (Follow)	1, 2, 3, 4 and 5m
B5	Cyclist × 3	FLYBY	1, 2, 3, 4 and 5m

Figure 19: A Comparison Between UE4 Animated Content and Real Footage for the same Scenario (sample frame).



(a) B1 (b) B2 (c) B3

(d) B4 (e) B5

Figure 20: Sample Frames of Test Sequences Shot at Müncheberg.

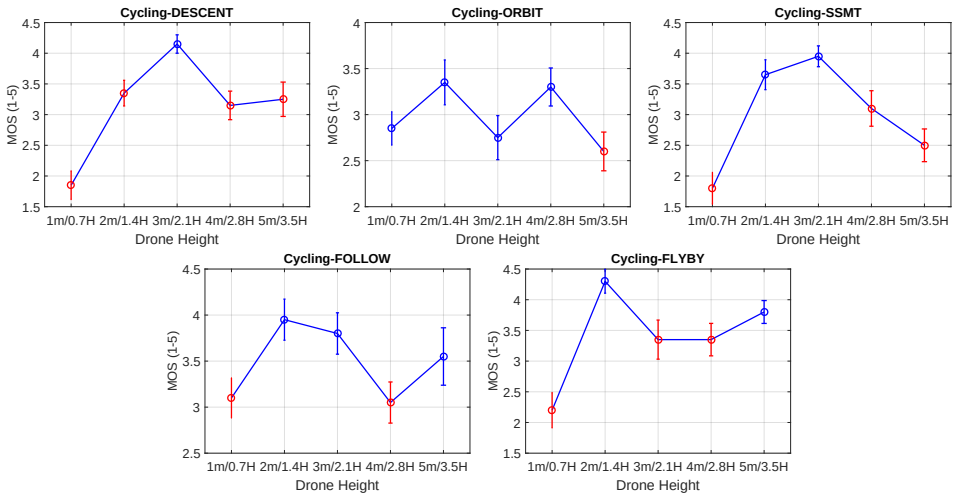


Figure 21: Test Results on the Real Footage Shot at Müncheberg. The error bar represents the standard error, with a red bar indicating that the MOS of the test parameter is significantly lower (through a paired t-test) than the best case.

4.3 Adapting Optimum Parameter Values for Specific Filming Requirements

A camera shot needs to be fully parameterized before it can be integrated into the control system of an autonomous filming platform. Many parameters such as the number of target subjects when filming a group, along with their velocities and spatial separation, will not be known in advance and will have to be determined in real-time through the target tracking system. Parameters such as the drone's speed or heading need to be defined relative to these target variables. For some shots there is relative amount of freedom in the value to use for some parameters (e.g. the distance to the side of the target when overtaking). The shot designer can set a value for the parameter in advance based on prior environmental knowledge (e.g. known road width) or the system can assign a sensible default value. Other shot parameters should be calculated by the system to fulfill the desired shot requirements and optimize the shot quality, based on results obtained from subjective testing.

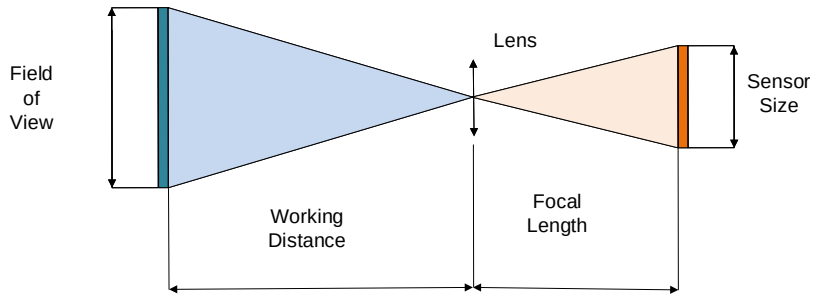


Figure 22: Relationship Between Field of View and Camera Parameters.

The optimal drone parameters obtained from the subjective tests are based on the specific camera and lens settings used to produce the shot simulations (focal length of 35mm and sensor size of 23.66mm x 13.3mm). In practice, in order to obtain a similar field of view or framing as that in the simulations, a reference working distance $\mathrm{WD}_{\mathrm{ref}}$ from the experiments (e.g. the optimum drone height) needs to be converted to a value suitable for the particular camera that is to be used for filming. The relationship between field of view (FOV), focal length (FL) and the camera sensor size (SS) is illustrated in Figure 22 and is given by Equation 24:

$$\mathrm{FL} \times \mathrm{FOV} = \mathrm{SS} \times \mathrm{WD} \tag{24}$$

The actual working distance $\mathrm{WD}_{\mathrm{act}}$ can thus be calculated by:

$$\mathrm{WD}_{\mathrm{act}} = \frac{\mathrm{SS}_{\mathrm{ref}}}{\mathrm{SS}_{\mathrm{act}}} \times \frac{\mathrm{FL}_{\mathrm{act}}}{\mathrm{FL}_{\mathrm{ref}}} \times \mathrm{WD}_{\mathrm{ref}} \tag{25}$$

In the above equation SS_{ref} and FL_{ref} are the camera parameters used for generating the simulation videos, while WD_{ref} is the reference (recommended) working distance (e.g. the optimal drone height or drone speed found from the simulations). SS_{act} and FL_{act} are the parameters for the actual camera which is to be used in filming.

4.4 Summary

The subjective testing of camera shot simulations has been found effective for evaluating the quality of the viewing experience and can be used to determine optimum values and suitable ranges for shot parameters. The amount of variation between viewers in the perceived quality varies between different shots and different shot parameters. The method was found generally to be less successful in finding an optimum camera speed for a shot than for finding an optimum height because of a wider variation in the preference for speed than in the preference for height. A selection of the shots proposed for the MultiDrone system have been evaluated using this method and a summary of the recommended parameter values for these shots is given in Table 10 of Appendix B. These parameter values are applicable for a camera similar to that used in the studies, but can be easily converted to appropriate values if using a camera with different properties.

The work of the subjective study was performed to evaluate the methodology and must only be seen as a starting point, since only a small number of shots in one particular genre (sports races) were evaluated. Automatic shooting systems will need to have shot libraries defined for each particular filming genre and it will probably be the case that similar shots used for different genres will vary in the optimum values for parameters. The subjective testing methodology discussed could also be extended to determine optimum parameters for shot transitions. For instance, when using a single drone there may be an optimum distance from the target subject at which a transition between shots (e.g. from a 'Chase' shot to an 'Orbit') should occur. For a transition between the views from two drones there may also be a minimum drone separation distance (or view angle separation) that is necessary to prevent jarring and also maximum values which should not be exceeded to prevent viewer disorientation (cf. *180 Degree Rule*).

When using simulations to optimize shot parameters there may be some variation with the type of the environment at the location (especially for wide shots) and hence it is important that the models used incorporate a realistic setting. Flight planning and training applications will also require highly accurate environment models to facilitate their use in ensuring flight safety (i.e. the avoidance of buildings and terrain) and for planning shots and flights with regard to features defining the route and landmarks that need to appear in-shot. The next section examines data sources and techniques that can be used in building accurate real-world models suitable for use in drone flight simulation systems.

5 Modelling Real-World Environments for Flight Planning and Training

Some of this work has previously been published in the following papers:

S. Boyle, M. Newton, F. Zhang, *et al.*, "Environment Capture and Simulation for UAV Cinematograhy Planning and Training," in *European Signal Processing Conference*, *Satellite Workshop: Signal Processing, Computer Vision and Deep Learning for Autonomous Systems*, 2019. [Online]. Available: https://research-information.bris.ac.uk/ws/portalfiles/portal/199953761/Environment_Capture_and_Simulation_for_UAV_Cinematography_Planning_and_Training.pdf

F. Zhang, D. Hall, T. Xu, *et al.*, "A Simulation Environment for Drone Cinematography," *IBC Technical Papers*, 2020. [Online]. Available: https://www.ibc.org/technical-papers/a-simulation-environment-for-drone-cinematography/6747.article

The models of the simulated drone camera shots, created for use in the subjective testing experiments, incorporated realistic environments of a city and a country-side setting. This was necessary to ensure the rating of shot quality was not adversely influenced by poor backgrounds and viewers were not distracted by unnatural looking content.

One of the requirements for MultiDrone was for a flight planning and training package that could be used to simulate single or combinations of MultiDrone shots with realistic target objects (e.g. cyclists, boats, cars etc). The system should also be able to provide an environment accurately and realistically modelled on any actual shooting location. The functionality of flight planning and training software can be greatly enhanced if the flight simulations integrate accurate real-world environments (i.e. showing details of terrain, buildings and roads etc). The software can then be used to fulfil many important requirements for flight planning including:

- Following features governing the route such as roads or rivers.
- Ensuring safety margin distances to buildings and the terrain are maintained.
- Determining flight times (e.g. to ensure adequate battery life).
- Identification of suitable emergency landing areas.
- Designing flights to ensure suitable coverage of landmarks which need to appear in-shot.

When using drones to film expensive 'one-take' action movie sequences it will also be of great benefit to rehearse flights and evaluate different filming strategies using simulation software incorporating accurate models of the scene (including representations of elements of the action such as the movement of vehicles and explosions).

5.1 Existing Drone Flight Planning and Training Software

A number of commercial and royalty-free software packages have been developed to simulate flying aircraft or drones within a realistic virtual environment. Examples include DJI Flight Simulator [29], AirSim [30], Zephyr Drone Flight Simulator, droneSimPro Drone Flight Simulator, RealFlight RF9 Drone Flight Simulator and Microsoft Flight Simulator [31]. Mapping animation tools such as Google Earth Studio [32] can also be used to design and preview flights.

A number of systems were examined to determine their suitability for drone flight-planning and the main features of each are summarised below:

- **DJI Flight Simulator** focuses on the training of drone pilots to improve their flying skills. Accurate simulations of flight control systems and the modelling of wind and ground effects produces a realistic drone behaviour. It offers a selection of realistic virtual environments and various weather conditions, however it does not include real-world models.

- **AirSim** (Aerial Informatics and Robotics Simulation) is an open-source plug-in package, developed for Unreal Engine and Unity (experimental version), as a platform for AI research into deep learning, computer vision and reinforcement learning algorithms for autonomous vehicles. Physically realistic drone behaviour using various flight controllers (e.g. PX4 and ArduPilot) is achieved through software-in-the-loop simulation. Weather effects (e.g. wind and rain) can be modelled and the lighting adjusted in real-time to match the time of day in the simulation (using a *BP_SkySphere* blueprint). Since AirSim is built on Unreal Engine it can be used with a wide range of downloadable environments that are available in the Unreal Marketplace. Custom made landscapes and building models would need to be used to create a realistic real-world environment. Other drawbacks include a complex user interface with a steep learning curve and since the software is no longer under maintenance there is the possibility that it may become incompatible with new drone hardware.

- **Microsoft Flight Simulator** is a flight simulator for use on Windows 10 PC's and the Xbox One platform which can be used for professional flight training. The software incorporates detailed models of

30 aircraft types (from light to wide-bodied) and 40 airports. Flights can be simulated for day or night in various conditions using real-time weather with effects modelling wind speed and direction, temperature, humidity, rain, cloud-cover and lightning. The entire Earth surface is simulated using textures and topographical data from Bing Maps, with Microsoft Azure used to generate models for buildings, trees and water. Only 341 cities worldwide (mostly in North America and Europe) are modelled in full 3D using photogrammetry [33].

- **Google Earth Studio** is a web-browser based tool, with a simple and intuitive user interface, which is used to animate Google Earth's satellite and 3D imagery data. Flight paths are created using a series of key-points defining the longitude, latitude and altitude of the camera at particular times. The camera orientation can be set using key-points to define 'Camera Target' coordinates or by using pan and tilt attributes. Properties such as the camera field of view and the position of the Sun can also be animated. The final fly-through can be rendered to video using specified values for the video resolution, texture quality and map-style.

None of the systems examined had both built-in models of the real-world and the functionality to be integrated with customized camera motions (with configurable parameters) to simulate different types of camera shot. Because of this it was decided that the MultiDrone flight planning and training system would not use a turn-key system, but would be custom built using the Unreal Engine platform. As Unreal Engine and the Unreal Engine Marketplace do not have any models of real environments, it was necessary to establish procedures to import 3D real-world data into the system.

5.2 Data Sources for Environment Models

There are many publicly or commercially available websites and databases from which data to build realistic 3D world models can be obtained. Some of the most useful are given below:

Google and Apple both provide 3D maps for many urban areas. These maps are constructed by combining 2D data from satellite and aerial photography (e.g. for modelling the ground texture) with rendered 3D data for objects such as buildings, vehicles and trees. Some areas are filled with automatically generated 3D data (e.g. generic models of trees) whilst 3D building data is created from 2D imagery using photogrammetric methods. The shape of the terrain is accurately modelled and is textured by mapping 2D aerial imagery onto the surface geometry. Most public mapping platforms, such as Google Earth and Apple Maps use data from satellite companies to supply imagery. Maxar can supply data at 40cm resolution and Planet Labs at 50cm resolution, although government restrictions can

limit the resolution available for certain areas [34].

OpenStreetMap [35] is a free website providing maps displaying a wide range of features (e.g. roads, paths, buildings, rivers and land-use). Data is added to the map by volunteers and it has extensive coverage. Details of particular features can be stored as meta-data and extracted using a query function. The OSM Buildings website [36] uses data from OpenStreetMap to produce maps with simple 3D representations of buildings, each of which can be queried for information such as type, height and number of floors. Rather than simply extruding the building outline data obtained from OpenStreetMap, some landmark buildings have been modelled more accurately. An add-on (blender-osm) for the Blender modelling system has been developed which will import OpenStreetMap models and real-world terrain data with a global coverage [37]. Features such as buildings, roads, fences, trees, rivers and the land surface are imported as separate Blender objects. The terrain is imported using digital elevation model (DEM) data in SRTM format prepared by Mapzen from a number of open data sets. The data sample spacing for individual surface points is 1 arc-second (approximately 30 meters) for latitude and longitude. The premium version of the software is able to texture the terrain by projecting satellite imagery onto the surface.

The Ordnance Survey [38] provides (at a cost) highly accurate maps of the United Kingdom for leisure or business use. Extensive information can be extracted from the topographical layers of data stored with the maps.

Free and publicly available resources for satellite data include the OpenTopography.org website [39] and the USGS EarthExplorer website [40]. The heightmap and raster image data is mostly supplied at a low resolution (e.g. 15m to 90m for ASTER Earth Data).

The data supplied by these resources is obtained using a variety of methods, the most common being high altitude aerial imagery (from satellite or aircraft), LIDAR scanning, 3D scanning techniques (e.g. 3D Laser Scanning) and standard (medium range) photography.

5.2.1 Satellite and Aerial Imagery

Satellites and aircraft are used to provide imagery of the Earth's surface with a wide coverage. Much of the publicly available data with global coverage is of low resolution, for example that from the Landsat program (a joint venture between NASA and the USGS) which has provided a continuous record of the Earth's land surface since 1972 [41]. At the present time the Landsat 7 and 8 satellites, in combination, can map the entire globe in 8 days (with each satellite completing an orbit in 99 minutes). Landsat 8 has two instruments, the Thermal Infrared Sensor (TIRS) with a spatial resolution of 100m and the Operational Land Imager (OLI) which records radiation in the visible, NIR and SWIR bands with a resolution of

30m (15m for panchromatic). Access to the collected data is free and the resource has been used by scientists studying changes in land use, urban growth and deforestation. Data at higher resolutions is available commercially. The aw3d.jp website [42] supplies data from the Advanced Land Observing Satellite (ALOS) at a global resolution of 5m and on demand at resolutions of 0.5m. Ortho-rectified imagery at resolutions up to 30cm and building data (e.g. 3D polygon representation and height) can also be supplied on demand.

Researchers such as Sirmacek *et al.* [43] have investigated the use of aerial and satellite photographs to detect urban areas and buildings. In addition to the outline of a building it is possible to determine a building's height from a single image by detecting shadows and using knowledge of the time that the image was captured [44]. By analysing multiple aerial images it is possible to extract 3D surface information for buildings [45].

In addition to providing imagery with a wide coverage over the Earth's surface, satellites and aircraft are used as platforms to collect height-map data using techniques such as RADAR and LIDAR, as described below.

5.2.2 LIDAR

LIDAR (Light Detection and Ranging) is a technique used to measure the distance to a surface by measuring the time for a pulse of laser light to be reflected back to a detector from the surface. By mounting the LIDAR apparatus on an aircraft with a GPS receiver it is possible to determine precise measurements for ground elevation. Rather than a single reflected pulse there may be multiple reflections (e.g. from the top of a tree and then from the ground). From the shape and intensity of the reflections it is possible to deduce information about the surface type and land use (e.g. whether the land is covered by buildings, fields or trees etc). Individual buildings can be discriminated from the LIDAR data and reconstructed as 3D surface models [46].

LIDAR has higher equipment and operating costs compared to 3D scanning but it has the advantage of a greater range and large areas can be scanned in a short time period. LIDAR data can yield more accurate surface heights compared to those calculated using aerial and satellite imagery. LIDAR is also generally more accurate than RADAR due to the smaller wavelength of the electromagnetic radiation employed, although for long range applications RADAR can give better results due it's lower absorption rate through the atmosphere and at reflecting ground level surfaces. A high resolution camera will often be used alongside the LIDAR equipment to take colour pictures of the ground along the flight-path.

5.2.3 3D Scanning

3D laser scanners use a laser light beam to determine the distance to the nearest object in the path of the beam. By scanning the entire field of view one point at a time a 3D picture of the surrounding environment is built up. The two most common methods to determine distance are time of flight (using a pulsed laser similar in operation to LIDAR) and phase-based using a continuous laser. Phased-based scanners have a faster acquisition speed but have a lower dynamic range. Scanners will often include additional hardware to capture High Dynamic Range colour imagery, but this can reduce the maximum possible scan rate. Many scanners have a built-in GPS receiver, which enables point clouds scanned at different locations to be automatically combined into a single data-set. A 3D model can be reconstructed from the point cloud data using software such as Autodesk ReCap.

3D laser scanning has lower equipment and operating costs than aerial LIDAR scanning. Laser scanning has a low range compared to LIDAR and hence large areas will often require a number of individual scans or the scanner to be operated continuously whilst mounted on a vehicle. 3D laser scanning at ground-level has the advantage of being able to capture detail that cannot be seen from above (e.g. the sides of buildings).

5.2.4 Close Range, High Resolution Imagery

High quality images of an environment can be used to provide data for 3D reconstruction using photogrammetry. A good quality of reconstruction for the environment model is dependent on both a good image quality (e.g. regarding the resolution, sharpness, contrast and level of shadowing) and the image set providing a complete coverage of the scene from a wide variety of angles. Images can be captured (using a still or video camera) from an aircraft or drone flying over the location multiple times in a series of different scans. However using actual flights to obtain data is expensive and time consuming. To be confident of adequate coverage, if the flight time (and hence the number of camera shots) needs to be minimized, a great deal of planning is required to optimize the flight path and camera angles. For many locations a quicker and more cost effective alternative is to capture images from resources such as Google Earth. Due to distortions and inaccuracies inherent in most of the 3D models available from these resources, the technique cannot produce as good a quality model as that ultimately possible from the use of images shot on location.

For the development of environment models to be used with the proposed MultiDrone flight planning and training application it was decided to investigate the use of height-map data to create Unreal *Landscape* objects and the use of photogrammetry to reconstruct 3D scenes from 2D images. The

3D models provided by Google Earth were deemed to be of sufficient quality for flight planning and training and so it was chosen as the image source for the preliminary work with photogrammetry. It was recognized that due to the limited 3D coverage of Google Earth, in most cases it would in practice be necessary to take photographs at the actual location to provide data for a 3D reconstruction. Sections 5.3 and 5.4 detail procedures that are typical of those used to build environment models using current software and highlight various problems regarding these methods.

5.3 Creating Landscapes in Unreal Engine

5.3.1 The Unreal Engine Landscape Object

Unreal Engine uses a *Landscape* actor object to define the terrain of a virtual world. A *Landscape* object can be created in the *Landscape Editor* and sculpted using tools such as 'Smooth', 'Flatten', 'Erosion' and 'Noise'. Landscapes can also be created by importing height-map data from a file. The *Landscape* can be textured using a specialist *Landscape Material* which can be used to define different environmental layers such as soil, grass and snow. The material can be applied to the terrain by painting the surface with a particular weighting for each layer or with the weighting defined by a height-map file. For added realism the *Foliage Editor* allows objects representing foliage such as trees, shrubs, flowers and grasses to be painted onto the *Landscape* surface (or erased from the surface).

On *Landscape* creation the size and number of vertices for the *Landscape* is defined by setting the 'Number of Components', 'Sections per Component', 'Section Size' and 'Scale' values. A *Landscape* object is composed of a number of square Components which consist of 1 or 4 (2x2) subsections. The number of vertices on each subsection edge is limited to certain values (i.e. 2^n with $n = 3 \dots 8$), resulting in a maximum number of 255x255 square divisions (Quads) per subsection. The default scale of 100 will result in a *Landscape* with 100 cm between vertices (i.e. a Quad size of 100cm x 100cm). Each Component has a CPU cost associated with rendering and Epic Games recommends a maximum of 1024 Components in a landscape. However small Components allow for quicker LOD transitions and hence for large landscapes there is a trade-off to be made between using a few large Components to reduce CPU cost and having more but smaller Components to allow faster LOD transitions. To improve performance when very large terrains are required *World Composition* can be enabled. In this mode *Landscape* objects are stored in separate levels which can be loaded and unloaded as required, each at a particular tile location in the world space.

5.3.2 Importing Landscape Data from a Heightmap File

A real-world terrain model can be created in Unreal Engine by importing surface elevation data from a height-map file into a *Landscape* object. The technique was examined, as described below, using data from the Shuttle Topography Radar Mission (STRM). However the method can be employed with height-map data obtained from other sources such as LIDAR surveys.

Example Procedure for the Creation of an Unreal Landscape from Height-map Data

Height-map data for an area around Bristol of approximately 62km^2 (with a range in latitude and longitude of 0.09°) was downloaded in GeoTiff format from the OpenTopography.org [39] website, using the SRTM GL1 data-set at 30m resolution. The output 16 bit grey-scale TIFF file had a size of 324x324 pixels and the intensity of the pixels ranged from 0 to 120, representing the height range in metres for the selected area. In addition to the height-map a hill-shade image for the area, automatically generated from the DEM data, was also downloaded.

The TIFF file was imported into Bundysoft L3DT software (Large 3D Terrain Generator) as a heightmap. Using this software the heightmap was resized to 2016x2016 pixels and exported as a 16 bit grey-scale PNG format file. This process also modified the pixel intensity values so that pixels at 120m were at full intensity (white).

In the Landscape Editor of Unreal Engine the 'Import from File' option was used to import data from the PNG heightmap (the original TIFF file at 324x324 pixels was not an allowable size for an Unreal Landscape). The 'Section Size' was set to 63x63 Quads, 'Sections Per Component' set to 2x2 and 'Number of Components' set to 16x16. The X, Y and Z scale of the *Landscape* object were modified to that required for the actual landscape, with the X scale given by:

$$\begin{aligned} X &= \text{width of actual landscape in cm / Unreal width in vertices} \\ &\approx 800000/2016 \approx 397 \end{aligned}$$

The Z scale was given by:

$$\begin{aligned} Z &= \text{Height corresponding to white level in cm / 512} \\ &= 12000/512 \approx 23.4 \end{aligned}$$

The *Landscape Material*, shown in Figure 23, was created from the hill-shade image generated by OpenTopography (which was shaded according to the height and gradient of the surface). The *Material* was applied to the *Landscape* object and the resultant landscape is shown in Figure 24. It is

possible to distinguish certain features from the Unreal landscape, such as the Avon Gorge and Clifton Suspension Bridge, even though the data used to produce the model is at a very low resolution (i.e. 30m between sampling points). A high resolution heightmap produced from an aerial LIDAR survey could produce a much more detailed model.

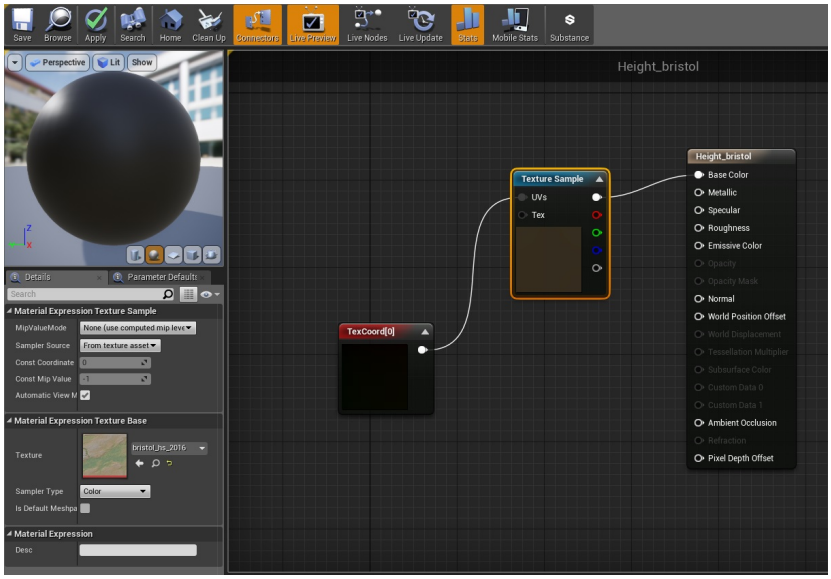


Figure 23: A Material to Apply an Image onto a Landscape.

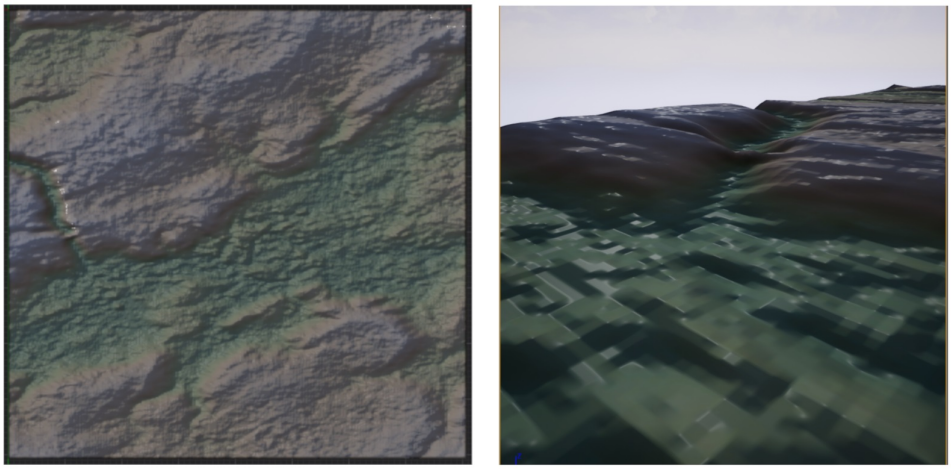


Figure 24: Landscape of the Bristol Area Created from SRTM Data.

5.4 Environment Model Creation using Photogrammetry

The technique of photgrammetry can be used to produce a 3D model using a number of 2D images captured from different viewpoints. It can be used to create environment models for drone flight-planning and training software, either from images captured on location or from images captured from 3D mapping software such as Google Earth.

5.4.1 Theory of Photogrammetry

The term photogrammetry encompasses a wide range of techniques that are concerned with the extraction of information from images. Photogrammetric methods are often employed to construct 3D models (e.g. surface models or point clouds) from a series of photographs and they have been extensively used in applications as diverse as surveying, flight training, virtual reality, simultaneous localization and mapping (SLAM), archaeology, heritage conservation and media production.

The first stage of most photogrammetry pipelines is a process known as 'Structure from Motion' which calculates the external parameters (e.g. the position and orientation) and, if the problem is un-calibrated, the internal parameters (e.g. the focal length and principal point) for the camera corresponding to each image. Camera parameters are estimated by matching well defined key-points to multiple images and calculating the homographies and fundamental matrices corresponding to pairs of matching images. The coordinates of the key-points can then be calcualted and hence this stage will often generate a 'sparse' point cloud representing the 3D positions of the key-points. Many 'Structure from Motion' algorithms employ a procedure known as 'Bundle Adjustment' in which bundles (rays of light from surface features to cameras) are adjusted to optimize both feature and camera location in parallel, so that the resultant image projection error is minimized. Image coordinates are calculated using a camera model which implements the required projection method. Typically, the camera model will use a perspective projection, although for distant cameras affine and orthographic projections are also useful. The camera model can also include parameters to model any internal aberrations such as radial distortion.

The second stage of a typical photogrammetry pipeline is a dense matching process which recovers the fine 3D structure of objects by matching surface details in multiple images and calculating their position using the camera parameters recovered from the 'Structure from Motion' phase. Whilst sparse matching typically only matches key-points having an obvious similarity, dense matching will often need to use more than one strategy to ensure enough matches for the required point density. A first stage will often use interest operators to extract feature points, with subsequent stages extracting matches based on other techniques (e.g. using image similarity

algorithms) [47]. Typically this stage will also use 'Bundle Adjustment' algorithms to refine initial estimates for surface point coordinates by minimizing the projection error.

The final stages of the photogrammetry pipeline will generate a surface mesh from the dense point cloud and then texture the mesh using data from the image-set. A photo-consistency mesh optimization process is sometimes used to adjust mesh triangles to maximize photo-consistency. This will ensure that for each mesh vertex there are similar colours at the locations in each image corresponding to the vertex re-projection position using the calculated camera parameters for the image.

Feature and Key-point Detection and Matching

In the first phase of the 'Structure from Motion' camera calibration process feature key-points are identified in each image. These key-points will need to be matched between images in which the features occur at differing size and hence a Scale Invariant Feature Transform (SIFT) is needed to extract those particular features which can be identified at different scale factors.

Lowe [48] describes a method to extract scale-invariant key-points from an image using a Gaussian difference function. A stack of scale space images $L(x, y, k\sigma)$ is formed by incrementally convolving the image $I(x, y)$ with a Gaussian function $G(x, y, k\sigma)$ using increasing values for the scale factor $k\sigma$. The image stack is divided into octaves, each corresponding to a doubling of $k\sigma$ and containing a number of images at intermediate scales. The final convolved image of an octave is down-sampled by a factor of two and this is used to generate the set of images for the next octave. Images from adjacent scales in each octave are subtracted to form a stack of difference of Gaussian images $D(x, y, \sigma)$. Candidate key-points are found by detecting the local minima and maxima in the Gaussian difference images. These can be found by comparing each sample point with it's eight neighbours in the image and nine neighbours in the scales above and below. The use of a stack of Gaussian images at different scales (i.e. differing values for $k\sigma$) enables a feature key-point identified in one image at a certain scale to be matched to another image at a different scale by searching through all levels of the stack (i.e. all scales) to find a correspondence.

Localization of each candidate key-point is performed by fitting a 3D quadratic function $D(\mathbf{x})$ to the Gaussian difference data $D(x, y, \sigma)$ at the key-point position using a Taylor expansion. Key-points with low contrast can be found and rejected by evaluating the value of the quadratic function $D(\mathbf{x})$ at the position of it's extremum (rejecting the point if it falls below a threshold value). The difference in Gaussian function will have a strong response for an edge but the position of a peak (i.e. a key-point) along the edge can be poorly defined. In such cases the function will have a large principal curvature across the edge and a small principal curvature in the

perpendicular direction. A measure of the ratio of the principal curvatures can be determined from the derivatives of the Hessian matrix for $D(\mathbf{x})$, which can be estimated from the differences between neighbouring sampling points. Any key-points with a ratio for the curvatures in excess of a threshold value should be rejected (Lowe used a value of 10 for the threshold). In addition to scale and location, another important attribute of a key-point is orientation, which Lowe calculates from the Gaussian image $L(x, y, k\sigma)$. The gradient and orientation at each sampling point (x, y) in a circular region around the key-point is calculated using pixel differences. This information is stored in an orientation histogram, with weighted gradient values accumulated into a number of bins, each representing a range of orientations (e.g. 36 bins each with a 10° range). The weighting (typically Gaussian) is used to give more importance to the values from samples close to the key-point. An orientation for the key-point can then be calculated by finding the peak in the histogram or, for greater accuracy, by fitting a parabola to the values close to a peak and using interpolation.

For purposes of key-point matching it is necessary to create descriptors for the local regions around each key-point that are invariant to changes in 3D viewpoint and to changes in illumination. Lowe creates a descriptor using orientation histograms (with 8 bins) for 4x4 regions around a key-point, each region having a size of 4x4 pixels. To achieve invariance to view rotation the coordinate system used to define the orientation of the descriptor regions and the gradients is aligned to the key-point orientation. The value of a gradient sample is distributed among adjacent histogram bins according to a weighting factor which increases as the distance of the orientation from the centre of the bin increases (an orientation at the centre of a bin will not be distributed). This, along with sampling orientations over sub-regions, gives invariance to small shifts in gradient positions and boundary effects due to the orientation of a sample point being shifted from one bin to another. A descriptor vector for the key-point is formed which contains the values from all sixteen orientation histograms. To help achieve invariance to illumination the vector is normalized so that a contrast change which multiplies each pixel by a constant will not change the descriptor.

Key-point matching (between key-points in different views) is performed by finding the nearest neighbours, defined by the Euclidean distance between the descriptor vectors, in the database of key-points. In the implementation of 'Structure From Motion' by Gherardi *et al.* [49], the number of key-points in each image is culled to 300. A k-d tree is then used to search the key-point descriptors for the eight nearest matches of each key-point (in different views), with this process taking a time of order $\mathcal{O}(n \log n)$ for n images. A 2D histogram is used to store the number of key-point matches between each pair of views. Rather than calculate fundamental matrices and homographies for each pair of matching images (with potentially the number of matches being of $\mathcal{O}(n^2)$), this histogram is used to match each

image with the eight images that have the most key-points in common with it (with the number of matches being of $\mathcal{O}(n)$).

Bundle Adjustment

For a scene having n individual 3D feature key-points given by X_p, $p = 1 \dots n$, shot in m images using cameras having internal and external parameters given by P_i, $i = 1 \dots m$ and having k calibration parameters constant across several images, given by C_c, $c = 1 \dots k$, Triggs *et al.* [50] define a predictive model that can be used to estimate the image position $\boldsymbol{x}_{ip}$ of feature p in image i:

$$\boldsymbol{x}_{ip} = \boldsymbol{x}(\mathrm{C}_c, \mathrm{P}_i, \mathrm{X}_p) \tag{26}$$

Measured values $\underline{\boldsymbol{x}}_{ip}$ for the image position of feature p in image i can be used to determine the feature projection error when using parameters X_p, P_i and C_c:

$$\Delta\boldsymbol{x}_{ip}(\mathrm{C}_c, \mathrm{P}_i, \mathrm{X}_p) = \underline{\boldsymbol{x}}_{ip} - \boldsymbol{x}(\mathrm{C}_c, \mathrm{P}_i, \mathrm{X}_p) \tag{27}$$

Starting from initial parameter estimates, the bundle adjustment process refines the parameter values so that a cost function measuring the total feature projection error is minimized. Common methods to implement the cost function include non-linear least squares, robustified least squares and intensity based methods. For the non-linear least squares method, if measurements $\underline{\mathrm{z}}_i$ are predicted by $\mathrm{z}_i = \mathrm{z}_i(\mathbf{x})$ with model parameters in vector $\mathbf{x}$, the weighted Sum of Squared Error (SSE) cost function is given by:

$$\mathrm{f}(\mathbf{x}) = \frac{1}{2} \sum_i \Delta\mathrm{z}_i(\mathbf{x})^{\mathrm{T}} \mathrm{W}_i \Delta\mathrm{z}_i(\mathbf{x}) \,, \quad \Delta\mathrm{z}_i(\mathbf{x}) = \underline{\mathrm{z}}_i - \mathrm{z}_i(\mathbf{x}) \tag{28}$$

In the above equation the W_i are chosen so that they approximate the inverse covariance of $\underline{\mathrm{z}}_i$ (resulting in a symmetric, positive definite weight matrix). The robustified least squares method uses a similar cost function but each measurement can be down-weighted so that the overall cost is less sensitive to outliers. Intensity based methods are used when comparing colours or gray-scale intensities in one image patch against another image or template, the error cost being measured in terms of intensity residuals.

Most of the common optimization procedures used in bundle adjustment are based on the second order (quadratic) Taylor series approximation for cost function $\mathrm{f}(\mathbf{x})$:

$$f(\mathbf{x} + \delta\mathbf{x}) \approx f(\mathbf{x}) + \mathbf{g}^{\mathrm{T}} \delta\mathbf{x} + \frac{1}{2} \delta\mathbf{x}^{\mathrm{T}} \mathbf{H} \delta\mathbf{x} \tag{29}$$

In the above equation $\mathbf{x}$ is the vector of parameters being optimized, $\mathbf{g} = \frac{\mathrm{df}(\mathbf{x})}{\mathrm{d}\mathbf{x}}$ (the gradient vector) and $\mathbf{H} = \frac{\mathrm{d}^2\mathrm{f}(\mathbf{x})}{\mathrm{d}\mathbf{x}^2}$ (the Hessian matrix).

Newton's method can be used to find a local minimum for f($\mathbf{x}$), converging in a sequence of steps, $\mathbf{x}_0, \mathbf{x}_1 \ldots \mathbf{x}_n$ from given initial parameters $\mathbf{x}_0$. The Newton step size is found by minimizing the quadratic approximation given in equation 29 with respect to step size $\delta\mathbf{x}$:

$$\frac{\mathrm{df}(\mathbf{x}+\delta\mathbf{x})}{\mathrm{d}(\delta\mathbf{x})} \approx \mathbf{g} + \mathbf{H}\delta\mathbf{x} = 0 \tag{30}$$

$$\delta\mathbf{x} = -\mathbf{H}^{-1}\mathbf{g} \tag{31}$$

In the damped Newton method the step size is dynamically altered, either to limit it's maximum size or to shorten it when a prediction is poor, using a weight matrix $\mathbf{W}_\lambda$ and a weighting factor λ:

$$(\mathbf{H} + \lambda\mathbf{W}_\lambda)\delta\mathbf{x} = -\mathbf{g} \tag{32}$$

For the weighted SSE cost function with error values in vector $\Delta\mathbf{z}(\mathbf{x}) = \underline{\mathbf{z}} - \mathbf{z}(\mathbf{x})$ and with weights in matrix $\mathbf{W}$ (see equation 28), a Jacobian $\mathbf{J} = \frac{\mathrm{d}\mathbf{z}}{\mathrm{d}\mathbf{x}}$ can be calculated from the predictive model. The gradient and Hessian for the cost function are then given by:

$$\mathbf{g} = \frac{\mathrm{df}}{\mathrm{d}\mathbf{x}} = \Delta\mathbf{z}^{\mathrm{T}}\mathbf{W}\mathbf{J} \tag{33}$$

$$\mathbf{H} = \frac{\mathrm{d}^2\mathrm{f}}{\mathrm{d}\mathbf{x}^2} = \mathbf{J}^{\mathrm{T}}\mathbf{W}\mathbf{J} + \sum_i (\Delta\mathbf{z}^{\mathrm{T}}\mathbf{W})_i \frac{\mathrm{d}^2\mathbf{z}_\mathrm{i}}{\mathrm{d}\mathbf{x}^2} \tag{34}$$

For a complex cost function the evaluation of the Hessian $\mathbf{H}$ using equation 34 is difficult and instead it is sometimes calculated from the change in $\mathbf{g}$ across iterations. If the prediction model is linear or $\Delta\mathbf{z}(\mathbf{x})$ is small, $\frac{\mathrm{d}^2\mathbf{z}_\mathrm{i}}{\mathrm{d}\mathbf{x}^2} \approx 0$ and using equation 31, the Gauss-Newton approximation for the least squares problem is given by:

$$(\mathbf{J}^{\mathrm{T}}\mathbf{W}\mathbf{J})\delta\mathbf{x} = -\mathbf{J}^{\mathrm{T}}\mathbf{W}\Delta\mathbf{z} \tag{35}$$

With this approximation the step-size $\delta\mathbf{x}$ can be found by solving the following least squares problem:

$$\min_{\delta\mathbf{x}} \frac{1}{2}(\mathbf{J}\delta\mathbf{x} - \Delta\mathbf{z})^{\mathrm{T}}\mathbf{W}(\mathbf{J}\delta\mathbf{x} - \Delta\mathbf{z}) \tag{36}$$

To aid convergence a line search algorithm can be employed to refine the Newton step size by finding the value α which minimizes the cost function along the line from $\mathbf{x}$ to $\mathbf{x} + \alpha\,\delta\mathbf{x}$. For a problem of size n Newton's method takes a time of order $\mathcal{O}(n^3)$. Hence for large problems it will often be necessary to exploit the sparsity of matrix $\mathbf{H}$ or consider the use of simpler first order methods.

The bundle adjustment process can be subject to constraints due to various factors:

- Constraints on elements of the scene (e.g. on camera or keypoint parameters).
- Implicit, rather than explicit, observation models of the form $\mathbf{h}(\mathbf{x}, \mathbf{z}) = \mathbf{0}$ (e.g. modelling the observation of noisy points on a 3D curve).
- Local parameterizations required because of non-linear behaviour (e.g. from singularities in affine point coordinates at infinity or unwanted degrees of freedom).

The method of Lagrange multipliers can be used to perform a constrained cost optimization. The minimum value of a cost function $f(\mathbf{x})$ subject to constraints $\mathbf{c}(\mathbf{x}) = \mathbf{0}$, with Lagrange multipliers in vector $\boldsymbol{\lambda}$, will occur at a saddle point of the Lagrangian function $f + \mathbf{c}^{\mathrm{T}}\boldsymbol{\lambda}$. The Newton step size $\delta\mathbf{x}$ is determined by the following equations:

$$\mathbf{0} = \frac{\mathrm{d}}{\mathrm{d}\mathbf{x}}(f + \mathbf{c}^{\mathrm{T}}\boldsymbol{\lambda})(\mathbf{x} + \delta\mathbf{x}) \approx \mathbf{g} + \mathbf{H}\delta\mathbf{x} + \mathbf{C}^{\mathrm{T}}\boldsymbol{\lambda} \tag{37}$$

$$\mathbf{0} = \mathbf{c}(\mathbf{x} + \delta\mathbf{x}) \approx \mathbf{c}(\mathbf{x}) + \mathbf{C}\delta\mathbf{x}\,, \quad \mathbf{C} = \frac{\mathrm{d}\mathbf{c}}{\mathrm{d}\mathbf{x}} \tag{38}$$

Determination of Camera Parameters using Homographies between Views

For the 'Bundle Adjustment' process to converge efficiently to the global minimum of the cost function (rather than a local minimum) it is usually necessary to have good estimates for the unknown parameters to use as initial values in the algorithm. In many cases it is not possible to accurately estimate parameters and a more efficient solution is to use an homography between a pair of views (or a trifocal tensor defining the relationship between three views) to calculate camera parameters and key-point coordinates. The values found can then be further refined using bundle adjustment. Further homographies between computed views and other views can be used incrementally to calculate camera parameters and key-point coordinates for new views, again using bundle adjustment at each stage to refine the solution.

Figure 25 shows the epipolar geometry which gives the relationship between the projected position of a point X for the two views corresponding to cameras $\mathbf{c}$ and $\mathbf{c}'$ [25]. The epipoles $\mathbf{e}$ and $\mathbf{e}'$ occur at the intersection of the baseline joining the camera centres with the image planes. The epipole for one view occurs at the position in the image corresponding to the camera centre of the other view. Given the known projected position $\mathbf{x}$ in one view, the projected position in the other view $\mathbf{x}'$ must lie on the epipolar line $\mathbf{l}'$. The relationship between the projected image points in the two views can also be expressed using the 3x3 **Fundamental Matrix F**:

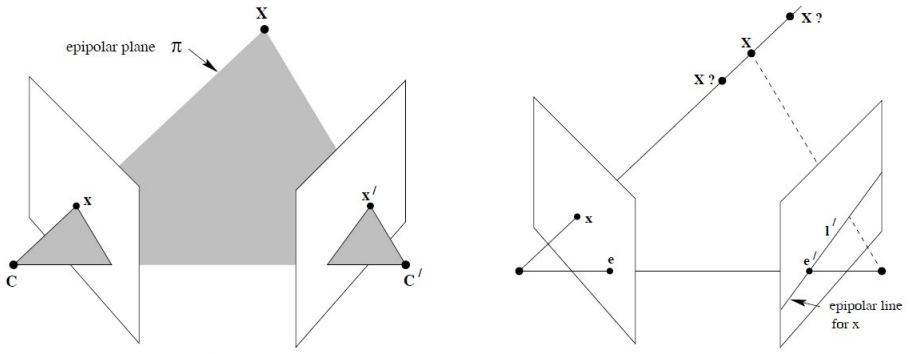


Figure 25: Epiploar Geometry showing Correspondence between Projected Positions for a Key-point in Two Views [25]. Key-point X is projected to positions $\mathbf{x}$ and $\mathbf{x}'$ for cameras $\mathbf{c}$ and $\mathbf{c}'$ respectively (left). Given a projection position $\mathbf{x}$ in one view, the projected position of the key-point in the other view must lie on the epipolar line $\mathbf{l}'$ (right).

$$\mathbf{x}'\mathbf{F}\mathbf{x} = 0 \tag{39}$$

Matrix $\mathbf{F}$ is of rank 2 and hence there is no unique solution of the above equation to find $\mathbf{x}'$ given $\mathbf{x}$ or vice-versa.

Hartley and Zisserman give an outline of a reconstruction method using two views [25]. Firstly, the coordinates of point correspondences between the two views, $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, are substituted into Equation 39 to give a number of linear equations. With eight correspondences these equations can easily be solved to yield the fundamental matrix $\mathbf{F}$ (up to a scaling factor). If there are more than eight correspondences a least squares method can be used to find the $\mathbf{F}$ that minimizes the residual terms $\mathbf{x}'_i\mathbf{F}\mathbf{x}_i$. The camera matrices for the two views, $\mathbf{P}$ and $\mathbf{P}'$ can then be calculated using the following equations:

$$\mathbf{P} = \left[\begin{array}{c|c} \mathbf{I} & \mathbf{0} \end{array} \right] \tag{40}$$

$$\mathbf{P}' = \left[\begin{array}{c|c} \mathbf{e}' \times \mathbf{F} & \mathbf{e}' \end{array} \right] \tag{41}$$

In the above equation $\mathbf{e}'$ represents the epipole position for the second view which is given by $\mathbf{e}'^{\mathrm{T}}\mathbf{F} = 0$.

After the determination of the camera matrices for the two views, key-point positions can be calculated through triangulation. Two rays projected from the camera centre of each of the views to the position of a key-point on the corresponding image (i.e. from $\mathbf{c}$ to $\mathbf{x}$ and from $\mathbf{c}'$ to $\mathbf{x}'$ in Figure 25) will lie in a plane (an epipolar plane containing the key-point and camera

centres) and intersect at the 3D location of the key-point.

After triangulation the final phase of the reconstruction process is to rectify the camera matrices and key-points to the correct orientation and scaling. If there are at least five control key-points with known positions $\mathbf{X}_{\mathrm{E}i}$ which have been reconstructed to positions $\mathbf{X}_i$, then the homography $\mathbf{H}$ required for $\mathbf{X}_{\mathrm{E}i} = \mathbf{H}\mathbf{X}_i$ can be calculated. The metric reconstruction is then given by:

$$\mathbf{P}_{\mathrm{M}} = \mathbf{P}\mathbf{H}^{-1}\,, \quad \mathbf{P}'_{\mathrm{M}} = \mathbf{P}'\mathbf{H}^{-1}\,, \quad \mathbf{X}_{\mathrm{M}i} = \mathbf{H}\mathbf{X}_i \tag{42}$$

If there are no known key-point positions, a suitable homography $\mathbf{H}$ can be found from the image by calculating the plane at infinity or the image of the absolute conic [25].

View Clustering

The efficiency of the 'Structure from Motion' pipeline can be greatly improved by reducing the reconstruction problem into a series of sub-problems that are better conditioned and more easily optimized, with a resultant lowering of the computational effort required for bundle adjustment.

Gherardi *et al.* [49] proposed a parallelizable method in which images are organized into a hierarchical binary cluster tree (dendrogram), with the technique reducing computation times by an order of magnitude (from $\mathcal{O}(n^5)$ to $\mathcal{O}(n^4)$) for well balanced trees. The grouping of views in the tree is determined by an affinity matrix, with the entry $a_{i,j}$ giving an affinity for the pair of images i and j. High affinity values will assigned to image pairs with more key-points in common and for pairs with key-points evenly spread over the images. The matrix element $a_{i,j}$ is given by:

$$a_{i,j} = \frac{1}{2}\frac{\left|S_i \cap S_j\right|}{\left|S_i \cup S_j\right|} + \frac{1}{2}\frac{CH(S_i) + CH(S_j)}{A_i + A_j} \tag{43}$$

In the above equation, for image i, S_i is the set of matching key-points in the image, $CH(S_i)$ is the area of the convex hull containing the set of points S_i and A_i is the total area of the image.

Nodes of the tree are repeatedly merged to form partial reconstructions until there is a single node containing the complete reconstruction. A pair of views (at the leaves of the tree) can be merged to form a view cluster at a new node using a standard two views reconstruction method. Clusters can be merged by using common key-points to calculate the homography required to register one cluster with the other. These common points are then recalculated using triangulation and further refinements are made using bundle adjustment. A single view can also be added to a cluster using the Direct Linear Transformation (DLT) algorithm [25] followed by refinement using triangulation and bundle adjustment.

Computational complexity is reduced for balanced trees and to achieve this balance the pair of views or clusters to merge at each stage was chosen by selecting the two with the lowest total cardinality (i.e. number of encapsulated views) from the ℓ pairs of clusters or views having the nearest neighbours in descriptor space (i.e. having the closest feature match). The authors found that setting $\ell = 5$ gave the best results.

5.4.2 Evaluation of Photogrammetry Software

Images for 3D environment reconstruction using photogrammetry can be captured through aerial filming (e.g. from a drone or helicopter). Such filming usually entails detailed preparation to fulfil pre-flight requirements such as obtaining licenses or permissions and planning to ensure flight safety. The actual filming is also time consuming and expensive. For certain (typically urban) locations a quicker and cheaper option is to generate the images from sources such as Google Earth. Although these images will be of lower quality than actual photographs (e.g. containing computer artefacts), they can be used to generate models of sufficient accuracy for flight planning and training applications.

There are many software packages available for the reconstruction of 3D models from 2D images and various systems (including Autodesk ReCap and 3DF Zephyr) were evaluated for use in producing models for the MultiDrone flight planning and training program. The main requirements for MultiDrone were the ability to produce a good quality model using images captured from Google Earth and compatibility of the output 3D model format with the simulation engine (Unreal Engine). Images for the reconstructions were created by using a screen capture program (ScreenToGif) to automatically generate screen shots at regular, short intervals (e.g. 0.25s) during navigation around a location in Google Earth. Figure 26 shows examples of these manually scanned environment images which were captured for the location of a roundabout in Bristol. The images were produced whilst orbiting the target at a series of heights so that each area of the environment was captured from a number of different angles. It was found that Autodesk ReCap produced poor quality models from Google Earth generated images and the software also required 'cloud credits' to perform analyses remotely on Autodesk servers. The 3DF Zephyr Aerial Photogrammetry software [51] was able to generate higher quality 3D models but for analyses using a large number of 2D images the execution time was highly dependent on the available local GPU processing power. Figure 27 shows the 3D model of the roundabout reconstructed from the Google Earth images using 3DF Zephyr software.

As a result of the evaluation process the 3DF Zephyr Aerial photogrammetry software was chosen for the initial research on the production of environment models for the MultiDrone flight-planning and training system.

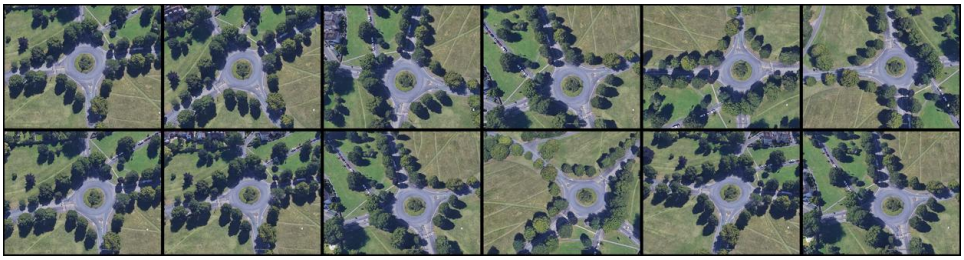


Figure 26: Manually Scanned Environment Images of a Roundabout Captured from Google Earth.

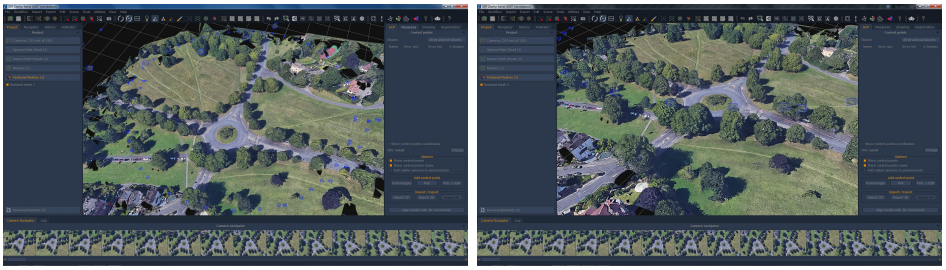


Figure 27: A 3D Model of a Roundabout Reconstructed using 3DF Zephyr Aerial Photogrammetry Software.

5.4.3 3DF Zephyr Workflow

The stages in a 3DF Zephyr workflow are as follows [51]:

1. **Structure from Motion**: Calculates camera position and orientation for each image and generates an initial 3D sparse point cloud.
2. **Multi-View Stereo**: Calculates the coordinates of surface points forming a Dense Point Cloud using the camera information obtained in phase 1.
3. **Surface reconstruction**: Creates a surface mesh from the dense point cloud generated in phase 2.
4. **Textured Mesh generation**: Applies texture to the surface mesh geometry calculated in phase 3.

Parameters for each stage of the workflow can be set by selecting the Category for the type of image capture used and a preset to select the required quality. Types of Category include:

- General: Works in most cases.
- Close Range: Used for small objects or close-up details.

- Aerial: Used with images from a top down view.
- Urban: Used for images of an urban environment or images from different cameras or with different focal lengths.
- Human: Optimized for full or part of a human body.

Example preset configurations for dense point cloud creation are Fast, Default or High Details. Parameters can also be set individually using the Advanced option. Good results can usually be obtained by selecting the appropriate category (e.g. General) and then using preset configurations of Deep for the camera orientation stage and High Details for both the dense point cloud creation and the surface reconstruction stages. However the reconstruction quality and analysis time are highly dependent on the particular parameters used and to optimize the reconstruction process it is necessary to manually configure the parameters at each stage. For example, camera orientation using the Deep preset could sometimes result in a large number of incorrect camera positions (see Figure 28). It was found that setting the Keypoint Density parameter to 'High', Matching Type to 'Accurate' and Matching Stage Depth to 'High' gave good results in most cases, although the processing time was increased compared to that of an analysis using the standard Deep preset. For optimum results in the dense point cloud generation stage it was also found that, in the Stereo Settings Page, the Output type should be set to 'Refined' and Speed up level set to 'Low'.

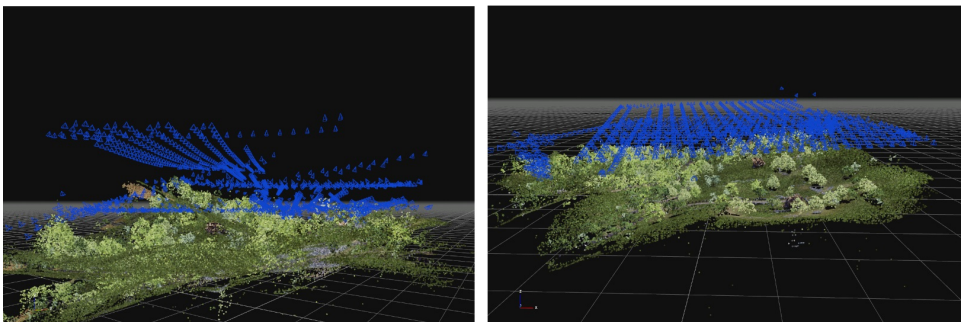


Figure 28: Improving the Camera Alignment in 3DF Zephyr. Original sparse point cloud with poor camera alignment (left), improved using 'Accurate' Matching Type (right).

Rather than use the automated Zephyr work-flow in which the camera orientation, dense point cloud creation, surface reconstruction and texturing phases are automatically run one after another, it can be beneficial to run each stage manually. If a particular stage yields poor results it can be rerun with new parameters without having to wait for subsequent stages to finish.

After the successful completion of each stage the resultant model (e.g. dense point cloud or surface geometry) can be edited to remove areas with distortion or areas which are not required. This technique can be used to ensure only the area of interest is included in the final mesh, and not distant areas on the edge of the reconstruction (which typically have large distortions). Hence the final textured mesh is reduced in size, sometimes significantly, and in many cases the model can be exported to a file with a single texture. For large analyses the total processing time can also be significantly reduced by eliminating unwanted geometry at each stage.

The final textured mesh model produced in 3DF Zephyr was exported as an FBX file which could then be imported into Unreal Engine. It was found that generally, for models with the same mesh density, smaller models having a single texture imported into Unreal Engine with better quality than models extending over a greater area which required multiple textures.

Some of the main conclusions reached regarding the use of photogrammetry generally and 3DF Zephyr specifically to produce environment models in Unreal Engine are summarized below:

- Increasing the number of images improves the quality of the final 3D model up to a certain point, after which an increase will have little noticeable effect.
- The analysis time increases rapidly with the number of images. An analysis of 330 images using a PC with an Intel i7-6700 CPU @ 3.40Ghz, NVIDIA Quadro K620 GPU and 16GB RAM took approximately 2 hours to process whilst an analysis with 720 images took over 11 hours (using 'Deep' and 'High Details' presets).
- Careful choice of image viewpoints will give better results (e.g. if a building has a courtyard orbits should be made inside the courtyard in addition to around the main building).
- Using the 'highest quality' setting for some 3DF Zephyr parameters may have little effect on reconstruction quality but can greatly increase processing time. For example in the camera orientation phase, using a Keypoint Density of 'Very High' and Matching Stage Depth of 'Full' (instead of using 'High' for both) can double the processing time for little discernible improvement in quality.
- Models with fewer textures generally have better quality when imported into Unreal Engine. The maximum texture size in Unreal is 8192×8192 and hence large models will need to be imported using multiple textures.

- At the edges of a reconstructed 3D model there will often be large distortions and missing areas due to incomplete view coverage for that part of the scene and because areas distant from the camera will be captured at oblique angles.

5.4.4 Recomendations for Photogrammetry Image Capture

In general when taking images for photogrammetry the area of interest should be kept at the centre of the frame and pictures should be taken from a wide range of viewing positions to ensure adequate coverage of the scene. For reconstructions of buildings images taken from high camera angles will provide information on roof details and images taken lower down at low angles will provide more information on the appearance of the walls. A high overlap between images (e.g. 70% - 80%) should be used and increasing the number of images will generally improve the reconstructed model quality.

In particular, when taking images using a camera the following guidelines should be followed [51]:

- Ensure the lighting is sufficient to prevent dark shadows hiding surfaces (e.g. when using direct lighting). Lighting conditions should remain consistent throughout image capture.
- Avoid high ISO values to reduce noise.
- High aperture values (e.g. f/8 - f/16) should be used to give a deep depth of field.
- Use the same camera and focal length for all images.
- Before photogrammetry unsuitable images should be removed (e.g. blurred and under or over-exposed images).

5.4.5 Common Problems with Models Reconstructed using Photogrammetry

There are a number of common problems that can occur in models reconstructed from photogrammetry, especially when there is an insufficient number of images or the images are of poor quality:

- Thin or small objects such as poles may be incorrectly reconstructed due to an insufficient number of close up images of the objects being captured during an overhead scan. This often results in a reconstruction with isolated floating artefacts (see Figure 64 on page 145).

- Flat surfaces with a uniform texture such as roads, water or grass, may be reconstructed with a bumpy or pitted appearance. This problem is often due to the reconstruction software having difficulty in matching surface points between images.

- Certain areas (especially near the edges of the area captured during a scan) may be highly distorted and irregular. The reconstructed mesh for distorted areas such as these will often contain very small faces or thin faces having one or two very small internal angles.

- The apparent level of distortion in a poorly reconstructed object is highly dependent on the texture applied to the object and viewer expectations. Man-made features or features with a regularly patterned texture can appear more distorted than natural or uniformly textured features having the same amount of distortion. For example, a highly distorted road surface with a uniform texture may appear relatively flat, whilst any distortion on a road surface having road markings will be more obvious.

3D Modelling software can be used to improve the final quality of a model before it is imported into a drone flight simulation environment. Appendix C gives details of procedures in Blender that can be used to repair many common problems found in environment models that have been reconstructed using photogrammetry.

5.4.6 Example Environment Models Created using Photogrammetry from Google Earth Image Data

A number of environment models were created using images from Google Earth and the 3DF Zephyr software. The models were imported into Unreal Engine using the FBX interface. For individual buildings it was found that the best results were obtained through capturing images during orbital scans around the building at a series of height levels. Figure 29 shows a model of the Wills Memorial Building created using this method from approximately 1200 images and requiring a reconstruction time of 3 days. The model has been placed on an Unreal Landscape of the Bristol area generated using SRTM height-map data (see Section 5.3.2). For models extending over a large area, such as that for Filton shown in Figure 30, the best strategy was to capture images from scans up and down the length of the model. Additional scans in a perpendicular direction to the first scan and scans at additional heights each gave improved quality models, with fewer holes due to unreconstructed regions and less distortion in reconstructed areas.

Google Earth provides full 3D modelling of the environment (i.e. 3D models of buildings and terrain) for only a small proportion of the total surface area of the Earth. For the majority of the Earth's surface the terrain

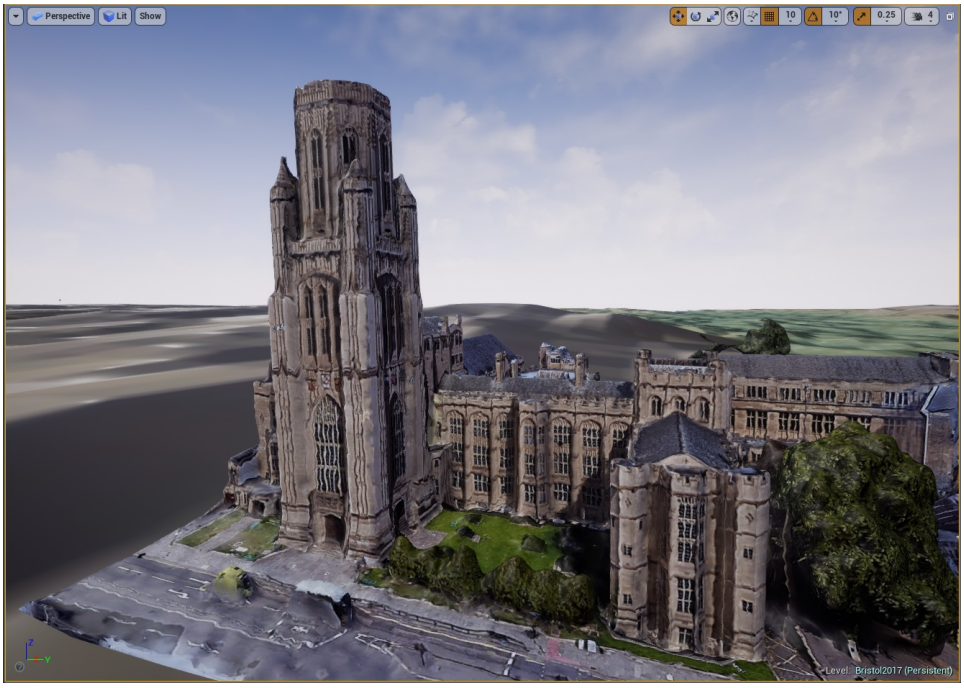


Figure 29: Wills Memorial Building Reconstructed from Google Earth Views using 3DF Zephyr Photogrammetry Software and Imported into Unreal Engine.

Figure 30: Large-scale Model for Area of Filton, Bristol Reconstructed from Google Earth Views using 3DF Zephyr Photogrammetry Software and Imported into Unreal Engine.

height is modelled but the surface texture is mapped from a 2D aerial photo leading to flat surface features (e.g. buildings appear to be 2D). For relatively flat areas photogrammetry will be unable to discriminate the height difference between different areas. For hilly or mountainous areas it should be possible to create a 3D model of the terrain using photogrammetry. Figure 31 on page 79 shows the Google Earth model, which is not reconstructed in full 3D, and the corresponding 3D model produced using 3DF Zephyr, for an area near Lake Serru in Italy. Although the model would be useful for high altitude flight planning (e.g. to avoid flying close to a mountain), it would not be useful for planning low altitude flights (e.g. for building avoidance etc).

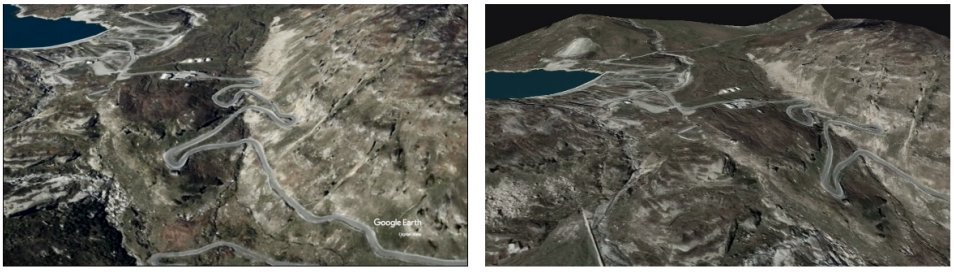


Figure 31: Google Earth and Reconstructed Models for Lake Serru. An area near Lake Serru, Italy in Google Earth (left) and reconstructed in 3DF Zephyr (right).

The low resolution of the Google Earth images for this particular area results in a model in Unreal which although of only slightly lower quality than the Google Earth model, will still have limitations for the design of camera shots (as illustrated in Figure 32).

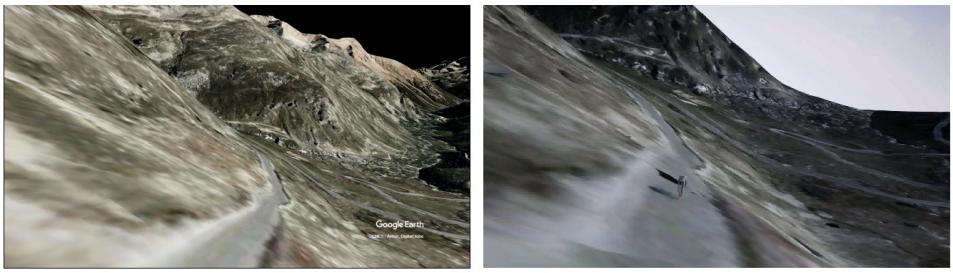


Figure 32: Low Altitude Views of Lake Serru Models. Google Earth model (left) and Unreal model with added cyclist (right).

5.5 Viability of 3D Environment Model Production

The previously discussed methods of producing 3D environment models are complex and require a good level of expertise in the photogrammetry software to optimize the model quality given limits on acceptable reconstruction time. Although it would be possible to automate many of the processes involved (i.e. create Blender 'Add-ons' to execute model repair procedures), in general it cannot be expected that members of a drone crew will have the skills and the time necessary to create 3D models of film locations. The work would probably need to be carried out by computer modelling professionals (with VFX experience for the case of modelling movie action sequences), either in-house or through a consultancy. When filming an event such as a cycling race it is probably the case that it would not be an efficient use of resources to model a rural village which the cyclists pass through quickly. Even for urban areas it may be sufficient to use online resources such as Google Earth Studio for flight planning (especially if the drone operators have previous experience of filming at the location). The modelling of large areas is time consuming because of the large number of reconstructions needed and the problems associated with stitching individual areas together. For these reasons, the integration of custom made 3D environment models into flight planning software may, at present, be only practicable for certain filming scenarios such as the planning of special stages in live shoots (e.g. optimizing shots for good views of both subjects and important landmarks in the background) or the planning of expensive movie action sequences that require a single take.

5.6 A Simulation Environment for Drone Cinematography

A system has been developed using Unreal Engine which allows camera shots, selected from the taxonomy designed for MultiDrone, to be simulated in a realistic environment modelled on the real-world [10]. Predefined environments for Clifton and the Harbourside at Bristol, shown in Figures 33 and 34, were created using data from Google Earth. Environments for other locations can be imported into the system from models created using techniques such as photogrammetry. The simulation package has three modes of operation: Editing, Simulation and Freeplay.

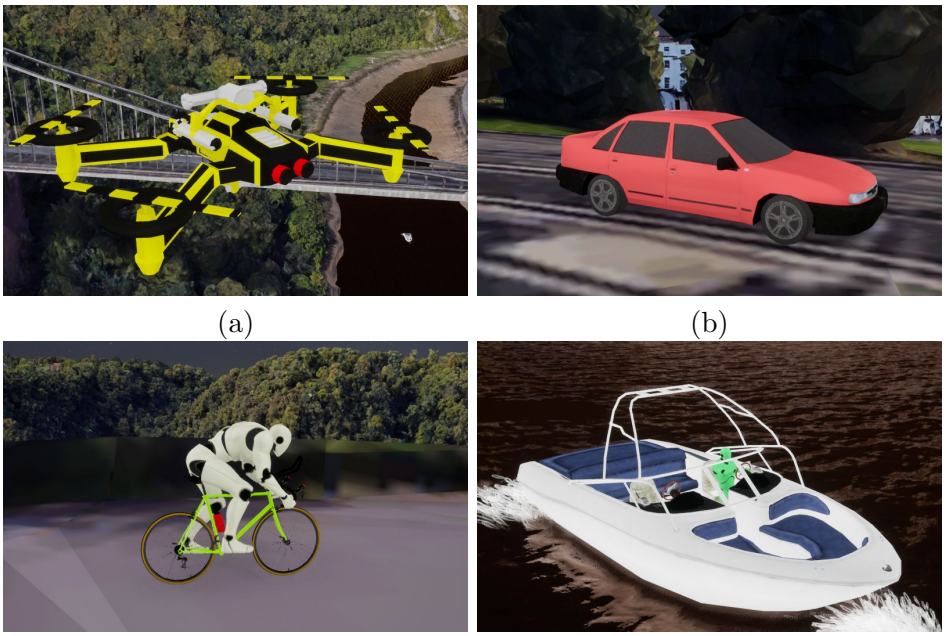
The customized interface of Editing mode allows users to drag and drop models for moving objects (i.e. the predefined models for bicycles, cars, boats and drones, shown in Figure 35 on page 83, or custom objects) into the scene. Path objects (created from a series of way-points) are used to define the trajectory of objects through the scene. General moving objects can be assigned a particular path and a speed. Drone camera objects can be assigned a target object to shoot, a shot-type and a speed as shown in Figure 36 on page 83. The demonstration version of the software had shot types of Establishing, Chase, Flyby, Elevator and Orbit, integrated with default shot parameters determined by the results of the subjective tests described in Section 4.2.3. These parameters were based on a camera with a focal length of 35mm and a sensor size of 23.66mm $\times$ 13.3mm and were automatically recalculated for a different camera using the procedure described in Section 4.3. Menu options are also provided to set environmental parameters defining the time of day for the simulation, the brightness of the Sun, the cloud thickness and the cloud speed.

Figure 33: Pre-defined Environment Modelling Clifton, Bristol [10].

Figure 34: Pre-defined Environment Modelling the Harbour, Bristol [10].

The option interface of Simulation mode is shown in Figure 37 on page 84. When the simulation is started each object will move along it's associated path at it's assigned speed. The user has the option to view the simulation from an external floating camera position or from the perspective of the drone. A view from a second drone can also be displayed in a camera window located at the top left of the screen.

Freeplay mode, shown in Figure 38 on page 84, is similar to Simulation mode except that the user can control the drone camera manually using the keyboard or a game controller, with an option to record the path of the drone so that it can subsequently used in Editing mode (e.g. edited and assigned to an object). When using Freeplay mode there are also options to set parameters defining wind speed and direction.



(a) (b)

(c) (d)

Figure 35: Pre-defined Moving Objects [10]: (a) Drone (b) Car (c) Cyclist (d) Boat.

Figure 36: The Option Interface for a Drone in Editing Mode [10].

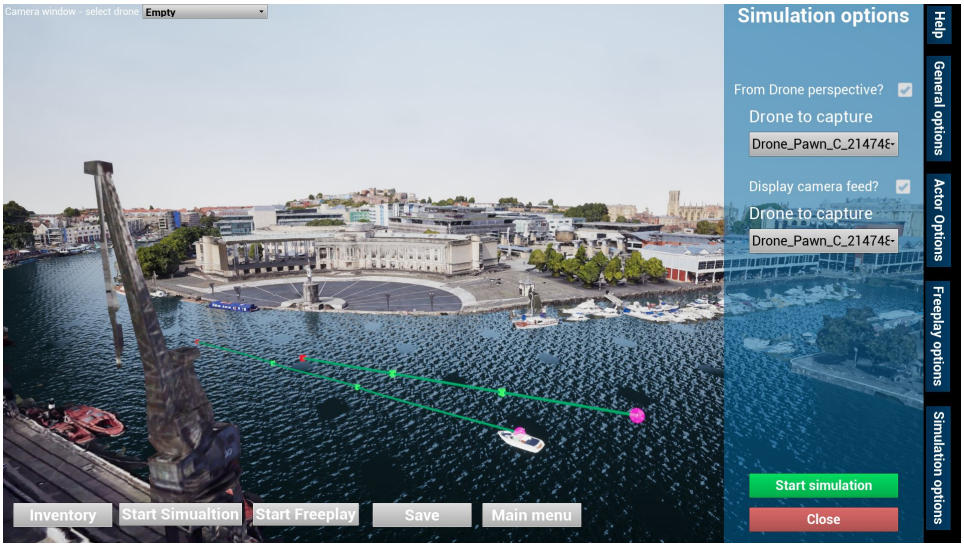


Figure 37: The Option Interface for Simulation Mode [10].

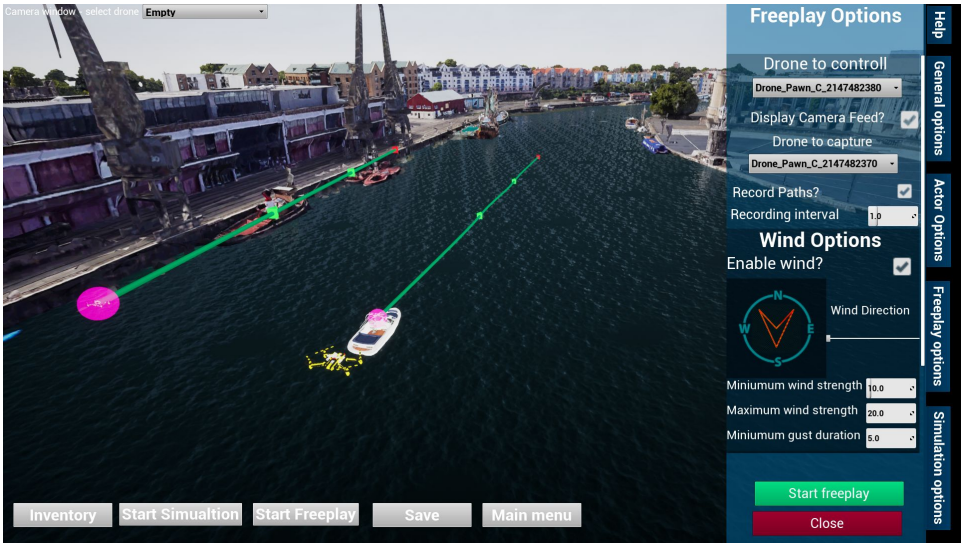


Figure 38: The Option Interface for Freeplay Mode [10].

5.7 Summary

Photogrammetry has been found to be an extremely useful technique for the generation of real world models to be used in applications such as drone flight planning. The quality of reconstruction is dependent on the image set capturing all areas of the scene from a variety of different angles. However the reconstruction time increases rapidly as the number of images increases and hence it is highly advantageous to optimize the image capture process to give the required coverage using the minimum number of images. In addition to a reduction in reconstruction time this can also help to minimize flight times when using a drone for image capture, which will be of great benefit when scanning large areas. The next section examines previous research into the optimization of drone flights for image capture and the development of a system built in Blender that is designed to optimize drone scans for view coverage.

6 Optimization of Scanning Flight Paths for Photogrammetry

Some of this work has previously been published in the following paper:

S. Boyle, M. Newton, F. Zhang, *et al.*, "Environment Capture and Simulation for UAV Cinematograhy Planning and Training," in *European Signal Processing Conference*, *Satellite Workshop: Signal Processing, Computer Vision and Deep Learning for Autonomous Systems*, 2019. [Online]. Available: https://research-information.bris.ac.uk/ws/portalfiles/portal/199953761/Environment_Capture_and_Simulation_for_UAV_Cinematography_Planning_and_Training.pdf

The quality of a model generated using photogrammetry will be dependent on a number of factors including the number of the images, the image resolution and the range of different views for each surface patch. In general, using more images from different viewpoints will give better results, but as reconstruction time typically quadruples for a doubling in image number, it is important (especially for reconstructions of large areas) that the views captured are optimized to give the best possible reconstruction for a given number of images.

6.1 Related Work on the Optimization of Drone Flights for Photogrammetry

The use of drones to capture images for the photogrammetric reconstruction of environments has become widespread and the optimization of the drone trajectories used for scanning has become an increasingly important area for research.

Path planning algorithms for covering a known environment have been described by Galceran *et al.* [52]. Algorithms to optimize the exploration of unknown areas using robots or aerial vehicles have been developed using strategies such as the identification of the boundaries between known and unknown areas [53], the maximization of newly visible areas [54] and the maximization of information gain [55]. Wang *et al.* [56] give an optimization procedure for scanning an area with a robot which minimizes a cost due to the total travelling distance and the number of points visited. Although these algorithms optimize for scene coverage they are not optimized for photogrammetry (i.e. multi-view stereo), which requires coverage with a wide range of viewing angles for every location in the scene.

Various researchers have examined the problem of selecting an optimum set of images for photogrammetry from an image data-set. Mauro *et al.* [57] minimized dense point cloud generation time by using the minimum number of points from the sparse cloud (generated by the 'Structure from Motion'

phase) necessary to give a good 3D reconstruction. The importance value of a sparse point and it's associated image was calculated from the sum (over all points contained in the image) of a point aggregate energy. The aggreagate energy for each point was determined by the weighted sum of individual energy components which depended on the feature density, the uncertainty in the position of the point, the 2D saliency (i.e. how much the point contributed to the image) and the 3D saliency (a measure of the complexity around the point). Hornung *et al.* [58] proposed an algorithm for image selection which improved 3D reconstruction (when compared with uniformly distributed views). A small subset of the image data-set was used to create an initial stereo-based surface proxy. Images were then added in two phases to improve the proxy, firstly to ensure coverage of each point on the proxy surface in at least two images and then to improve critical regions such as cavities, which were detected using estimates of the local photo-consistency. Schmid *et al.* [59] created an initial 2.5D Digital Surface Model (DSM) from images captured at viewpoints positioned on a regularly spaced grid, with the drone at a constant height and the camera pointing vertically down. A spherical view hull was created at a minimum distance of d to the DSM with all edges smoothed to a radius of d. A set of regularly spaced candidate viewpoints were generated over the hull with view directions given by the hull surface normal. An optimization algorithm was then used to generate a subset of views from the candidate list that provided complete coverage subject to the following constraints:

- Reconstructed points should be visible from at least two views.
- For every point there are two views containing that point that have view directions differing by less than a maximum value β_{max}.
- Each view must have a degree of overlap with all neighbouring views.

The optimization of drone flight paths for multi-view stereo reconstruction has become an important area for research. The most common strategy has been to use a two stage process in which data from an initial exploration flight is used to produce a coarse model of the scene, which is then used to generate an optimized flight path for capturing images which are conducive to good photogrammetric reconstruction.

Roberts *et al.* [60] proposed an automated method for drone trajectory planning that uses this two stage approach, with images from a flight having a default trajectory used to build an initial model. A bounding box spanning the model is then sampled to produce nodes representing possible view (camera) locations. The optimization procedure has two stages, exploiting the submodularity of the problem to reduce the computational overhead. In the first stage an optimal viewing direction is found for every node by maximizing the quality for a reconstruction involving one viewpoint from every

node. In the second stage of optimization a path, formed from a subset of nodes, is found which optimizes coverage (using the view direction at each node determined in the first stage), subject to a constraint on the maximum path length. The quality of reconstruction for each surface point $\mathbf{s}_j$ on the initial model is estimated by considering a hemisphere $\mathbf{H}_j$ of radius 1m centred at the point and orientated so that the pole is aligned with the surface normal. For each camera viewpoint $\mathbf{c}_i$, a disk $\mathbf{D}_i^j$ is formed on the surface of hemisphere $\mathbf{H}_j$, centred at the projection of $\mathbf{c}_i$ onto the hemisphere, as shown in Figure 39.

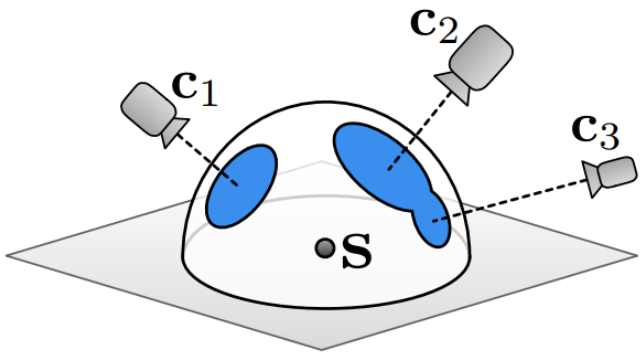


Figure 39: View Coverage for a Vertex. Disks corresponding to three camera viewpoints projected onto the surface of the hemisphere surrounding point S [60].

The size of $\mathbf{D}_i^j$ is defined by the half angle θ_i^j of a cone with a apex at $\mathbf{s}_j$ and base $\mathbf{D}_i^j$, given by:

$$\theta_i^j = \theta_{max} 2^{\frac{-max(t_i^j - t_0, 0)}{t_{half}}} \tag{44}$$

In the above equation t_i^j is the distance from camera $\mathbf{c}_i$ to the surface point $\mathbf{s}_j$, $\theta_{max} = \frac{\pi}{8}$rad and the values of the other variables are chosen to give an appropriate falloff with distance t_i^j. A weighting function $w_j(\mathbf{h})$ is also defined by:

$$w_j(\mathbf{h}) = cos\, \alpha_{\mathbf{h}} \tag{45}$$

In the above equation $\mathbf{h}$ is the vector from the hemisphere origin to a location on the hemisphere surface and $\alpha_{\mathbf{h}}$ is the angle from $\mathbf{h}$ to the hemisphere pole.

The integral of the weighting function over areas of the hemisphere $\mathbf{H}_j$ covered by disks $\mathbf{D}_i^j$ gives a measure of how well the surface point $\mathbf{s}_j$ is viewed

from a wide variety of angles, giving preference to close-up views (which will produce disks with a large area, see equation 44) and non-oblique views (i.e. with small values of $\alpha_{\mathbf{h}}$). The total coverage for the scene (over $J+1$ surface points j) is given by:

$$f(C) = \sum_{j=0}^{J} \int_{V_j} w_j(\mathbf{h})\, d\mathbf{h} \tag{46}$$

In the above equation, for each hemisphere j, the integral is over that part of the surface V_j that is covered by a disk. The integral can be evaluated by sampling K equally spaced points over the surface of each hemisphere and using a coverage indicator function $v_j(\mathbf{h}_k)$ that returns a value of 1 if the hemisphere location $\mathbf{h}_k$ is covered and 0 if it is not covered:

$$F(C) = \frac{2\pi}{K} \sum_{j=0}^{J} \sum_{k=1}^{K} w_j(\mathbf{h}_k)\, v_j(\mathbf{h}_k) \tag{47}$$

Equation 47 can be written in matrix form using a stacked vector of weights $\mathbf{w}$ which contains entries for the weighting at every hemisphere location $\mathbf{h}_k$ for every point j, and a similarly formed stacked coverage indicator vector $\mathbf{v}$:

$$F(C) = \mathbf{w}^T \mathbf{v} \tag{48}$$

For the optimization process it was necessary to evaluate the coverage $F(C_{\mathbf{P}})$ provided by a subset of cameras and this was determined using coverage indicator vectors $\mathbf{v}_i$ for each individual camera $\mathbf{c}_i$. An entry in the coverage indicator vector $\mathbf{v}_{\mathbf{P}}$ for a subset of cameras will be a 1 if any of the corresponding entries in the individual coverage indicator vectors $\mathbf{v}_i$, for cameras in the subset, are 1, otherwise it will be a 0. The drone path $\mathbf{P}$ was then optimized for maximum coverage subject to constraints of a maximum path length B and a fixed start and end point $\mathbf{p}_{\text{root}}$, with optimal path $\mathbf{P}^*$ given by:

$$\mathbf{P}^* = \arg\max_{\mathbf{P}} F(C_{\mathbf{P}}) \quad : \quad l(\mathbf{P}) \leq B \quad \mathbf{p}_0 = \mathbf{p}_{\text{q}} = \mathbf{p}_{\text{root}} \tag{49}$$

Results from simulations using Unreal Engine with the 'Grass Lands' environment model showed the method gave an improved reward (evaluated using a quantitative measure of model quality) for a given travel budget, when compared to other path selection algorithms such as 'Next-Best-View' and 'p-SPIEL Orienteering'.

An optimization method for urban scene reconstruction was described by Smith *et al.* [61]. A reconstructability heuristic was used to ensure camera

positions and angles were suitable for accurate model reconstruction, based on principles that apply generally to MVS (Multi-View-Stereo) algorithms:

- Triangulation error increases with distance and decreasing parallax.
- Triangulation matchability decreases with shallower observation angles.
- For a surface to be reconstructed it needs to be seen in at least two views.

The pairwise contribution of two views, V_1 and V_2 to a sample point s was modelled as:

$$c(s, V_1, V_2) = w_1(\alpha)w_2(d)w_3(\alpha)\, cos\theta \tag{50}$$

In the above equation θ is the shallower observation angle (between view direction and surface normal) and d the maximum view distance, of the two views. The parallax angle between the views is given by α.

The function $w_1(\alpha)$ models the dependency on parallax triangulation error:

$$w_1(\alpha) = (1 + \exp(-\mathrm{k}_1(\alpha - \alpha_1)))^{-1} \tag{51}$$

The function $w_2(d)$ models the dependency on distance:

$$w_2(d) = 1 - \min(d/d_{max}\ , 1) \tag{52}$$

The function $w_3(\alpha)$ models the dependency on surface matchability:

$$w_3(\alpha) = 1 - (1 + \exp(-\mathrm{k}_3(\alpha - \alpha_3)))^{-1} \tag{53}$$

The parameters k_1, α_1, k_3 and α_3 in the above equations were determined using experimental tests. The model gave a maximum for the reconstructability of two views with a parallax angle of approximately 20° (corresponding to a maximum for $w_1 \cdot w_2$). A heuristic for the reconstruction quality of a point s with a set of views $\boldsymbol{\mathcal{V}}$ is given by:

$$h(s, \boldsymbol{\mathcal{V}}) = \sum_{\substack{i=1\ldots|\boldsymbol{\mathcal{V}}| \\ j=i\ldots|\boldsymbol{\mathcal{V}}|}} v(s, V_i)\, v(s, V_j)\, c(s, V_i, V_j) \tag{54}$$

In the above equation $v(s, V)$ is a visibility function which determines if point s is visible in view V.

A geometric proxy for the scene was created using images of nadir viewpoints captured from an initial flight at a high altitude of 100m. The proxy was repaired to remove outliers and holes (e.g. on vertical surfaces such as the walls of buildings) and a set of uniformly distributed sampling points $\boldsymbol{\mathcal{S}}$

created over this new surface. An initial camera network was then created, uniformly distributed over the proxy. The reconstruction of points $\mathcal{S}$ was optimized by minimizing an objective function $\mathcal{O}$:

$$\arg\min_{\mathcal{V}} \mathcal{O} = \arg\min_{\mathcal{V}} \lambda|\mathcal{U}| + \sum_{s \in \mathcal{S} \setminus \mathcal{U}} (\max(h_{\max} - h(s, \mathcal{V}), 0))^2 \qquad (55)$$

In the above equation $\mathcal{U} \subset \mathcal{S}$ is the set of points s that are not visible in any view. The first term encourages a minimum number of unseen points whilst the second term maximizes the reconstruction quality of points. The threshold value $h_{\max}$ ensures that the estimated reconstruction quality for a point does not become increasingly large as the number of views containing that point is increased (and hence $h(s, \mathcal{V})$ increases), since there are diminishing returns with increasing number of views. A downhill simplex method was used to optimize view positions and directions in parallel. After viewpoint optimization, a travelling salesman algorithm was used to find a suitable short path through the view positions, which was smoothed to generate a B-spline curve representing the drone trajectory.

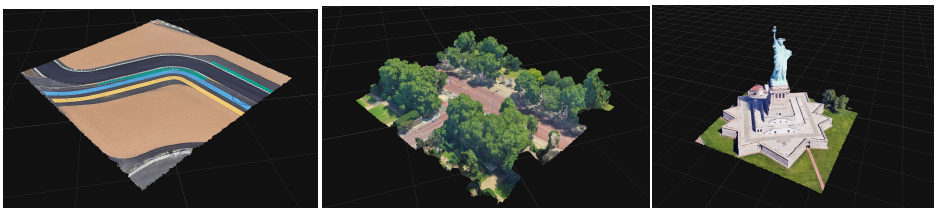
In addition to using actual flights with a drone to validate the results qualitatively, flights simulated in Unreal Engine, with environments modelled using buildings from *GOTH-1* and *CA-1* datasets, were used to generate images for reconstructions that could be analysed quantitatively. Results from the simulations showed that when compared against the sub-modular approach of Roberts *et al.* and Next-Best-View techniques, the method gave improvements of up to 21% in model completeness and lower values for the average depth error. Fewer images were required to achieve a given level of reconstruction quality, although this was achieved through a greater flexibility in the choice of optimum camera position, which sometimes resulted in longer flight paths in comparison to the other methods. It also gave a significant improvement in computational performance and scalability, which was attributed to the parallel implementation and the approach of optimizing the trajectory and orientation in one step. Field tests of the method showed that when compared with a nadir flight it reduced the time necessary to capture complex details of buildings and improved the reconstruction of vertical surfaces.

6.2 Variation of Model Reconstruction Quality with Image Number

Newton *et al.* [9] conducted a series of experiments to determine a baseline for the minimum number of images needed to give good photogrammetry reconstructions. Three environments having an equal area of 10000m^2 (100m×100m) but with varying complexity were used as shown in Figure 40. The section of the Le Mans racetrack had few features, the Statue of

Liberty had more complexity whilst the section of the Mall in London contained many small features and lots of foliage. Images of each area were captured from Google Earth and reconstructions created from 15, 30, 60, 120, 180 and 240 images using 3DF Zephyr software. Fly through videos for each reconstruction were created along with benchmark videos for the original models (created using Google Earth Studio).

In a subjective study, using a double stimulus continuous quality scale (DSCQS) methodology, 12 subjects were asked to rate each reconstruction video and its corresponding reference (each shown twice) on a continuous scale from 1 to 5 (1=Bad, 2=Poor, 3=Fair, 4=Good and 5=Excellent). The order in which the reconstruction and reference were shown was randomly selected, as was the overall order of test sequences. For each reconstruction video a Difference Mean Opinion Score (DMOS) was calculated by finding the mean of the difference scores (the difference between the reference video and reconstruction video scores) using data from all trials.



(a) Le Mans (b) London (c) New York

Figure 40: Subjective Testing to Determine the Effect of Image Number on Reconstruction Quality. Screen shots of reconstructions from each of the three test scenarios [9].

.

The results from the study, given in Figure 41, show that for a fixed number of images used with a reconstruction, the subjective quality is highly dependent on the scene complexity. The complex model of The Mall in London has higher average DMOS scores (corresponding to a lower quality) for all numbers of input images and there is a fall off in quality at low image numbers due to regions of the surface being visible in few or no images (e.g. being obscured by foliage). For the Statue of Liberty scenario the DMOS score increased rapidly as image numbers were reduced below 120. Even though some areas still had acceptable reconstruction with these low image numbers, regions of interest with high detail (such as the crown and torch) were poorly reconstructed and this had a large impact on the overall assessment of quality. In the Le Mans model low DMOS values were obtained until the image number was reduced to 60 or below, in which case the reconstructed surface became bumpy, probably due to the reconstruction software being unable to accurately determine depth values.

It was concluded that to achieve a good reconstruction quality in object

(a) Le Mans (b) London (c) New York

Figure 41: Results of the Experiment to Determine the Effect of Image Number on Reconstruction Quality. The error bar represents the 95% confidence interval [9].

based scenarios, such as the Statue of Liberty, an image number in excess of 180 per 10000m^2 is needed. For background based scenarios, such as Le Mans, an image number in excess of 120 per 10000m^2 was recommended.

6.3 Variation of Model Reconstruction Quality with Scanning Overlap Parameters

A widely used method of obtaining environmental images for photogrammetry is to capture them from a drone flying over the area in two orthogonal directions, using a rectangular or grid-like scanning pattern, as shown in Figure 42.

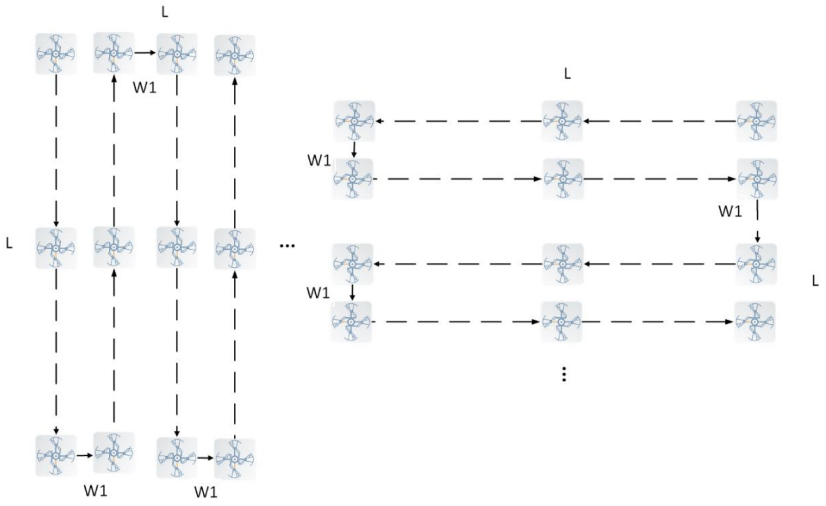


Figure 42: Orthogonal Rectangular Grid Scanning Patterns.

To ensure complete coverage of the area and provide an adequate number of views of each surface point (to allow accurate depth map calculation) a

large overlap, ideally 70% to 80%, between successive images and scan lines is needed [51]. Xu [62] used subjective testing to evaluate the effect of varying the in-track overlap (see Figure 43). Using a model of the countryside in Unreal Engine (obtained from the *Country Side* package in the Unreal Marketplace) images were generated for scan patterns with in-track overlaps of between 30% and 90%. These images were used to create 3D reconstructions using the 3DF Zephyr software. In a study using a double stimulus continuous quality scale methodology, similar to that described in Section 6.2, 15 participants were asked to rate fly-through videos of the reconstructions and the original model. For each video, scores from 1 to 5 were given for both the quality of the overall shape of the landscape and the resolution of local details (e.g. buildings or walls), from which DMOS values were calculated. Results from the study are shown in Figure 44. It was observed that to achieve a good reconstruction quality, with a DMOS value below 1, the overlap ratio needs to be greater than 70% for the landscape and 80% for detail resolution.

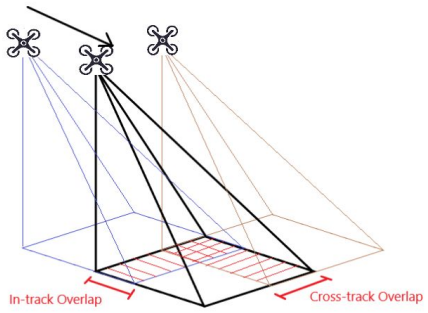
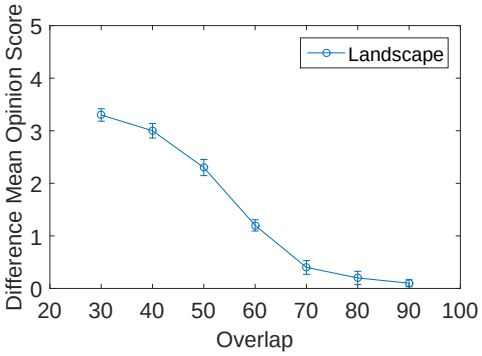


Figure 43: Definitions of In-track and Cross-track Overlap [62].

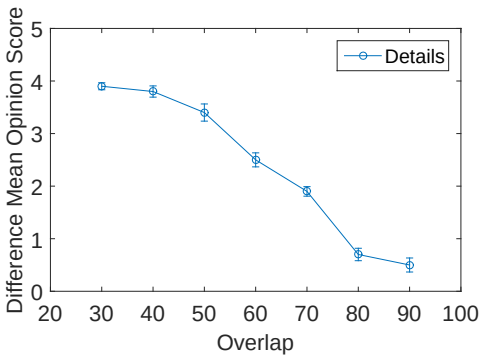
To achieve a particular in-track overlap for a scan it is necessary to ensure that the video frame rate or image capture rate is sufficiently high, with higher rates giving a greater overlap. In addition to the frame rate the amount of overlap is also determined by scan parameters (i.e. the drone height and speed) and the camera focal length and screen size.

To ensure sufficient cross-track overlap (i.e. the overlap between images from successive scan lines) it will be necessary to ensure that the cross-track separation distance (i.e. the distance between scan lines) is at or below a certain value. The required overlap can be achieved by using appropriate parameters for the scan pattern and/or the camera, e.g. using one or more of the following:

- Adjust the distance between scan lines: Reducing the distance will increase the image overlap.



Variation of Landscape Quality.



Variation of Detail Resolution.

Figure 44: Variation of Reconstruction Quality with In-Track Overlap. Variation of DMOS for landscape quality (left) and detail resolution (right) with in-track overlap ratio. [62].

- Adjust the camera focal length: Reducing the focal length will increase the field of view and hence increase image overlap.
- Adjust the drone height: Increasing the drone height will increase the ground area within the field of view and hence increase image overlap.

For a fixed focal length lens having a fixed angular field of view (AFOV), the distance on the ground D_W visible over the width of an image for a drone at height H with camera focal length FL and sensor width size SS_W is given by [63] :

$$D_W = \frac{SS_W \times H}{FL} \tag{56}$$

If a cross-track image overlap O_C (0 for no overlap, 1 for full overlap) is required, the distance between successive scan lines D_S will be given by:

$$D_S = (1 - O_C) \times D_W = \frac{(1 - O_C) \times SS_W \times H}{FL} \tag{57}$$

The Ground Sampling Distance (GSD) for a camera with a sensor having an image width of I_W pixels is given by:

$$GSD = \frac{D_W}{I_W} = \frac{SS_W \times H}{FL \times I_W} \tag{58}$$

To maintain image overlap a trade-off needs to made between the time to scan an area and the image resolution (i.e. the GSD). For large areas, using a low value for the distance between scan lines (D_S) may result in a prohibitively high scanning time. Using a higher value for D_S whilst increasing drone height (H) or reducing the focal length (FL) can maintain overlap,

but this will increase D_W and hence the images will be of lower resolution (i.e. have a larger GSD).

The drone forward travel distance D_F between image captures for a drone with speed v and camera frame rate f is given by:

$$D_F = \frac{v}{f} \tag{59}$$

The distance on the ground, in the direction of drone travel, visible in an image is given by:

$$D_H = \frac{SS_H \times H}{FL \times \sin\theta} \tag{60}$$

In the above equation SS_H is the camera sensor height and θ is the pitch angle (90° corresponding to the camera looking vertically down). Hence the in-track image overlap O_I (0 for no overlap, 1 for full overlap) will be given by:

$$O_I = \frac{D_H - D_F}{D_H} = 1 - \frac{v \times FL \times \sin\theta}{f \times SS_H \times H} \tag{61}$$

The minimum frame rate f_{min} to achieve a required overlap is given by:

$$f_{min} = \frac{v \times FL \times \sin\theta}{(1 - O_I) \times SS_H \times H} \tag{62}$$

Knowledge of the minimum frame rate necessary to achieve a specific level of image overlap can be useful in minimizing the number of images that are used for a reconstruction. To achieve 75% overlap with a drone travelling at 10 m/s at a height of 20m above ground level, using a camera (looking vertically down) having a focal length of 35mm and a sensor size of 23.66mm x 13.3mm, the minimum frame rate is:

$$f_{min} = \frac{10 \times 0.035}{(1 - 0.75) \times 0.0133 \times 20} \approx 5.3 \text{ fps} \tag{63}$$

If the images were actually captured using a video camera with a frame rate of 60 fps, a reasonably good reconstruction could be achieved using only one in ten images from the scan. This could reduce reconstruction times by a significant amount (i.e. by a factor of about 100), since it was found that when using 3DF Zephyr the processing time was approximately proportional to the square of the image number (see Section 5.4.3).

To ensure image overlap requirements are met it is also useful to have knowledge of the maximum drone speed allowable given the camera frame rate:

$$v_{max} = \frac{f \times (1 - O_I) \times SS_H \times H}{FL \times \sin\theta} \tag{64}$$

6.4 Variation of Reconstruction Quality with Camera Angle

Good photogrammetric reconstruction of a surface patch requires a number of images of the patch from differing and not too oblique viewpoints, with each image having an acceptable resolution (taken from close enough to resolve the level of detail required given the GSD). An angle between two view directions of approximately 20° is optimal for calculation of image depth using paralax, with angles greater than 40° giving poor results [64].

It is common practice for scans capturing images for photogrammetry to be performed using fixed scan heights and camera angles (see Foster [65]). Although not optimal, this approach is much simpler than optimized methods and scans can often be performed manually or using simple flight planning tools. It is also quicker than optimal methods, since no initial scan and reconstruction are needed to develop a proxy model. The quality of a model reconstructed from such a scan will be dependent on the particular parameters used. To achieve good results multiple scans, each using a different height and camera angle may be needed. For scanning a building, Foster recommends a series on circular scans around the structure, with a high level scan looking downwards and at least two more (one at high level and others at lower levels) using oblique camera angles to capture side details.

Simulated scans of the Unreal Engine *Country Side* model were used to gain an insight into the magnitude of the variation in model quality with differing values of camera angle pitch. Three reconstructions were performed, each using two orthogonal scans at three different heights (30m, 50m and 70m). The particular angles used at each scan height varied between reconstructions as given below:

- Reconstruction 1, higher angles of pitch: 90° (70m), 67.5° (50m) and 45° (30m).
- Reconstruction 2, intermediate angles of pitch: 85° (70m), 60° (50m) and 35° (30m).
- Reconstruction 3, lower angles of pitch: 70° (70m), 47.5° (50m) and 25° (30m).

Figure 45 shows the two orthogonal scan-line patterns created in the *Matinee* Editor of Unreal Engine which were used to generate 3960 images for photogrammetry reconstructions in 3DF Zephyr. Screenshots of the three reconstructed models are shown in Figures 46 and 47 on pages 99 and 99.

Using high angles of pitch resulted in a model with areas of significant distortion (for the both the shape of the landscape and road) and there were also large holes corresponding to areas which could not be reconstructed. These problems may have been a result of the uniform grass texture over

much of the landscape, which could have made depth calculations inaccurate or impossible for many surface points. 3DF Zephyr could only determine 1668 camera positions from the 3960 images and many of those were inaccurate, as shown in Figure 48(a) on page 100. The reconstruction for areas of high detail (e.g. for buildings) was very good, as shown in Figure 49(a) on page 100.

Using the lower angles of pitch resulted in a good overall shape for the landscape. There were localised areas of high distortion (e.g. on some parts of the road) and some holes, which may have been due to those areas being obscured by trees. 3DF Zephyr could determine 1904 camera positions, most being accurate, as shown in Figure 48(b) on page 100. The reconstruction of buildings was poor, as shown in Figure 49(b) on page 100, but the road surface in general was good (due to the low distortion of the landscape).

The intermediate angles gave the best overall result, with a good reconstruction of the landscape shape and no highly distorted areas. 3DF Zephyr determined 1727 camera positions, most being accurate, as shown in Figure 48(c) on page 100. Detail on buildings had better reconstruction than that obtained with the lower angles but was not as good as that with the higher angles, as shown in Figure 49(c) on page 100. The intermediate angles gave the best reconstruction of road detail.

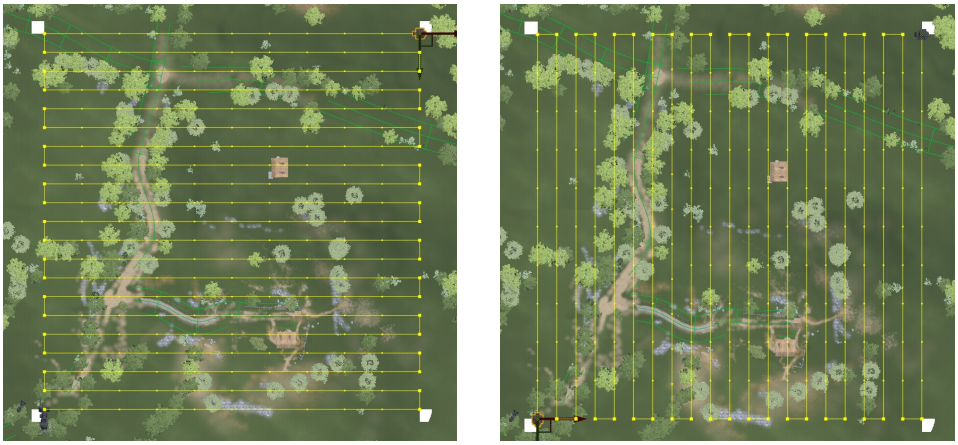
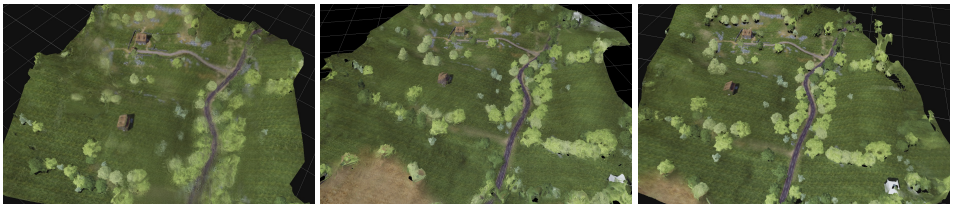


Figure 45: Orthogonal Scans of the Unreal Engine 'Country Side' Model Created in the Matinee Editor.

In conclusion, it is apparent that the choice of camera angle at each scanning level can have a large impact on the overall reconstruction quality. The optimum values for the heights and angles to use will be dependent on the type of environment. For landscapes with a gently undulating shape (such as in the Unreal Engine *Country Side* model) a scan with a low angle of camera pitch (i.e. 25° - 35°) may be necessary to resolve the small differences



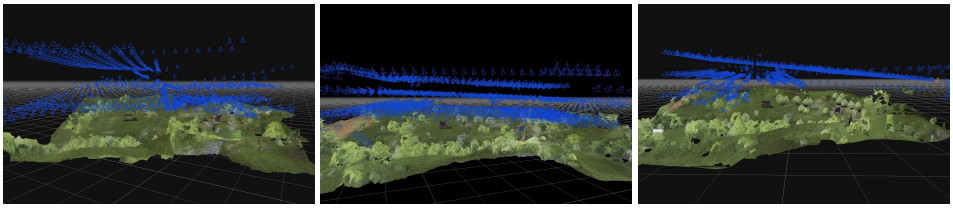
(a) High angles (b) Low angles (c) Intermediate angles

Figure 46: 3DF Zephyr Reconstructions from Multi-level Scans Differing in Camera Pitch. 3DF Zephyr textured meshes reconstructed from multi-level scans of the 'Country Side' model using different sets of camera pitch angles.

(a) High angles (b) Low angles (c) Intermediate angles

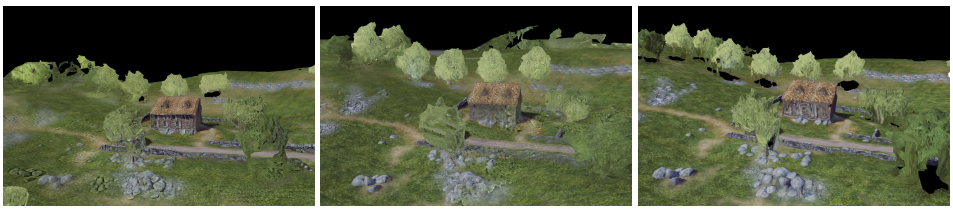
Figure 47: Reconstructions from Multi-level Scans Differing in Camera Pitch Imported into Unreal Engine. Reconstructions of 'Country Side', from multi-level scans using different sets of camera pitch angles, imported into Unreal Engine.

in height between nearby points. The separation between two points in an image due to a height difference will be greatest for low angles of pitch whilst the separation in the image due to the distance between the points horizontally will be greatest for high angles of pitch. The surprising result that the set of higher angles gave the best results for the reconstruction of detail on building walls may be due to the small size of the buildings compared to the extent of the landscape and the large camera to building distance during the greater part of a scan. The distance from the camera to the location on a building wall corresponding to a pixel will be very large for most images and a small percentage error in the depth calculation can lead to a significant distortion of the wall (because of it's relatively small size). Using higher angles of pitch will on average reduce the camera to surface distance corresponding to pixels and hence it can be expected that the set of higher angles will give better results by reducing depth errors. Because of the small size of buildings and the relatively high scan heights, images of buildings in camera shots are typically small with most of the frame being taken up with landscape. Increasing the camera pitch will also increase the average size of a building in an image and make feature recognition easier. From the results it appears that a pitch angle of 45° (used in the higher set of angles) gives low depth errors and is also able to resolve detail on vertical surfaces such as walls (i.e. views of such surfaces are not too oblique).



(a) High angles (b) Low angles (c) Intermediate angles

Figure 48: Computed Camera Positions for Multi-level Scans. Computed camera positions in 3DF Zephyr for models reconstructed from multi-level scans using different sets of camera pitch angles.



(a) High angles (b) Low angles (c) Intermediate angles

Figure 49: Building Details for Multi-level Scans. Building details in 3DF Zephyr textured mesh models reconstructed from multi-level scans using different sets of camera pitch angles.

Because of the relatively large size of landscape features compared to the buildings and the ability of the grass texture on the landscape to mask small distortions, the errors from depth calculations at low pitch angles (i.e. the set with a lowest pitch of 25°) do not cause a noticeable distortion of the landscape. To obtain a good reconstruction for both landscape and buildings it is probably advisable to perform an additional orbital close-up scan (with little or no background in the frame) at a low angle of camera pitch around each building. It should be noted that for any set of scan height and angle parameters, capturing more images during the scan (i.e. increasing the in-track or cross-track overlap) will improve reconstruction quality.

It is clear from this analysis that an optimization process is needed to determine which camera heights and angles to use for a scan of a particular environment. If a single scan is to be used then a compromise may be needed, with the reconstruction of one element, such as buildings or terrain prioritised.

6.5 Modelling Constant Height, Fixed Camera Angle Scans for Photogrammetry Image Capture

Previous research on the use of drones as a platform for capturing images, to be used in photogrammetric environment reconstruction, has largely concentrated on flights in which the camera gimbal rotation and drone height are allowed to vary continuously throughout the flight, with the aim of optimizing the overall reconstruction quality subject to constraints on the maximum flight length (and hence the maximum flight time). In many cases an initial scan of the area at a default height is used to generate a scene proxy for use in the optimization process. Using such methods to scan wide areas is impracticable in many cases because of time constraints. The path optimization itself, given a proxy scene model, can be very made very efficient, with computational times of approximately 5s and 1.5s using 'Submodular Orienteering' and 'Next-Best-View' algorithms for travel budgets upto 1500m [60]. However the time to perform an initial scan and then a reconstruction to create a proxy model can be considerable, and it will increase rapidly as the scan area becomes larger. Another problem with these strategies is that, even though most drones now have an autopilot mode for following a preprogrammed flight plan, the precision to which a drone can actually follow the programmed path can be limited by many factors including the accuracy of the GPS receiver, the drone control system and the weather [66] [67]. There is usually a trade off to be made between flight accuracy and speed, which is more pronounced in difficult conditions (e.g. high winds), making highly accurate scans for large areas unfeasible. In many cases (especially for urban environments) it will also not be possible to adhere strictly to the optimum trajectory due to safety and privacy considerations. The flight path will need to be checked and possibly modified to ensure adequate distances are maintained to buildings, the ground and other hazards. Filming permissions may be needed in certain areas and obtaining these may take a considerable time. Because of these limitations with drone trajectory optimization most scans for photogrammetry are currently carried out under manual control using empirical methods [65].

As an alternative optimization approach, rather than using an initial scan to generate a scene proxy which is then used to determine the camera angles and trajectory of the final drone scan, in many cases it may be sufficient to optimize a single scan, using a regular scanning pattern (e.g. rectangular or orbital) at a fixed height and camera angle. Executing such a single scan will be faster and drone control is simplified, with manual operation possible in some situations. As noted in Section 6.4, the particular height and angle to use will need to be carefully chosen to optimize the photogrammetry reconstruction, subject to a minimum height to maintain safety margins. For environments consisting of structures or features which are mostly similar in shape and size, which have a relatively uniform

arrangement (e.g. buildings of similar height, ground profile and spacing), and in which there are no specific features of particular importance, it can reasonably be assumed that using such an optimized single scan will give good results. This was confirmed using reconstructions from simulated scans in Unreal Engine of the 'Industrial City' model (see Figure 56 (b) and (c)). Test flights or simulations of flights over uniformly structured areas could be used to gain insight of how the reconstruction quality varies with drone height and camera angle parameters for different types of environment (e.g. rural and urban) and with different scales and densities of features (e.g. average building density and height). This information could then be used to determine the optimum drone parameters to use in actual drone scans, given prior knowledge of the general form of the environment, which in many cases can be easily determined from sources such as Google Earth or a quick on-site survey.

A scan at one height may not be sufficient to give the required level of quality in the reconstructed model. This is more likely to be the case for environments which are very non-uniform in the shape, size and arrangement of structures or features, or which are generally uniform but have some areas of significance which cannot be accurately resolved from the generally optimal scan height. For these areas additional scans at one, two or three other levels will in most cases give satisfactory results [65].

To study the effectiveness of employing a scan at a constant height with a fixed camera pitch and develop an optimization procedure for such scans, a system was created, using the Python programming environment provided in the Blender 3D modelling software, to estimate the photogrammetric reconstruction quality resulting from a simulated scan.

6.5.1 Benefits of Simulation and Selection of the Programming Environment

Rather than carry out a time intensive study using actual drone flights to investigate the effect of scan parameters on photogrammetry reconstruction, it was decided to develop a tool to estimate the reconstruction quality from a simulated scan over a 3D model of an environment. In many cases the simulations can use a simple, quick to build environment model or an existing model. The effect of a single scan parameter (e.g. focal length) can then be determined by running a series of simulations, each varying only in that one parameter. In addition to savings in time when compared with the approach of performing many real scans and photogrammetry reconstructions, another benefit of simulation is that the results are repeatable whilst those using actual scans are dependent on the accuracy of the drone trajectory (which will be affected by factors such as the weather conditions and the precision of the positioning system) and also by the prevailing lighting con-

ditions. The aim of the study was to develop a simulation tool that could be used to determine how the choice of parameters affects photogrammetric reconstruction, to find the optimum scanning parameters for a particular environment and to investigate how the form of the environment influences the optimum parameter values. This novel approach of optimizing particular scan parameters contrasted with previous research which largely focused on optimizing a scan trajectory for a given environment. The simulation was created using the Python development environment provided within the Blender 3D modelling program. This environment has a number of useful features which made it suitable for the development of the system:

- Environments can be modelled in Blender or imported from other packages through interfaces for various formats such as Filmbox (FBX), Wavefront (OBJ) or Stereolithography (STL).
- The Blender Python *bpy* module can be used to access Blender model data, classes and functions from a Python script.
- Python supports the Object Orientated programming paradigm.
- Python packages are available that fulfil many common requirements for scientific programming. The *NumPy* library adds support for efficient programming with multi-dimensional arrays. The *Matplotlib* library allows data to be visualized using a wide variety of 2D and 3D plot formats. The *SciPy* library provides many modules for use in mathematics, science and engineering, such as *optimize* for minimizing or maximizing objective functions.
- Python can be used to automate Blender command sequences and create add-ons that extend functionality.
- The Blender software is free, open-source and can be run on a wide range of platforms.

6.5.2 Program Structure

Program Classes

A number of class types were defined to implement objects storing geometry and view coverage data and to model camera objects. Appendix D.1 gives a list of these classes along with definitions for class methods.

A *Model_Object* class object is used as a container to store information about each Blender object of type 'MESH'. Together, these objects form the model of the environment. The object contains pointers to the Blender data structures holding the vertices, edges, loops and polygons for the corresponding mesh. The object also contains lists holding pointers to front (camera) facing polygons, vertex normals and *Hemisphere* objects, each of

which holds viewpoint direction data for a vertex.

The *Camera* class is used to model a camera and has properties defining the focal length, screen width, screen height, pitch angle, yaw angle and a vector defining position. It also has an array property containing three vectors defining the directions of the x,y and z camera axes, and a matrix property representing the transformation matrix required to convert global coordinates to camera coordinates. Methods are defined for this class to set camera properties, to move the camera by an incremental distance or to an absolute position and to check if a given vertex is visible on the camera screen.

A *Hemisphere* class object is used to store data representing the view directions from which a vertex has been captured by a camera and directions giving views similar to these captured views (which would not provide significant extra information for reconstruction). A *Hemisphere* object is created for each vertex in every selected Blender 'Mesh' object. Rather than use a continuous distribution for view direction data, directions are discretized (using the method of Roberts *et al.* [60]) by defining a set directed from the vertex to a uniform distribution of points over a hemisphere of radius 1m, centred at the vertex and having the pole axis directed parallel to the vertex normal. The locations of the hemisphere surface points are stored as an array of vectors (*hemi_locs*). An array *hemi_values* is used to store the view coverage information, with a value of 1 stored in an array element indicating that the corresponding direction in array *hemi_locs* has been covered by a camera, and a value of 0 indicating it has not been covered. A method *set_hemi_values* is defined in the *Hemisphere* class to update the view coverage stored in *hemi_values* with view data corresponding to a camera at a location given as a parameter. This will set the corresponding coverage value in *hemi_values* to 1 for all sampling points in *hemi_locs* that are within a disk, centred on the projection of the camera on to the hemisphere and having a size defined by equation 44 (given on page 88). A method *calc_hemi_metric* is defined to calculate the contribution of the vertex j corresponding to the *Hemisphere* to the overall reconstruction quality metric using equation 47 (given on page 89):

$$F(C_j) = \sum_{k=1}^{K} w_j(\mathbf{h}_k)\, v_j(\mathbf{h}_k) \tag{65}$$

In the above equation the $v_j(\mathbf{h}_k)$ for each sampling point k on the hemisphere are the values stored in coverage array *hemi_values* and the weighting factors $w_j(\mathbf{h}_k)$ are defined by equation 45 (given on page 88).

Global Data Structures and Functions Defined in the Main Program

For each Blender object of type 'MESH' (assumed to be part of the environment model) the main program will create a corresponding object of type *Model_Object*, which is added to a list *model_objs*. For certain types of investigation it is not necessary to evaluate the view coverage of every vertex in the environment model in order to determine the optimum scanning parameters. It may be the case that only the reconstruction of particular buildings or features is of interest, and for uniform environments it is only necessary to consider a centrally positioned representative area. Although the coverage of other areas does not need to be calculated, it is still necessary to include any features surrounding those of interest (particularly tall buildings or hills) in the environment model, since they may obstruct the view from certain positions and thus have an influence on the optimum parameters. The program will assume that all objects (of type 'MESH') that are selected in Blender need to be evaluated for reconstruction quality and on initialization the corresponding *Model_Object* class objects are added to a list *model_sel_objs*.

The following functions (detailed in Appendix D.2) were defined in the main program of the system:

- *calc_poly_visibility*: A back-face culling algorithm is used to update the list of front facing polygons that is stored in *Model_Object* class objects.
- *vertex_visible*: Determines if a vertex (given as a parameter) is visible on the camera screen.
- *update_coverage*: Updates the accumulated view coverage information for all objects in *model_sel_objs* with new data using the current camera position.
- *calculate_metric*: Calculates the average reconstruction metric for all vertices of objects in *model_sel_objs*. The metric used is a scaled version of the coverage metric used by Roberts *et al.* [60], given in equation 47. The scale is chosen so that the maximum metric value of 1.0 represents every vertex having coverage from all possible directions.
- *scan_x_dir*: Perform a rectangular scan with scan lines directed alternately in the +x and -x directions and with an increment in the y coordinate of the camera position after each scan line. After each movement of the camera the accumulated view coverage information is updated.

- *scan_y_dir*: Perform a scan with scan lines directed alternately in the +y and -y directions.

On initialization the main Python script performs the following tasks:

1. An object of type *Camera*, representing the drone camera, is created.
2. For every Blender object of type 'Mesh' a corresponding object of type *Model_Object* is created and added to list *model_objs*.
3. Objects from list *model_objs* are added to list *model_sel_objs* if the corresponding Blender object is selected (and hence needs to be evaluated for reconstruction).
4. A scan is performed, for example a rectangular scan in the x direction followed by a rectangular scan in the y direction, using functions *scan_x_dir* and *scan_y_dir*.
5. A metric value estimating the reconstruction quality is evaluated using the function *calculate_metric*.

Support for the Optimization of Multi-level Scans

Rather than using a single scan at a fixed height it may be required to perform a series of scans, each at a different height and using a different camera angle, to achieve good reconstruction. Global functions and methods for the *Hemisphere* class have been added to the system to support such multi-level optimization. After optimizing a scan at a particular level, a scan at the next level should start with the *Hemisphere* objects initialized with coverage data (stored in array *hemi_values*) corresponding to that at the optimum for the previous scan. This will ensure that the new scan will be optimized to add the maximum additional coverage information to that provided from previous scans. To support this type of scan, data structures *hemi_values_best* and *hemi_values_saved* have been added to the *Hemisphere* class to store the best coverage data found so far during an optimization and the optimum coverage data found after completing the optimization for a level. Global functions added to the program include:

- *set_best_hemi_values*: Save the coverage data (to each Hemisphere's *hemi_values_best* array) during a level optimization when the reconstruction metric exceeds the previous highest value.
- *save_best_hemi_values*: After the optimization of a level, save (to each Hemisphere's *hemi_values_saved* array) the best coverage data found.

Number of model vertices	Number of scan lines	Total number of image capture points	Scan analysis time (s)
805	5	25	90
805	10	100	310
805	20	400	1120
6387	10	100	16200

Table 6: Effect of Model Complexity and Scan Parameters on Metric Calculation Time.

- *set_to_saved_hemi_values*: Used at the beginning of each trial scan during level optimization to initialize the coverage data to that at the optimum of the previous level (stored in each Hemisphere's *hemi_values_saved* array).

6.5.3 Program Testing and Response to Changes in Scan Parameters

Tests were carried out to evaluate the program performance for different model complexities and to determine the typical variation of the calculated reconstruction quality metric with changes in scan parameters such as the camera height, camera angle, scan cross-track separation and the camera focal length.

Program Performance

The model shown in Figure 50 (a) on page 109, with buildings at varying height, was used for testing program performance. For the analysis, different versions of the model were created using various sizes for the surface mesh. The time to perform a rectangular scan in x and y directions, for different values of mesh density, cross-track separation and image capture point density, were recorded. Some of the results, for a PC with an Intel Core i7-6700 processor running at 3.4 GHz, are shown in Table 6. It can be seen that for a fixed model complexity, the analysis time is directly proportional to the total number of image capture points, i.e. the number of scan lines (determined by the cross-track separation) multiplied by the number of capture points along each scan line. The analysis time was found to be approximately proportional to the square of the model complexity, so that doubling the number of surface points increased the time by a factor of four. To achieve a practical level of performance with large environment models it will be necessary to exploit the high degree of parallelism possible for the calculations performed by the system. In particular, the computation of the visibility of a vertex for a given camera position and the corresponding update in coverage data for the vertex can be carried out independently

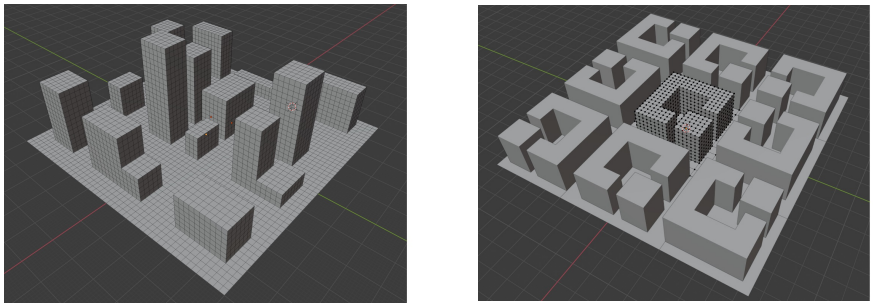
for each vertex. These calculations can also be carried out for each camera position in parallel, although the shared access to *Hemisphere* objects may need to be controlled to prevent data inconsistency. One method of achieving parallel execution for Blender Python code is to configure the Blender system to use a Python compiler such as Numba instead of it's own compiler [68]. The Numba high performance JIT compiler can be used to translate Python functions into PTX (Parallel Thread Execution) code which can be executed on CUDA compatible GPU hardware [69].

Response of the System to Variation in Scanning Parameters

To achieve meaningful results in optimizing scan parameters with the system, any objects in the environment for which an accurate reconstruction is important will need to be accurately represented in the Blender model and the facets of the object (e.g. rectangular surfaces modelling walls) need to have a mesh size (i.e. the distance between vertices) that is small compared to the size of the facet. If the model is to be used to determine distance parameters such as the optimum camera height, any results will only be accurate (at best) to this mesh size. Finer meshes may be needed to model particular areas if it is important that the shape of the surface in those areas is accurately reconstructed. For building sizes similar to that in the model of Figure 50 (a), a mesh size of approximately 2m was deemed adequate for testing system performance. For realistic behaviour, when testing the response of the system to changes in scan parameters, models of an urban environment should contain buildings that are fully or partially obscured from certain vantage points due to other structures. For an environment of a typical city centre containing large buildings a test area of at least 100m x 100m was thought necessary, and using a 2m mesh size over this entire area could result in excess of 5000 vertices on the model surface.

The tests on program performance indicated that it would be very time consuming to use a model containing 5000 vertices for the large number of tests required to evaluate the effect of varying each parameter. Because of this a different model, shown in Figure 50 (b), was developed for this phase of testing. This model encompassed an area of 150m x 150m and contained a uniform distribution of buildings of equal height (20m) within a grid-like street pattern. Within each street block the buildings were arranged around a central courtyard having a single entrance. The road width, distance between road junctions and building footprints were modelled on that found typically in an area immediately surrounding a city centre core, using measurements taken from Google Maps. Since the building shape and distribution was uniform it was only necessary to evaluate the reconstruction metric for the central block, which was modelled using a higher mesh density than the surrounding blocks. The total number of vertices in the new model was reduced to approximately 1600, with 1200 vertices in the

central area used for the evaluation of the reconstruction metric.



(a) (b)

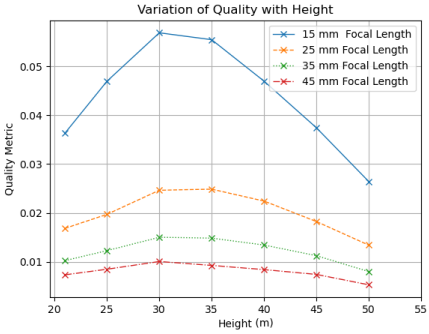
Figure 50: Blender Models of an Urban Environment used for Program Testing: (a) Varying height. (b) Uniform city blocks.

Using this new environment model a series of scans were executed to test the effect of varying scan parameters.

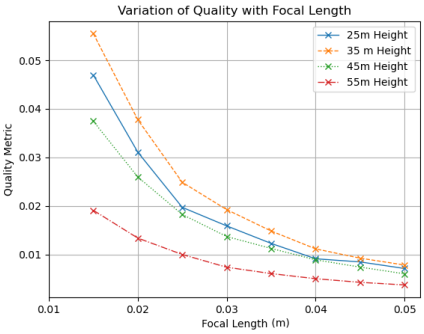
The plots in Figure 51 on page 110 show the variation in the reconstruction quality metric with change in camera height, change in camera focal length and change in camera pitch. A rectangular scan in two perpendicular directions, with a cross-track separation of 8m was used. For plots (a) and (b) a fixed camera pitch of 80° (downwards), was used and for plot (c) a fixed height of 30m was used.

From the results it can be seen that for all values of camera focal length there is an optimum camera height of approximately 30m above ground level. There is a fall-off in quality as the camera height is reduced below this level due to an increasing number of surface points on the central block becoming obscured or exterior to the camera image view. Reducing the camera focal length improves the reconstruction metric as expected, since this will reduce the magnification and increase the number of surface points visible on an image, leading on average to an increase in the number of different view directions captured for a surface point. From plot (c) it can be seen that at a fixed height there is small variation in quality with camera pitch. For focal lengths of between 25mm and 45mm the variation in optimum camera pitch is small with a 60° pitch giving near optimum results for all values. With low values of focal length the optimum pitch (downwards) is increased, with an optimum of 75° at a focal length of 15mm. This could be a result of the low focal length giving a wide field of view, and hence at lower pitch values more images will capture areas outside the model. In reality when scanning a large area nearly all images will contain only the area of interest and hence the optimum pitch can probably be considered independent of the focal length.

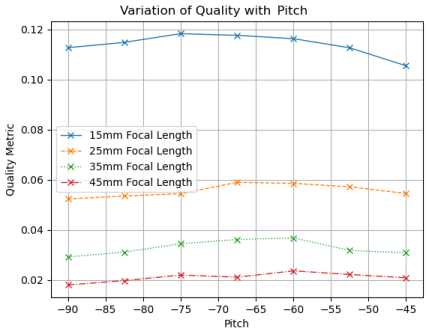
The plots in Figure 52 on page 111 show the effect of varying the number



(a)



(b)



(c)

Figure 51: Effect of Scan Parameters on Quality Metric: (a) Variation with height. (b) Variation with focal length. (c) Variation with camera pitch.

of scan lines used in the scan (i.e. varying the cross-track separation). For plots (a), (b) and (d) a fixed focal length of 35mm was used and for plots (b), (c) and (d) a fixed height of 30m was used. It is apparent from plots (a) and (b) that the optimum height or optimum pitch does not appreciably change when using a different number of scan lines, although as expected the reconstruction quality increases as the number of scan lines increases. Plot (d) shows that the rate of increase in the quality metric (i.e. the increase in the metric for a given increase in the number of scan lines) falls as the number of scan lines becomes larger. This plot also shows that increasing the number of image capture points along a scan line will increase the quality metric, and again the rate of increase in the metric appears to fall as the number of capture points increases. It can be concluded that increasing the number of scan lines (i.e. reducing the cross-track separation) and increasing the number of images captured along each scan line (track) will have diminishing returns for the improvement in reconstruction quality.

The results from these simulations suggest that reducing the cross-track separation and focal length will improve photogrammetric reconstruction.

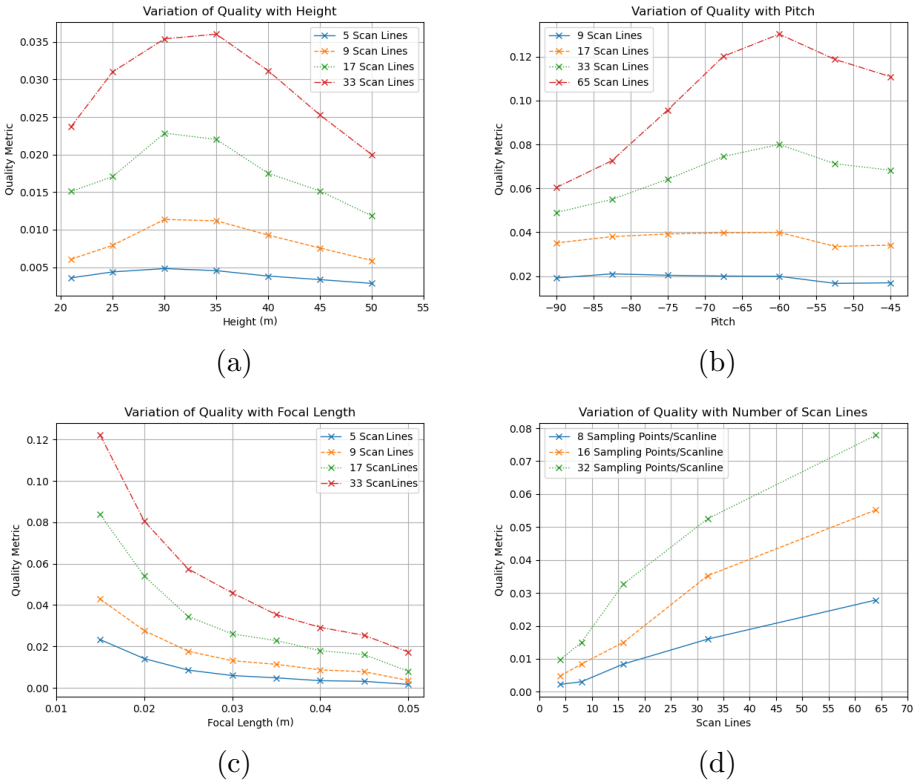


Figure 52: Effect of Number of Scan Lines (Tracks) on Quality Metric: (a) Variation with height. (b) Variation with camera pitch. (c) Variation with focal length. (d) Variation with number of scan lines for different image capture (sampling point) densities.

However decreasing the cross-track separation will increase scanning time, whilst decreasing the focal length (at a fixed height) will increase the ground sampling distance (GSD) and make smaller features more difficult to resolve. In addition, for a particular value of cross-track separation the focal length needs to be small enough to ensure adequate overlap between successive scan lines, as described in Section 6.3. Hence there is a trade-off between scanning time and the level of detail that can be reconstructed. Increasing the cross-track separation will reduce the scan time but for a fixed height it will increase the minimum field of view required to achieve an adequate overlap and so increase the minimum possible GSD. This will increase the minimum size of an object that can be resolved in the scan.

It is clear that there are no optimum values for focal length and cross-track separation, instead the values used will need to be calculated using equations 56, 57 and 58 (see page 95), given constraints on scan time and GSD. The reconstruction metric appears to have a maximum for particular

values of camera height and camera angle, and these optimum parameter values do not appreciably change with other scan parameters such as the focal length and cross-track separation.

Optimization of Scan Height and Camera Pitch

A study was carried out to investigate the use of the system for the optimization of scan height and angle parameters. The *minimize* function contained in the *SciPy optimize* library was used for the optimization procedure. This function can be used to implement many common types of optimization method including Nelder-Mead, Newton-CG and L-BFGS-B, but none were found to successfully converge to the parameters corresponding to the global maximum of the reconstruction quality metric. The convergence problems may have been due to the use of discrete sampling locations, over the hemispheres surrounding each vertex, to estimate view coverage. A small increment in camera height or angle will result in no additional view directions being added to the coverage model and hence no change in the reconstruction quality metric. For this reason every point in the parameter space of the quality metric function is effectively at a local minimum or maximum. Instead of the *minimize* function the less efficient *optimize.brute* function was used to evaluate the metric over a grid of parameter values and return the maximum value, for example:

grid_params = (slice(22.0, 42.0, 2.0), slice(-90.0, -37.5, 2.5))

res = optimize.brute(func_opt, grid_params, full_output=True, finish=None)

In the above Python code the height and angle parameter ranges and increments for the grid are given by a tuple *grid_params* containing two slice objects. Since *brute* finds the minimum value for a function, the function to optimize, given by parameter *func_opt*, returns the negated value of that returned by function *calculate_metric*. The results of the optimization are returned in the data structure *res*, which contains the minimum function value, the optimum parameter values and the function values at all grid points.

The results from the optimization of the uniform model (see Figure 50 (b) on page 109) using a cross-track separation of 8m and a focal length of 35mm are shown in Figure 53. The reconstruction metric can be seen to fall away rapidly from a central peak occurring at a height of 34m and a camera pitch angle of 60°.

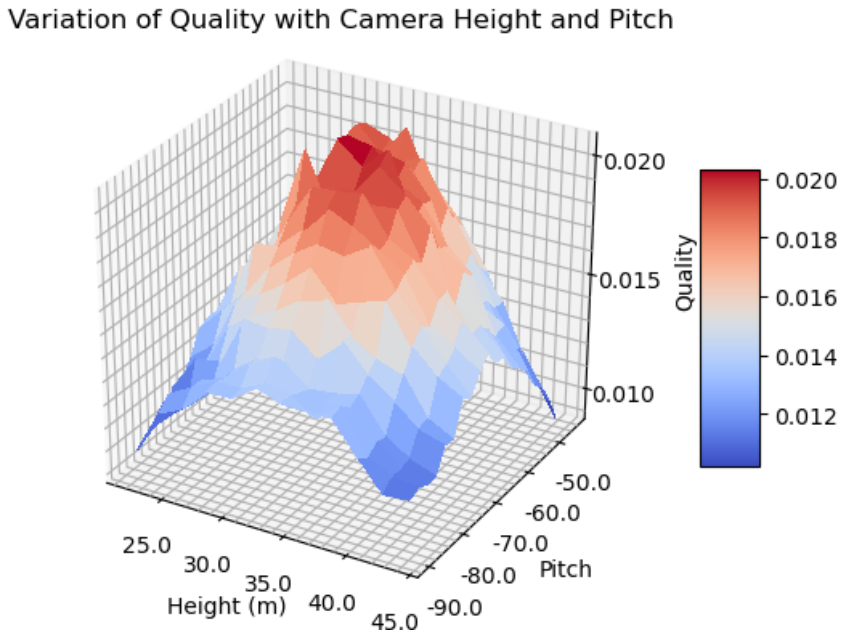


Figure 53: Variation of Reconstruction Quality Metric with Scan Height and Camera Angle for Environment Model Shown in Figure 50 (b).

Response of the System to Changes in the Environment Model

A series of tests were carried out, using modified versions of the uniform environment model (see Figure 54 on page 115), to examine how the calculated reconstruction metric varied with changes in building height and with changes in the ground mesh. Table 7 summarizes the results of these tests.

For the model with no ground mesh, increasing the building height from 20m to 30m increased the optimum camera height by 10m. This is higher than expected, since the average height for a surface patch on the building will only increase by approximately 5m. A possible reason for this is that the

Building Height	Ground Mesh	Maximum Reconstruction Metric	Optimum Camera Height	Optimum Camera Pitch
20m	No Mesh	0.0223	30m	-42.5
20m	5m Mesh	0.0205	34m	-60
20m	2.5m Mesh	0.0206	28m	-62.5
30m	No Mesh	0.0162	40m	-42.5
30m	2.5m Mesh	0.0136	44m	-60.0

Table 7: Effect of Changes to the Environment Model on the Reconstruction Metric.

greater obscuration that occurs with the higher building height will promote clearer views from higher positions. This may also explain why the addition of the ground mesh (of size of 2.5m) to the 30m building model increases the optimum height, whilst for the 20m building model (with less obscuration) the optimum height is reduced. Adding a ground mesh will reduce the average surface height and hence it will be advantageous to move closer to the ground if views are not obscured by surrounding buildings. Changes in mesh density can be used to give certain areas a higher or lower importance with regard to reconstruction. For instance, if only buildings need to be reconstructed then the surrounding ground area can be given a low mesh density or left without a mesh. From the results it can be seen that when there is no ground mesh the optimum pitch angle is 42.5° (down) for both 20m and 30m building heights. Increasing the mesh density on the ground, so that it given more importance, increases the optimum downward pitch angle.

Although there is a complex relationship between the form of the environment (e.g. building height, building and ground mesh densities) and the optimum camera parameters, in all models tested the results were similar in form to that shown in Figure 53 and ranges for parameter values that gave an optimum for the reconstruction metric could clearly be identified.

6.5.4 Comparison of Estimated Reconstruction Quality with Results from Actual Photogrammetry

To assess the effectiveness of the system for use in finding optimum scanning parameters, a series of photogrammetric reconstructions were created using images from simulated scans of the *Industrial City* model in Unreal Engine (illustrated in Figure 55 (a) on page 116), with varying values for camera height and pitch. The results of these reconstructions were compared against the optimum parameter values predicted using the Blender system. The scans extended over an area of 120m x 120m containing a central building (of height 13m) along with seven surrounding buildings (with a maximum height of 17m). For the Blender analysis a simplified model of the scanned area was created, consisting of buildings modelled as blocks along with a ground plane. The dimensions and layout of these buildings accurately matched the Unreal model. This approximation was deemed appropriate since the objective of the study was to optimize the reconstruction of large scale features, such as the terrain and buildings, in an environment. The buildings in the Unreal model were also very block like, having flat roofs and few 3D features that significantly diverged from the wall surface planes. It is usually unnecessary to accurately reconstruct in 3D small scale features on buildings, such as window and door frames, drain-pipes and window ledges. In most cases, when the viewing distance is not too close, they can be convincingly represented by the texture applied to the reconstructed surface

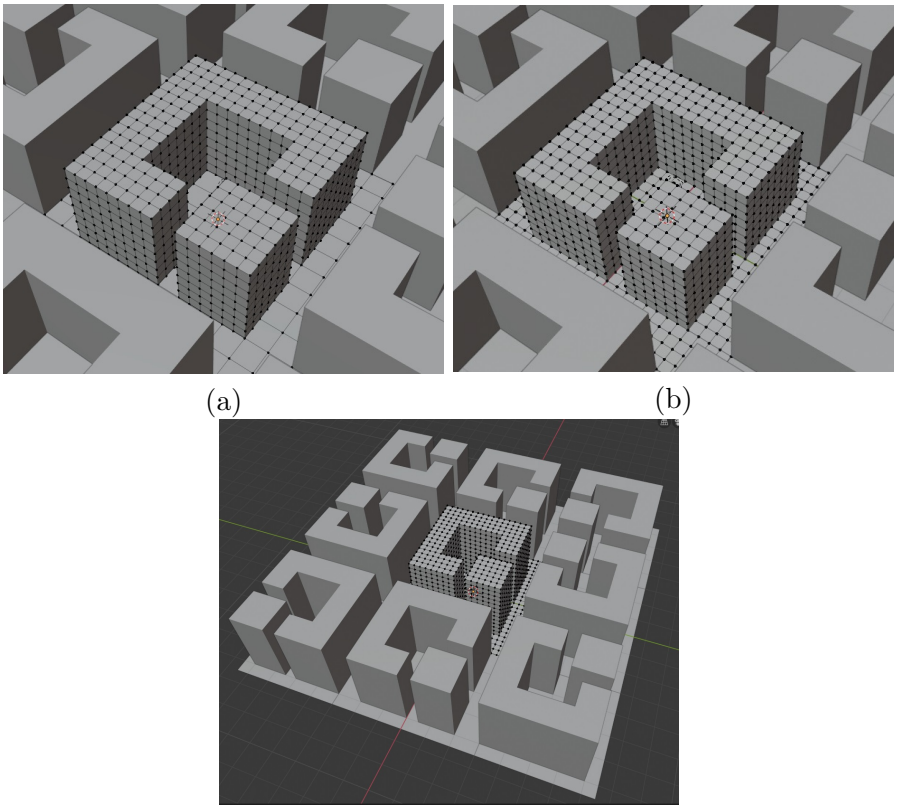


Figure 54: Modifications to the Test Environment Model: (a) Original model with 20m building height and coarse (5m) ground mesh. (b) 20m building height with fine (2.5m) ground mesh. (c) 30m building height with fine (2.5m) ground mesh.

mesh in the texture mapping phase of reconstruction. Small or thin objects and scene clutter such as electric power lines, road signs, traffic lights, traffic cones, rubbish bins and wire fences were also removed from the Unreal scene. These features would be difficult to accurately reconstruct using images from a single overhead scan and would result in many unwanted artefacts in the reconstructed model. For evaluation of the reconstruction quality only a central area of 45m x 45m, consisting of a single building and its immediate surroundings, was considered (see Figure 55 (b)). This area was modelled in Blender with a higher mesh density than the surrounding area, using a 1.5m face size, which resulted in approximately 1500 surface points being used for evaluation of the reconstruction quality metric. The scanning pattern used for image capture, consisting of two perpendicular rectangular scans with a cross-track separation of 10m, is shown in Figure 55 (c). In the Unreal simulation images were captured every 5m along a scan line, resulting in 24

images per scan line and 624 images per scan. A *CineCameraActor* with a Filmback Setting of 16:9 DSLR (screen size of 36mm x 20.25mm) and a focal length of 35mm was used in Unreal and an equivalent camera, implemented with the *Camera* class, was used in the Blender system.

The results of the optimization using the Blender system are shown in Figure 55 (d). The maximum reconstruction metric occurs at a height of 22m above ground level and at a camera pitch of 62.5° downwards, with a rapid fall-off in quality as the height changes from this optimum value. It can be seen that the optimum pitch increases as the height increases, with the maximum in the reconstruction metric occurring at a pitch of 57.5° for a height of 18m and at a pitch of 85.0° for a height of 26m.

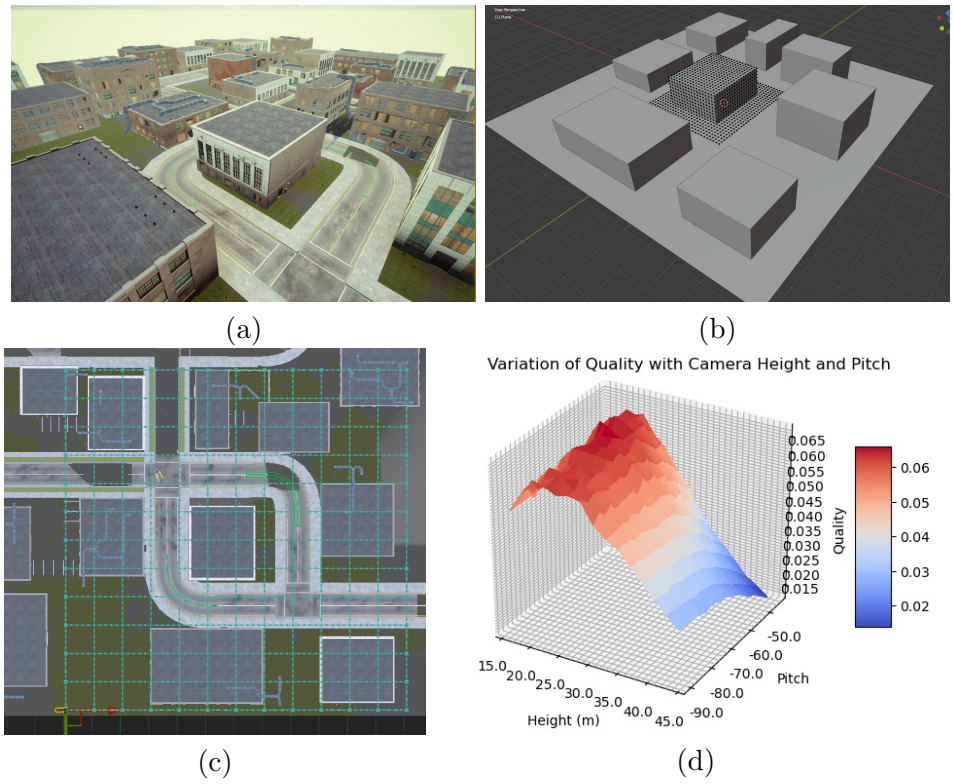


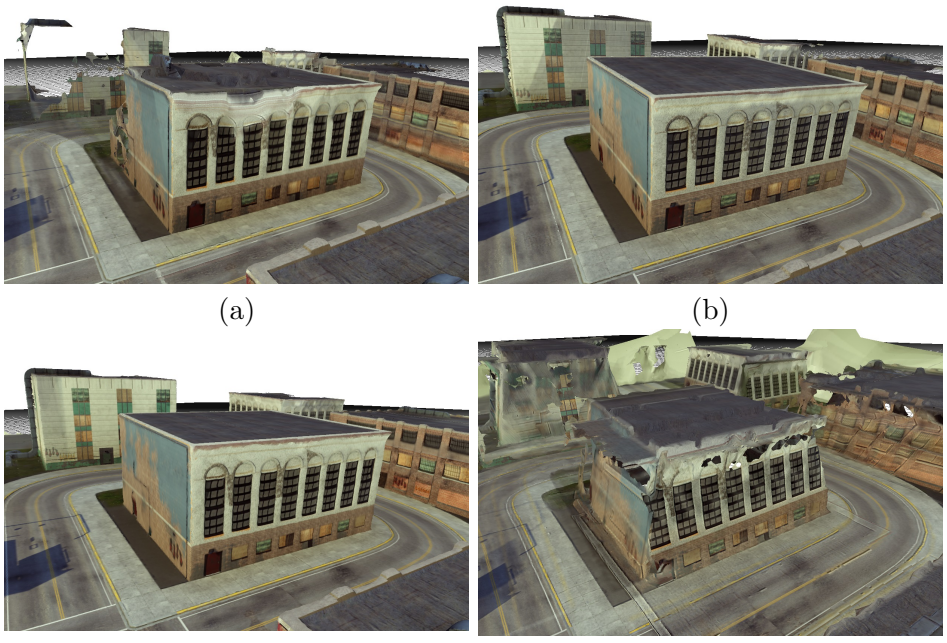
Figure 55: 'Industrial City' Scan Simulation: (a) Unreal Engine model. (b) Simplified model in Blender. (c) Rectangular scanning pattern. (d) Results from scan optimization in Blender.

The image sets from the Unreal Engine scans were used to create 3D reconstructions of the scene using the 3DF Zephyr photogrammetry software. Reconstructions for camera heights ranging from 32m to 44m, all using a camera pitch of 60° are shown in Figure 56 on page 118. Reconstructions for scans using a camera height of 40m, with camera pitch ranging from 45° to 90° are shown in Figure 57 on page 119. The Blender model predicted

that when using a camera pitch of 60° a scan height between 20m and 22m will give the optimum reconstruction quality. However the reconstructions of the Unreal model showed that camera heights of between 36m and 40m gave good results, with scan heights below 32m and above 44m producing models with significant distortions. A possible explanation for this disparity is that the reconstruction metric only gives a measure of the average quality and distribution of views for vertices. Close up views of particular vertices can promote a high metric value even if those vertices are only viewed from a single direction or there are some other vertices that are not present in any views. The calculation does not take in to account the image overlap requirements necessary for successful reconstruction, which can vary greatly with image quality and the particular photogrammetry software used.

Using equation 56 (given on page 95), at a height of 22m (the optimum found using the Blender system) an image, taken using a camera with a focal length of 35mm and a screen width of 36mm, will cover a width of 22.63m at ground level and a width of 9.26m at the height of the central building roof (which is at 13m above ground level). With a cross-track separation of 10m, adjacent images of the ground from successive scan lines will have an overlap of approximately 56%, whilst there will be no overlap for adjacent images of the roof (with a strip 0.74m wide being off-screen between each scan line). At a height of 32m (for which the reconstruction using the Unreal simulated scan had some distortion) an image will cover 19.54m of roof (a 49% image overlap) and at a height of 36m (which gave a good reconstruction with the Unreal scan) an image will cover 23.66m of roof (a 58% image overlap). It is apparent that an image overlap in excess of 50% is necessary for a good reconstruction of the *Industrial City* model using the particular scan pattern and camera employed. In general, the minimum required overlap may be influenced by the image quality and hence be dependent on factors such as image resolution and lighting conditions. The image quality will affect the ability of the photogrammetry software to match key-points between images and the performance of different photogrammetry software with a given overlap will vary due to differences in matching algorithms. An overlap of 50% probably represents a minimum requirement in most cases, since this will ensure that all surface features appear in at least two images.

A simple work-around to the mismatch between the Blender reconstruction metric and actual reconstruction quality due to image overlap considerations could be to limit the optimization process to height ranges which are known to be suitable, given the cross-track separation and camera focal length. Modification of the parameter values used in the calculation of the reconstruction metric may also improve the accuracy to which it models real-world photogrammetry applications.



(a) (b) (c) (d)

Figure 56: 'Industrial City' Reconstructions for Scans Differing in Height with Camera Pitch at 60°: (a) Camera at 32m height. (b) Camera at 36m height. (c) Camera at 40m height. (d) Camera at 44m height.

6.5.5 Analysis of the View Coverage Reconstruction Metric

To gain insight into the finding that the Blender system gave lower than expected values for optimum scan heights, further investigations were carried out to determine the response of the system to changes in scan height and the effect of varying parameters governing the estimation of the view coverage.

Variation of the Reconstruction Metric with Scan Height

Figure 58 shows how the number of surface points that have a reconstruction metric of zero varies with the scan camera height, for the Blender *Industrial City* model. The scan used a fixed camera pitch of 60° and a cross-track separation of 10m. For low camera heights the points with zero metric are not visible in any image due to gaps in the coverage between scan lines, with a height of between 22m and 24m being necessary to ensure visibility. At camera heights above 34m an increasing number of points have a metric of zero. This is due to the fall off with camera height in the size of the disk projected onto a hemisphere (given by Equation 44 on page 88) and possibly limitations due to an insufficient density of the sampling points used to evaluate coverage data in the *Hemisphere* class. The optimization

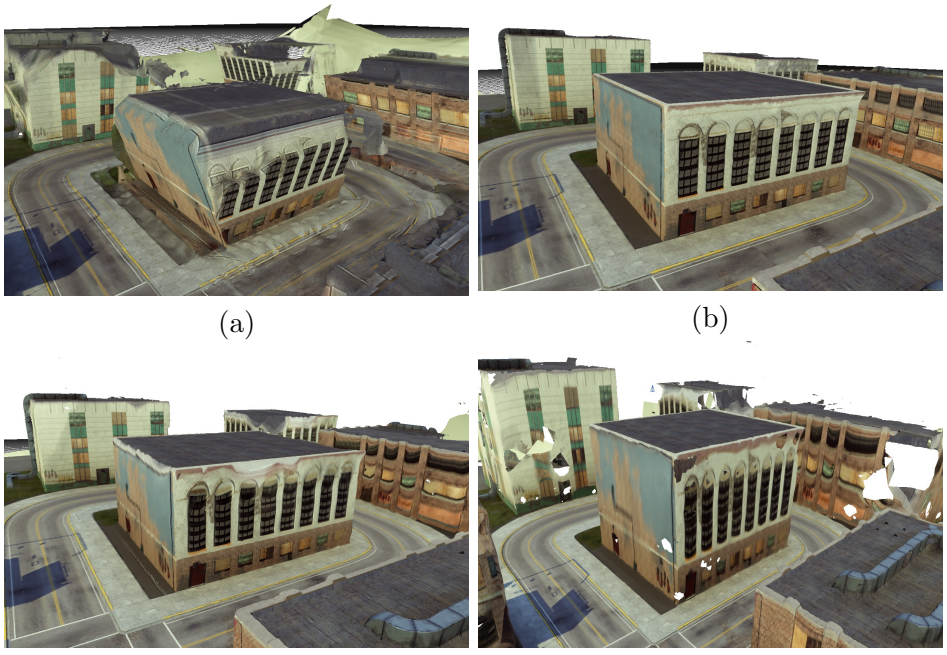


Figure 57: 'Industrial City' Reconstructions for Scans Differing in Camera Pitch with Camera Height at 40m: (a) Camera pitch of 45°. (b) Camera pitch of 60°. (c) Camera pitch of 75°. (d) Camera pitch of 90°.

of this scan described in Section 6.5.4 had shown that for a camera pitch of 60° the optimum height predicted by the Blender system was 20m. At this height, from Figure 58, there will be 60 surface points (out of a total of approximately 1500) that are not visible in any image. The number of points which cannot be reconstructed using actual photogrammetry will probably be even larger, since it is necessary to have a point in at least two images for reconstruction. At a height of 24m all points are captured, but the greater average camera to surface point distance results in a lower value for the calculated reconstruction metric than that for a height of 20m.

The relatively small size of the *Industrial City* model limited the range of scan heights that could be tested with the system, since over a certain height large proportions of an image may cover areas that have not been modelled. To investigate the effect of scan height on the reconstruction metric over a wider range of height, an environment model of much larger extent than the *Industrial City* model was required. For this purpose an OpenStreetMap model of an area near Lancaut, Gloucestershire was imported into Blender using the blender-osm add-on, as shown in Figures 59(a) and 59(b). The model was edited to remove unwanted details such as fences, paths and trees. Although the terrain, building positions and building footprints were accu-

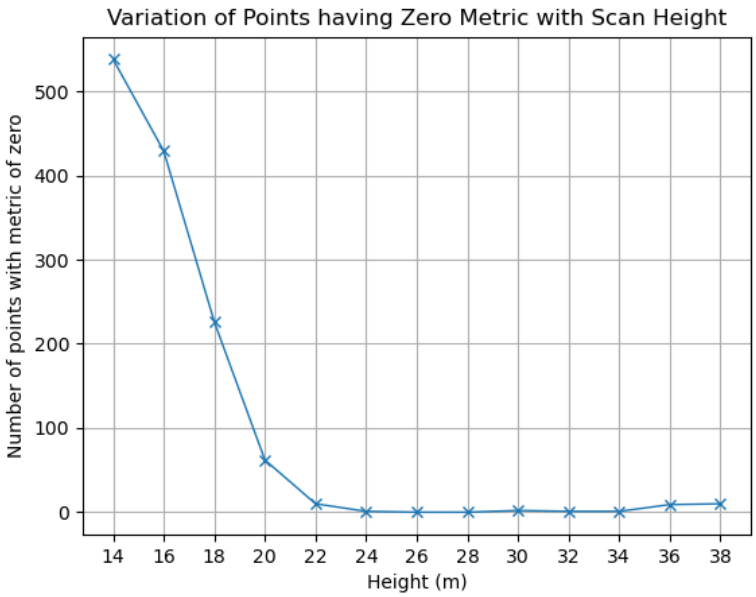
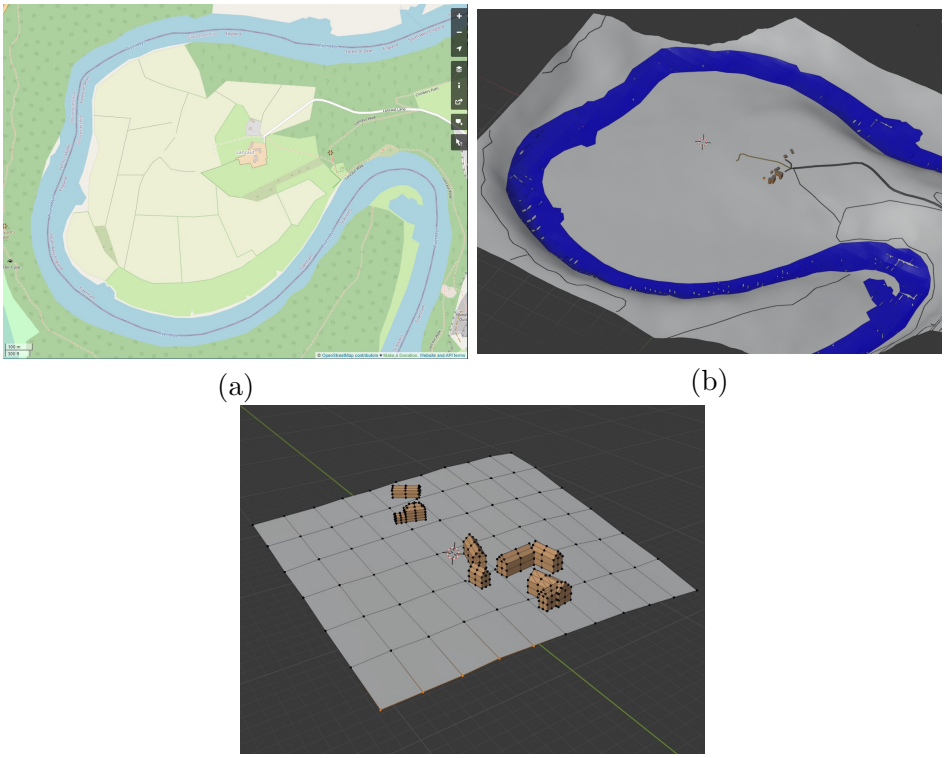


Figure 58: Effect of Camera Height on Coverage for 'Industrial City' Model.

rately represented, the buildings themselves were simply modelled as blocks. Extra details such as building roofs were added to the model in Blender, using information from sources such as Google Earth and photographs of the area, to create the model shown in Figure 59(c). Rectangular scans of the model were performed in Blender for camera heights ranging from 80m to 250m above sea level (the model itself varied in height from approximately 51m to 80m above sea level). The scans used a cross-track separation of 20m and a camera having a focal length of 24mm, a screen size of 23.5mm x 15.7mm and a pitch of 80°. Using equation 57 (given on page 95), a camera height of approximately 40m above the highest point on the model (i.e. at 120m above sea level) is required to achieve an image overlap of at at least 50% between scan lines, and hence ensure good reconstruction. Figure 60 (a) on page 122 shows how the average reconstruction metric and the maximum reconstruction metric for a surface point vary with scan height. The average reconstruction metric (which does not take into account image overlap) indicates that the optimum height is at 90m above sea level. The maximum metric value for an individual surface point can also be seen to fall steeply as the scan height increases. From Figure 60 (b) on page 122, it can be seen that the number of points which have a reconstruction metric value of zero falls to a minimum at a height of 100m above sea level. At this height the width of terrain (at the maximum surface height of 80m) captured in each image is nearly equal to the cross-track separation, and

hence most surface points will be captured on at least one image. As the height increases the number of points having a value of zero for the metric increases rapidly and with a scan height of 160m above sea level this is true for 95% of points.



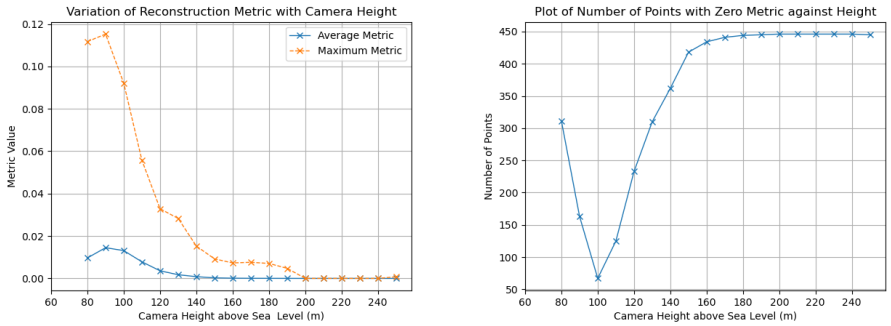
(a) (b)

(c)

Figure 59: Lancaut Models: (a) OpenStreetMap model. (b) Model of buildings and terrain imported into Blender using Blender-osm. (c) Final Blender model.

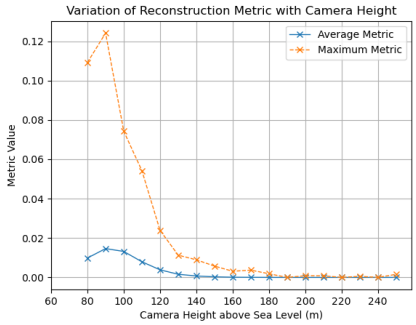
An analysis was also performed to determine if an insufficient number of view sampling directions for vertex *Hemisphere* objects could be responsible for this rapid fall off with height. The coverage data for a sampling point (corresponding to a view direction) on a hemisphere is set to a value of 1 only if the point falls within a disk on the surface of the hemisphere, centred at the projection of a camera on the hemisphere surface, with a size given by equation 44 on page 88. With only a small number of sampling points, at large camera distances the size of the disk will be small and it is possible that no sampling points fall within the disk. Hence, even with a large number of images capturing a point, if the camera distances are large the reconstruction metric for that point can have a value of zero. The previous analyses used 256 sampling points over a *Hemisphere* object.

Figures 60 (c) and 60 (d) show the results of the Lancaut analysis when using 1024 hemisphere sampling points. Increasing the number of sampling points slightly increased the maximum reconstruction metric for a vertex and reduced the minimum number of points that had a metric value of zero, but had little effect on the average reconstruction metric. It was concluded that 256 hemisphere sampling points was sufficient and the rapid fall off in the reconstruction metric value with increasing height was largely due to the parameters used in the estimation of view coverage.

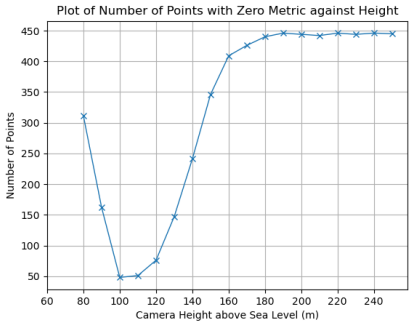


(a)

(b)



(c)



(d)

Figure 60: Lancaut Reconstruction Metrics: (a) Average and maximum reconstruction metric using 256 sampling points on Hemisphere. (b) Number of points with a metric of zero using 256 sampling points on Hemisphere. (c) Average and maximum reconstruction metric using 1024 sampling points on Hemisphere. (d) Number of points with a metric of zero using 1024 sampling points on Hemisphere.

Variation of the Reconstruction Metric with the Parameters of the View Coverage Calculation

In the previous analyses the parameters used in the calculation of the reconstruction metric (see equation 44 on page 88) had values that were equal to those used in the implementation by Roberts *et al.* [60]: $t_0 = 4$, $t_{half} = 12$ and $\theta_{max} = \pi/8$. It was clear that these values were not always appropriate and to ensure the reconstruction metric accurately matched the actual reconstruction quality obtained through photogrammetry they would needed adjusting to suit the particular scan employed.

Parameter t_0 represents the distance which is considered close enough for maximum reconstruction quality. In particular t_0 specifies the distance from a camera $\mathbf{c}_i$ to a vertex $\mathbf{s}_j$ for which the size of the disk (projected on a hemisphere surrounding $\mathbf{s}_j$), defined by cone half angle θ_i^j, is a maximum. At camera distances t_i^j less than or equal to t_0, $\theta_i^j = \theta_{max}$. One possible strategy for matching the reconstruction metric to actual photogrammetric reconstruction quality is to adjust the value of t_0 so that it equals the camera height at which all images on a scan line overlap with images on adjacent scan lines by an amount that has been found sufficient for the particular photogrammetry software being used (probably 50% or more). For high angles of camera pitch, if there is little obscuration of surface points, the maximum reconstruction metric for a scan will usually occur at a height of approximately t_0 above the lowest point on the surface. For this scan height most surface points will be within a distance t_0 of the camera when it is overhead and hence the coverage data for the camera will be maximized. Even though points will be closer to the camera for lower scan heights the overall view coverage will be reduced since less surface points will appear in each image. As the camera height is increased above t_0 more of the lower lying surface points will become further away than t_0 from the camera when it passes overhead and hence the coverage data for these points due to overhead camera positions is reduced. However there is a counter-balancing effect due to more points becoming visible in each image which will tend to increase coverage at greater heights. The optimum scan height will be dependent on the relative strengths of the fall-off due to increased camera to surface point distances and the effect of more points becoming visible (which will be dependent on the height distribution of surface points), and may be greater than the expected value of t_0 above the lowest point. Using equation 57 given on page 95, an estimation for the minimum value of t_0 necessary to model image overlap requirements for surface points is given by:

$$t_{0min} = \frac{\mathrm{D_S} \times \mathrm{FL}}{(1 - \mathrm{O}) \times \mathrm{SS_W}} \tag{66}$$

In the above equation $\mathrm{D_S}$ is the cross-track separation, FL is the focal length, O is the required overlap and $\mathrm{SS_W}$ is the camera sensor width. Using this

value for t_0 will guarantee image overlap for those regions of the environment at or near the lowest altitude, but not necessarily for higher regions.

If there is a large amount of surface obscuration (e.g. due to tall buildings) the maximum reconstruction metric can occur at heights significantly above t_0.

Modifying parameter t_0 to model image overlap requirements for reconstruction is applicable in some situations, such as the case of a single scan using a regular scanning pattern in which there is a fixed distance between individual scan lines. In other situations, such as when using a series of scans at multiple levels, it is not necessary to have such strict image overlap requirements for each individual scan, rather there needs to be sufficient overlap when considering the entire set of images. In these cases more complex strategies for setting an appropriate value for t_0 would be needed.

The rapid fall off in reconstruction metric with height, as shown with the analysis of the Lancaut model, is due to the relatively low value of 12 used for parameter t_{half}. Increasing the distance t_i^j from camera $\mathbf{c}_i$ to vertex $\mathbf{s}_j$ by 12m will decrease the half cone angle θ_i^j, defining the size of the disk projected onto the vertex *Hemisphere*, by a factor of 2. This will in turn reduce the area of the disk and the number of enclosed sampling points by a factor of approximately 4 (for small values of θ_i^j), and potentially reduce the reconstruction metric by this factor.

Although this value for t_{half} will be suitable for close-up scans of buildings, in many situations such as high altitude scans over large areas it will not be appropriate. The actual fall off in the usefulness of an image for photogrammetry with height is dependent on many factors such as camera resolution, camera settings (e.g. aperture, shutter speed and ISO), atmospheric conditions, light level, uniformity of lighting and the stability of the camera platform. For a good quality HD camera having a sensor width of 36mm with 1600 pixels and a focal length of 35mm, the GSD (using equation 58) at 10m height will be approximately 3mm/pixel. At a height of 20m the GSD will be approximately 6mm/pixel. For photogrammetry camera calibration, assuming clear weather conditions and good lighting, it should be possible to match reasonably sized features using images from either height. For example, if a feature size of at least 10x10 pixels is needed for matching then features of size 3cm and above can be matched at 10m and features of size 6cm and above can be matched at 20m. The GSD values at both heights are also sufficient for the reconstruction and texture mapping of most large scale environment models. From this simple analysis it can be deduced that, for large scale environment reconstructions, the effective change in reconstruction quality when increasing the camera scan height by 12m (the default value used for t_{half}) may be small. To accurately model the true fall off in reconstruction quality with height the value of t_{half} will need to be chosen with regard to the level of detail required in the reconstruction,

camera properties and environmental factors. For detailed scans of small objects (low GSD) a low value for t_{half} is appropriate, whilst a higher value will be needed for scans of large objects that require less detail. Increasing the camera resolution will generally increase t_{half} whilst poor lighting conditions will decrease t_{half}.

6.5.6 Configuration of View Coverage Parameters for a Scan

Figure 61 shows how changes in parameters t_0 and t_{half} affect the calculated reconstruction metric for scans of the Lancaut model at varying heights, using a fixed camera pitch of 80°. From Figure 61 (a) it can be seen that for $t_0 = 40$, which corresponds to the height above the surface required to give an overlap O of 50% as given by Equation 66 (given on page 123), the optimum scan height is 110m above sea level, which corresponds to 40m above the average altitude for the environment. For a value of $t_0 = 52$ the optimum scan height is approximately 120m which corresponds to 40m above the maximum altitude of the environment. Hence using a value of $t_0 = 40$ will ensure that at the optimum height (as determined from the reconstruction metric) there is a 50% image overlap between scan lines for areas of the model at, or below, the average height. A value of $t_0 = 52$ will ensure there is always a 50% image overlap between scan lines for all areas of the model at the optimum scan height. It can be seen that Equation 66 will need to be modified to ensure the calculated value for t_0 gives adequate overlap for all regions of the environment.

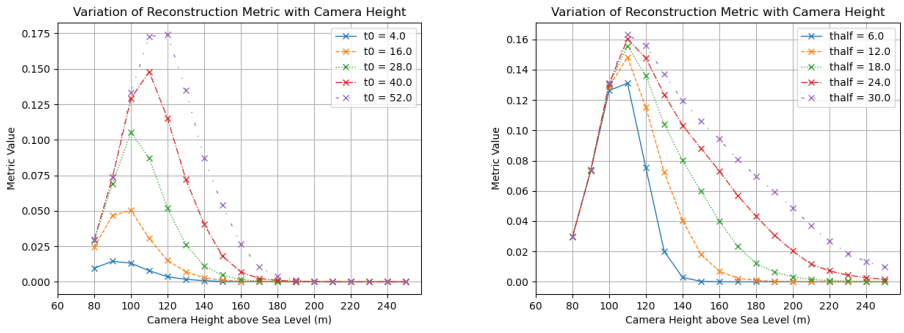
For a camera looking vertically downwards, at a height t_0 above the lowest point of a surface (which is at a height H_{min}), the distance to the camera for all surface points will be less than or equal to t_0 when the camera is overhead and hence for these camera positions the contribution to the reconstruction metric is maximized. For relatively flat surfaces or surfaces with isolated peaks, as the height of the camera increases above t_0, large numbers of surface points will become further away from the camera than distance t_0 and this will tend to reduce the coverage metric, since this factor will dominate over any increase due to more points becoming visible in each image. Hence for relatively flat surfaces the reconstruction metric maximum will occur at a height close to $H_{min} + t_0$. To ensure adequate image overlap for all areas of a surface, including any peaks at height H_{max}, the value of t_0 will need to be increased by $(H_{max} - H_{min})$ from the value given by Equation 66 on page 123. This will move the height for the maximum metric a distance of at least t_{0min} (specified by Equation 66) above H_{max} in all situations. A suitable value for t_0, for use with high values of camera pitch and an environment which is generally flat with isolated peaks is given by:

$$t_0 = \frac{\mathrm{D_S} \times \mathrm{FL}}{(1 - \mathrm{O}) \times \mathrm{SS_W}} + (H_{max} - H_{min}) \tag{67}$$

This value for t_0 can also be used when only the maximum height difference in the landscape is known. However, a smaller value for t_0 can be used if a good estimation for the mean surface height $\overline{H}$ is known. Since the maximum in the reconstruction metric will occur at a height of approximately $\overline{H} + t_0$, a value of t_0 given by equation 68 (given on page 126) will ensure that the optimum height will occur near $H_{max} + t_{0min}$:

$$t_0 = \frac{\mathrm{D_S} \times \mathrm{FL}}{(1 - \mathrm{O}) \times \mathrm{SS_W}} + (H_{max} - \overline{H}) \tag{68}$$

From Figure 61 (b) it can be seen that increasing t_{half} does not affect the optimum height although it slightly increases the maximum reconstruction metric.



(a) (b)

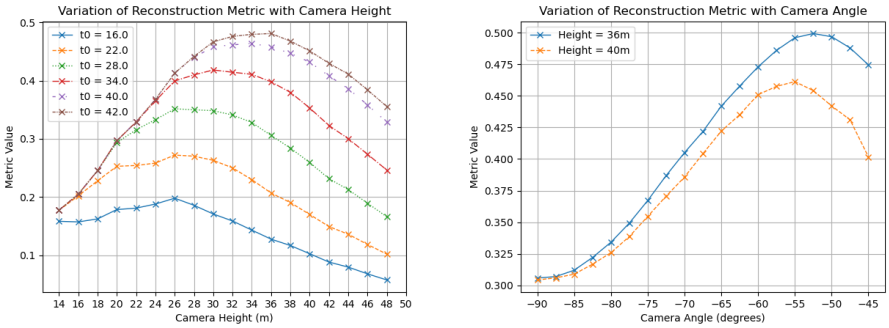
Figure 61: Variation of the Reconstruction Metric with View Coverage Parameters for Lancaut Scans: (a) Variation with height for different values of t_0 (t_{half} fixed at 12). (b) Variation with height for different values of t_{half} (t_0 fixed at 40).

For scans with low or intermediate angles of pitch, the calculation of a suitable value for t_0 is more complex. Figure 62 (a) shows the effect of parameter t_0 on the reconstruction metric for scans of the *Industrial City* model, using a fixed camera pitch of 60°.

For a camera with a pitch angle θ, the distance d from a point centred on an image to the camera will be related to the height h of the camera above that point by:

$$\sin\theta = \frac{h}{d} \tag{69}$$

From the plots of Figure Figure 62 it can be determined that for large values



(a) (b)

Figure 62: Reconstruction Metric for 'Industrial City' Scans: (a) Variation with height for different values of t_0 (t_{half} fixed at 12). (b) Variation with pitch angle for camera heights of 36m and 40m ($t_0 = 42, t_{half} = 12$).

of t_0 there appears to be a similar relationship between t_0 and the camera height above H_{min} corresponding to the maximum reconstruction metric:

$$\sin\theta \approx \frac{H_{opt} - H_{min}}{t_0} \tag{70}$$

In the above equation, H_{opt} is the optimum camera height found from the analysis. A better approximation, analogous to equation 68, for cases where the difference in heights for the landscape ($H_{max} - H_{min}$) are significant compared to the scan height ($H_{opt} - H_{min}$) is:

$$\sin\theta \approx \frac{H_{opt} - \overline{H}}{t_0} \tag{71}$$

For good reconstruction H_{opt} should have a minimum value equal to the camera height necessary to give the required overlap for images of the surface at H_{max}:

$$H_{opt} = \frac{\mathrm{D_S} \times \mathrm{FL}}{(1 - \mathrm{O}) \times \mathrm{SS_W}} + H_{max} \tag{72}$$

Hence a suitable value of t_0, for a camera with angle of pitch θ is given by:

$$t_0 = \frac{\frac{\mathrm{D_S} \times \mathrm{FL}}{(1-\mathrm{O}) \times \mathrm{SS_W}} + (H_{max} - \overline{H})}{\sin\theta} \tag{73}$$

For the *Industrial City* scan at 60° camera pitch, $H_{min} = 0.0$, $H_{max} = 13.0$, $\mathrm{D_S} = 10.0$, $\mathrm{FL} = 0.035$, $\mathrm{SS_W} = 0.036$ and $\theta = 60°$. Assuming $\overline{H} \approx H_{min}$, Table 8 gives the values of the estimated optimum height (calculated using $H_{opt} = t_0\, sin\theta$) and the image overlap O that the optimization should guarantee (calculated using Equation 72), corresponding to various values

of t_0. The optimum height found using the reconstruction metric returned from the Blender simulation is also given, but it should be noted that this has an accuracy limited by the height discretization used in the optimization (2m).

For the larger image overlaps (corresponding to higher scans) it can be seen that there is a good agreement between the estimated value for the optimum height ($H_{opt} = t_0\, sin\theta$) and the actual height obtained from the optimization, and hence it can be deduced that for high scans (where $\overline{H}$ can be assumed to be approximately equal to H_{min}) Equation 73 can be used to determine a suitable value of t_0 for the optimization given the required overlap. For lower scan heights (where $H_{max} - H_{min}$ is comparable to the scan height) the relationship between t_0 and H_{opt} is more complex and this approximate model is not appropriate to calculate a suitable value of t_0 to use for the optimization.

Actual reconstructions using simulated scans in Unreal Engine and 3DF Zephyr photogrammetry software had determined that a scan height of 32m produced a distorted model. Scans at heights of 36m and 40m both produced good results, with the scan at 36m being slightly better. This suggested that the minimum image overlap between scan lines for good reconstruction was between 50% and 58%. From this it can be deduced that a value of $t_0 = 42$, corresponding to an overlap requirement of 58.4% is appropriate to model the *Industrial City* reconstruction for a 60° scan, with the optimum in the reconstruction metric occurring at approximately 36m. This is very similar to the actual height of 36.37m required to achieve 58.4% overlap for surfaces at the roof height of 13m.

A value of t_0 below 26.24 corresponds to a value of $H_{opt} = t_0\, sin\theta$ below 22.72m, which is less than the camera height required to give any image overlap for surfaces at H_{max}. Low values of t_0 corresponding to no overlap should not be used in practice since they would result in poor reconstructions. While high values of t_0 result in a maximum reconstruction metric near to $H_{opt} = t_0\, sin\theta$, these lower values of t_0 were found to give a maximum in the reconstruction metric significantly above this value, being closer to the height needed to achieve overlap. This is to be expected, since when there is no image overlap there is an advantage in moving the camera higher, with the increase in the reconstruction metric due to more surface points appearing in each image outweighing the reduction in the metric contribution for some surface points due to the increased camera distance.

Figure 62 (b) on page 127 shows how the reconstruction metric varies with camera pitch angle for the case of scans at 36m and 40m, using a value of $t_0 = 42$. The optimum angle at 40m was 55°, which is consistent with the reconstructions from simulated scans (shown in Figure 57 on page 119), for which, out of the limited number of test cases used, an angle of 60° gave the best results. The optimum pitch angle for a scan height of 36m was 52.5°, which is also consistent with previous findings (i.e. higher scans generally

Parameter t_0	Image Overlap, O (%)	Optimum Height Calculated using $H_{opt} = t_0\, sin\theta$	Optimum Height from Reconstruction Metric (m)
42	58.40	36.37	36
40	55.08	34.64	34
34	40.88	29.44	30
28	13.57	24.25	26
22	0.0	19.05	26

Table 8: Effect of Parameter t_0 on the Optimum Height Determined from the Reconstruction Metric.

have higher optimum pitch angles). The results of the optimization are in much closer agreement with real photogrammetry when compared against the first optimization of the *Industrial City* model, which used the same value for t_0 as Roberts *et al.* ($t_0 = 4$) and for which the optimum pitch angle at a height of 40m was 82.5°. It is clear that to accurately model reconstruction quality the Blender system will need to be carefully configured with appropriate values for t_0 and t_{half}.

Care needs to be taken when using Equation 73 (given on page 127) to calculate t_0 for the optimization of scan camera angles, since it implies that the best value to use for t_0 varies with the particular camera angle used. However during an optimization this value needs to be kept constant to prevent the reconstruction metric for low angles of pitch being artificially high when in fact camera to surface point distances can be large. Although this equation will guarantee that the optimum solution obtained from the Blender system, using the calculated value for t_0, will have the required overlap for all angles of pitch greater than θ, it may discount better solutions with higher angles of pitch, for which the average camera to surface distance is smaller than t_0, but have less surface points in each image. For optimizations involving small ranges in pitch angle variation or for analyses involving a fixed pitch angle and varying height then using a fixed value of t_0 given by Equation 73 is appropriate. With these analyses it is not necessary to limit the height range under test, as the value of t_0 should guarantee overlap requirements.

Scans at a low angle of camera pitch and at heights significantly below t_0 can still have camera to surface point distances near to t_0 and hence give high values for the reconstruction metric. For the optimization of scan camera angle (or both height and angle), a work-around to prevent the system finding an optimum for low angles of pitch with heights below that necessary for the required overlap, is to use a value of t_0 corresponding to Equations 67 or 68 (given on page 126) and restrict the range of heights to those that give adequate overlap (i.e. above t_0).

Figure 63 shows the results from the analysis of the *Industrial City* model using this approach, with a value of $t_0 = 37.3$ which corresponds to the height needed for a 60% overlap for images of the roof of the central building. It can be seen from Figure 63 (a) that the maximum in the reconstruction metric for a camera pitch of 60° occurs at a height of 33m (since camera to surface point distances at this pitch can have values close to t_0 for heights significantly below t_0) and hence in the optimization it is necessary to restrict heights to those above t_0. Figure 63 (b) shows that the optimum pitch for a camera at a height of 40m is 60° (downwards), which agrees with the actual reconstructions from the simulated scans in Unreal Engine. The full optimization of both height and pitch shown in Figure 63 (c), with height restricted to 38m (i.e. above t_0) gives an optimum in the reconstruction metric for a height of 38m and a pitch of 55°. Since there is little obscuration of the central building the fall off in the reconstruction quality metric as the height increases above 38m is to be expected. The optimum pitch angle of 55° is also consistent with the value of 60° at 40m and the results from photogrammetry.

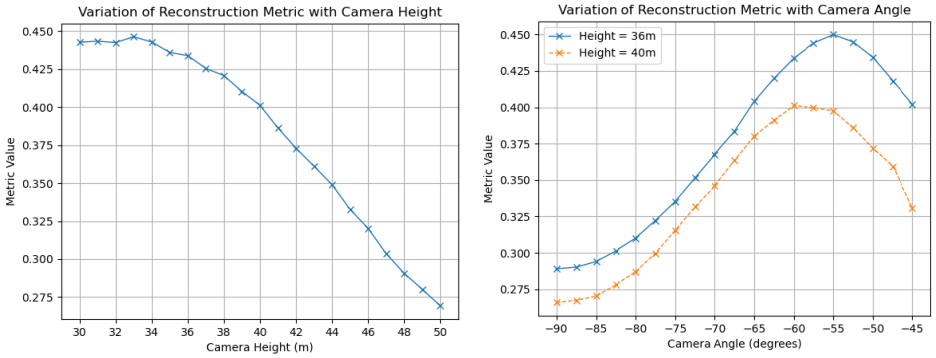
It has been found in practice that using a low angle of pitch generally results in a poor quality reconstruction due the presence of distant objects (outside of the scan area) in many images. However, as shown in Section 6.4, it has been found that an pitch angle of 45° is sufficient to give a good reconstruction for vertical surfaces such as walls and hence for most cases the scan parameter optimization can be limited to pitch angles of between 45° and 90°. If a higher quality of reconstruction is needed for certain structures having vertical surfaces such as buildings, a separate scan around the structure, using a vertical scan pattern with a low angle of camera pitch, may need to be executed.

In conclusion, the hemisphere model used by Roberts *et al.* [60] can be used to estimate the quality for photogrammetry reconstructions given information on scanning parameters and a model of the environment. The accuracy of this method will be highly dependent on the parameters of the coverage model used to estimate reconstruction quality, which will need to be set appropriately for the particular scan being employed.

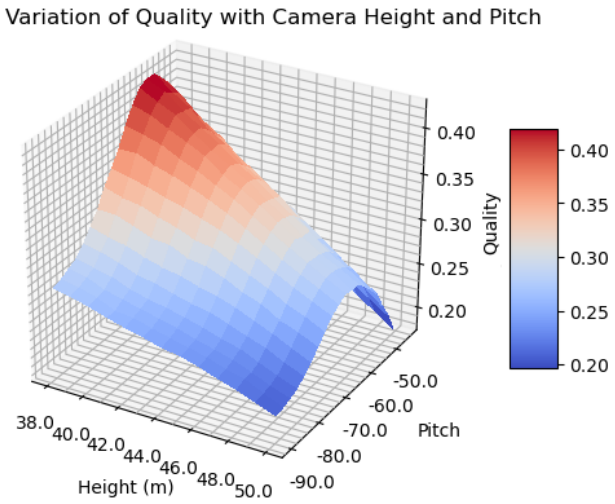
6.5.7 Choice of Scan and Camera Parameters

Optimization using the Blender system will require scan parameters FL (focal length), D_S (cross-track separation) and O (Overlap between successive scan lines) to be known in advance so that coverage parameter t_0 can be calculated. From Equations 57 and 58 (given on page 95), the Ground Sampling Distance (GSD) is given by:

$$\text{GSD} = \frac{D_W}{I_W} = \frac{D_S}{(1 - O) \times I_W} \tag{74}$$



(a) (b)



(c)

Figure 63: Analysis of 'Industrial City' using Modified Coverage Model: (a) Variation of Reconstruction Metric with Height (Camera Pitch Fixed at 60°). (b) Variation of Reconstruction Metric with Camera Pitch (Camera Height Fixed at 40m). (c) Variation of Reconstruction Metric with Camera Height and Pitch.

Hence the maximum value for parameter D_S, for a given image overlap, will be determined by the maximum allowable GSD. The minimum value will be determined by the maximum time allowed for scanning the area, since smaller values of D_S will require more scan lines. The focal length FL will normally be set to the smallest value possible, limited by the minimum clearance distance H_C to surface points. This minimum value for the focal length can be found using Equation 57 (given on page 95):

$$FL_{min} = \frac{(1 - O) \times SS_W \times H_C}{D_S} \tag{75}$$

6.5.8 Improving the Reconstruction Metric

A reconstruction metric based on the vertex hemisphere model proposed by Roberts *et al.* [60] exhibits some unrealistic behaviour and has a number of limitations:

- Close up views of surface points can result in a high reconstruction metric, even if each of those points are only captured in a single view.
- View coverage parameters (t_{half}, t_0 and θ_{max}) need to be matched to the particular scanning situation (e.g. to model the minimum suitable camera to surface distance and fall-off in view quality with height).
- The model does not take into account the use of parallax between views to calculate depth.
- The model does not explicitly take into account reconstruction requirements such as image overlap, which will need modelling through the use of appropriate coverage parameters.

Because of time limitations it has not been possible to evaluate alternative formulations for the photogrammetry reconstruction metric. The method proposed by Smith *et al.* [61] could have a number of advantages. It explicitly models the use of triangulation to calculate depth using information from a pair of views, by incorporating factors representing the variation, with parallax angle and distance, of the triangulation error and the ability to match features between views. Image overlap is encouraged since at least two views of a point are required for it to contribute to the overall reconstruction metric. One problem with the method is that the metric calculation involves five parameters, k_1, α_1, d_{max}, k_3 and α_3, which may need to be modified to suit the details of a particular scan (e.g. the scanning pattern and camera properties) and possibly characteristics of the reconstruction software used. Another factor that may lead to inconsistencies in the reconstruction quality estimation is that the heuristic defined by equation 54 (given on page 90) involves a sum of contributions due to each possible pairwise combination of views. In reality, increasing the numbers of views gives diminishing returns for the resultant improvement in model quality. More realistic behaviour could be achieved by only considering views which provide significant additional information or limiting the sum to the pairwise view combinations that give the best contribution.

Combining the metric calculation (using pairwise view contributions) of Smith *et al.* with a hemisphere model, similar to that used by Roberts *et al.*, to store view information, could result in a reconstruction model which overcomes some of the limitations of these methods. In this proposed method each surface vertex will have a corresponding *Hemisphere* object to store view distance information. For each image (i.e. for each camera position

corresponding to the capture of an image along a scan line), if a vertex is visible the camera to vertex distance is stored at the location in *hemi_values* corresponding to the hemisphere sampling point which has a direction closest to the actual vertex to camera direction (see Section 6.5.2). If there is already a distance stored at the location (from a previously processed image), it will only be overwritten if the new distance is shorter. Using this technique, the number of different views stored for each surface point is limited to the number of hemisphere sampling points, with only the closest camera distance stored for the view direction corresponding to a sampling point. Hence the reconstruction metric will not become artificially high when there are a large number of images or many images are captured from similar viewpoints. There will also be a reduction in the metric calculation time (using equation 54 given on page 90) when the number of images becomes large.

In practice, when matching a particular key-point between images (e.g. during the first phase of Structure from Motion), photogrammetry software will not use every image containing that key-point. Some images may not be of sufficient quality for a successful match and the numbers of matches selected for further processing may be limited to improve efficiency. Hence, rather than use every possible pairwise combination of the views stored in a *Hemisphere* to calculate the sum representing the reconstruction quality heuristic (using equation 54), a closer match to the actual reconstruction quality may be obtained by using the sum or average of the n best pairwise view terms, or in some cases using just the best term. If using a selection of view pairs for the reconstruction metric calculation, each of the pairs should provide significant additional information to that provided by other selected pairs. Hence none of the selected view pairs should have both view directions similar (i.e. within a certain threshold angle) to those in another selected pair.

6.6 Summary

A good quality photogrammetric reconstruction requires an image set containing a wide variety of views for each part of the scene. It has been found that typically between 120 and 180 images per 10000m^2 are necessary for good reconstruction (dependent on the type of the environment). Adding additional views to an image set will have diminishing returns on the improvement in reconstruction quality, since the level of improvement is dependent on the amount of extra surface information contained in those views. Close-up views from new, non oblique directions give the optimum extra information for surface points, with an angle of 20° between two views of a surface point giving the optimum parallax for depth calculation. Since the reconstruction time rises rapidly with the number of images, being of $\mathcal{O}(n^2)$ or greater for n images, it is important to optimize photogrammetry scans to minimize the number of images required to obtain a particular reconstruction quality. An important consideration when scanning an area is to ensure that there is an adequate overlap between the images (e.g. between adjacent images from successive scan lines). To ensure feature key-points appear in at least two images an overlap of at least 50% is required. To ensure a match for a sufficient number of key-points in an image with key-points in another image (to enable the calculation of the homography between the views) it is typically necessary to have an overlap in excess of 60%. Increasing the overlap still further (i.e. up to 80%) can significantly improve quality but will result in longer scanning times. Reducing scan line separation and reducing focal length will also increase the number of views of surface points and improve the reconstruction. However the required level of detail in the final model will determine the maximum GSD and hence the minimum focal length that can be used.

A system built using the Python development environment within Blender has been created to optimize image capture scans. Using this system, scan parameters such as camera height and pitch can be optimized, given a simple model of the environment. It has been found that the optimum height and camera angle for a scan are largely independent of the focal length and cross-track separation. By configuring parameters governing the estimation of view coverage to model image overlap requirements, it has been possible to match the results of the optimization to actual photogrammetry reconstructions.

7 Conclusions and Future Work

UAVs (or drones) provide a flexible and cost effective solution for many filming requirements and their use in media production has become widespread. Novel types of shot that would be impossible or impractical using traditional techniques can be realised when using a drone. Whilst many of the standard techniques of cinematography are still be applicable when filming using a drone, there has been little research in how best to exploit their use.

Filming sports with a drone gives the director a greater flexibility in the choice of shot, they can follow the action to provide close-up shots or fly high to give an overview. The use of multiple drone systems provides further benefits, especially when filming events taking place over a large area. Strategically placed drones will enable the director to switch between multiple shot angles, switch the view to different areas of the action and to respond to incidents. The manual control of a multiple drone system is complex and requires a large team including a director, drone pilots and camera operators. Autonomous functionality such as target tracking, collision avoidance and automatic shooting can greatly reduce operator load and enable control using a small team. For effective filming using a drone automatic shooting system, shot libraries will need to be defined for each particular genre of filming and each shot optimized for viewing quality.

An important consideration in the development of drone filming platforms (including systems with an automatic shooting capability) is the provision of software for flight planning and training. For certain types of film shoot the planning and training can be enhanced if an accurate real-world environment model is provided by the simulation software and hence techniques to efficiently produce such models are required.

7.1 Research Summary

The typical shot types of standard cinematography are not well suited to filming fast moving live action, especially when employing multiple cameras to film multiple targets. To ease the workload on the director and drone operators for the filming of events such as sports races, a custom shot library has been proposed. Shot types are defined according to the motion of the camera relative to the target and arranged using a hierarchical taxonomy based on natural language. Such a rationalized shot library, optimized for live shooting, can reduce the time for the director to decide and communicate which shot to use for a particular situation and also supports the use of a simplified drone control interface for semi-autonomous shooting. Thus it will potentially help to make a multiple drone platform a viable solution for the filming of live events such as sports. The proposed shots have been parameterized to enable integration into the control system of an automatic shooting platform.

Automatic shooting will require suitable default values and operating ranges for each parameter of a shot to be determined to ensure the quality of the viewing experience. A methodology using subjective testing for determining suitable parameter values has been developed and experimental tests performed for a representative selection of the shot types proposed for filming races. For these tests, models were created representing the scenarios of a cycling race and cars racing along a city street using Unreal Engine. Video sequences of these models, simulating camera shots at various drone heights (with a fixed speed) and at various speeds (with a fixed height) were created. In the subjective tests each video sequence was rated for the quality of the viewing experience by a number of participants (59 in total for all tests).

The optimum parameter values to use when filming will be dependent on camera properties such as the focal length and a simple procedure for converting the optimum parameters obtained from the subjective tests to those required when using a different camera has been detailed.

For most filming locations there are no available accurate 3D models that can be imported into drone flight planning software and techniques such as photogrammetry or laser scanning may need to be used to produce a model. A widely used photogrammetry system has been evaluated and procedures to optimize reconstruction quality given limits on reconstruction time and techniques to repair common problems have been established. For certain locations it is possible to use images captured from mapping websites such as Google Earth for photogrammetric reconstruction. However, because full 3D coverage is limited to certain regions (e.g. large urban areas), in most cases images will need to be captured at the location (e.g. by scanning the area with drone). The quality of the reconstruction obtained through photogrammetry is dependent on the number and distribution of images. Using more images generally gives better results, however there are diminishing returns on the improvement in quality as the number of images increases and the time for the reconstruction rapidly increases. Hence for efficient reconstruction it is important to optimize the scanning process so that a limited number of captured images can give good results. A Blender Python module has been developed to help optimize image capture scans for photogrammetry. Scans of simulated environments using the system have shown how the predicted reconstruction quality varies qualitatively with changes in scan parameters and in the type of environment. The system can be used for the optimization of scanning parameters (e.g. camera height and pitch), however it will need to be configured to ensure the estimation for the reconstruction quality given by the program is a good match to the actual quality obtained from the particular photogrammetry software that is to be used. This can be achieved by setting appropriate values for the parameters used in the estimation of the view coverage. In particular, parameters defining the maximum camera to surface distance for optimum surface in-

formation retrieval and the fall-off in surface information with height can be set to model specific scan types and conditions, such as photogrammetry requirements for the image overlap between scan lines. Using this technique the variation in the reconstruction metric with height and camera angle has been matched to actual reconstructions created using images captured in simulated scans of an industrial area in Unreal Engine.

7.2 Research Conclusions

The main conclusions regarding the subjective testing of camera shots for viewing quality can be summarized as follows:

- The methodology of subjectively testing simulated drone camera shots was able to successfully discriminate variations in viewing quality with changes in shot parameters and for most of the shot types tested there was a height range that gave a significantly improved viewing experience compared to that for other heights.
- The optimum camera height relative to the target size generally increased with target size, being approximately 1.4 times the target height for the case of a cyclist and 3.4 times the target height for the case of a car.
- For certain types of shot (e.g. an 'Establishing' shot) males preferred a lower height than females.
- There was a wider variation in the preferences for camera speed compared with the preferences for height, with a statistically significant range for an optimum speed only identified for two of the four shot types tested ('Establishing' and 'Flyby' shots).
- Males generally preferred shots with a higher drone camera speed.

It can be concluded that the technique of subjectively testing simulated drone camera shots is an effective method for determining the ranges for shot parameter values which give a good viewing experience and ranges which should be avoided. It has many advantages over testing using real footage, being much more efficient in terms of cost and time, and allowing parameters to be precisely controlled.

The main conclusions that were ascertained from the work on photogrammetry can be summarized as follows:

- For scanning areas for image capture one of the most important factors has been found to be the overlap between images from successive scan lines. An overlap of 50% has been found to be a minimum requirement, although typically an overlap of approximately 60% or more is necessary for good reconstruction.

- Rectangular scans at a fixed height and camera angle can give good results, however the reconstruction quality is highly dependent on the particular height and angle used.
- The optimum scan height and camera angle do not appreciably change with other parameters such as camera focal length and track separation distance.
- Lowering the cross-track separation and focal length were both found to improve the predicted reconstruction quality, although reducing the focal length also increases the Ground Sampling Distance (GSD) and hence there is a limit to which the focal length can be reduced before the resolution becomes too low to resolve the required level of detail.
- An estimate of reconstruction quality based on the metric of Roberts *et al.* [60] can give a good match to actual reconstruction if parameters of the view coverage model are set appropriately (i.e. to model scan overlap requirements).

The developed Blender system can provide a useful resource for the optimization of scans capturing images for use in environment reconstructions. It can be used to determine optimum scan heights and camera angles given a basic model of the environment. Results obtained from simulated scans of generic environment models can also be used to inform which range of heights and camera angles will probably produce a good reconstruction given basic information on the form of the environment (e.g. average building height and spacing).

7.3 Future Work

There are a number of further research areas in camera shot optimization which could be productive:

- Effect of target height and width on the optimum parameters (the present research focused on a limited range of target types).
- Effect of speed on perceived shot quality, in particular the nature of the dependency on absolute or relative camera speed.
- Effect of shot framing on optimum parameters.
- Effect of shot framing on the importance of absolute versus relative camera speed for influencing the perceived video quality. For example, it may be the case that for close-up shots the speed of the camera relative to the target has more relevance for determining shot quality, whilst for long-shots the absolute camera speed might have greater importance.

- Further work on the effect of gender and age on shot preferences.
- Optimization of the transitions between shots from different drones (e.g. determining optimum drone separation to minimize jarring and confusion).
- Optimization of synthetic views produced using the views from multiple drones.

The purpose of the work carried out so far has been to evaluate the subjective testing methodology and it has focused on a small number of shots for a particular filming scenario. The design of an automatic automatic shooting system using such methods will require the testing of particular shot libraries for each type of filming genre required.

Additional work regarding the Blender Python system developed for the optimization of photogrammetry image capture scans includes:

- Further testing to determine if the system can give optimization results comparable to actual photogrammetry reconstructions for different types of scan pattern and environment (e.g. with a natural landscape or where the area of interest is obscured by surrounding features).
- Testing of the system using multi-level scans. Determine the predicted improvement in reconstruction quality from using one or two additional scanning levels and compare the results with those from actual photogrammetry.
- Development of alternative formulations for the reconstruction metric and evaluation to determine which gives the closest match with actual photogrammetric reconstruction.
- Addition of support for parallel processing using GPU hardware.
- Development and analysis of a series of generic environment models using varying parameters for feature size and placement (e.g. building height and space between buildings). The results of these analyses could be used as a resource to choose suitable parameters for a particular scan given basic information on the form of the environment.

Appendices

A Shooting Requirements for MultiDrone

Table 9: Shot Types and Framing for MultiDrone [6].

Shot Id	Name	Description
CMT1	Still shot	The camera shoots a scene from a fixed point in space without focusing on a specific visual target.
CMT2	Still Shot of non-moving target	The camera shoots a scene from a fixed point in space focusing on a specific non-moving visual target.
CMT3	Static Aerial Pan	The camera shoots a scene from a fixed point in space without focusing on a specific visual target. The camera gimbal rotates slowly (left-right) to capture the scene context.
CMT4	Static Aerial Tilt	The camera shoots a scene from a fixed point in space without focusing on a specific visual target. The camera gimbal rotates slowly (up-down) to capture the scene context.
CMT5	Static Shot of Moving Target	The camera shoots a scene from a fixed point in space focusing on a specific visual target. The gimbal rotates so that the camera focuses on the moving visual target at all times.
CMT6	Moving Aerial Pan	The camera shoots a scene from a linearly moving point in space without focusing on a specific visual target. The camera gimbal rotates slowly (left-right) to capture the scene context.
CMT7	Moving Aerial Pan with Moving Target	The camera shoots a scene from a linearly moving point in space focusing on a specific linearly moving visual target. The gimbal rotates slowly (left-right) so that the camera focuses on the moving visual target at all times. The projections of the drone and target trajectories on the ground are approximately perpendicular.
CMT8	Moving Aerial Tilt	The camera shoots a scene from a linearly moving point in space without focusing on a specific visual target. The camera gimbal rotates slowly (up-down) to capture the scene context.
CMT9	Moving Aerial Tilt with Moving Target	The camera shoots a scene from a linearly moving point in space focusing on a specific linearly moving visual target. The gimbal rotates slowly (up-down) so that the camera focuses on the moving visual target at all times. The projections of the drone and target trajectories on the ground are approximately parallel.

Shot Id	Name	Description
CMT10	Lateral Tracking Shot	The camera shoots a scene whilst focusing on a specific visual target and moving sideways/parallel to the visual target to match its speed if possible. The gimbal is stable and the camera is always focused on the linearly moving visual target. The camera axis is approximately perpendicular to the visual target trajectory and approximately parallel to the ground.
CMT11	Constrained Lateral Tracking Shot	The camera shoots a scene whilst focusing on a specific visual target and moving along the projection of the visual target trajectory on a pre-defined flight plane, matching its speed if possible. The gimbal is stable and the camera is always focused on the moving visual target.
CMT12	Vertical Tracking Shot	The camera shoots a scene whilst focusing on and moving exactly (forwards/backwards) above a specific visual target, matching its speed if possible. The gimbal is stable and the camera is always focused vertically down, on the visual target. The camera axis is approximately perpendicular to the visual target trajectory.
CMT13	Pedestal /Elevator Shot	The camera shoots a scene whilst slowly moving up or down without any rotational movement. The gimbal is stable with the camera always facing ahead (not focusing on a specific visual target).
CMT14	Pedestal /Elevator Shot with Target	The camera shoots a scene whilst slowly moving up or down without any rotational movement, focusing on a specific visual target. The gimbal rotates (up/down) so that the camera is always focused on the linearly moving visual target.
CMT15	Fly-Over	The camera shoots a scene whilst approaching or intercepting the visual target from behind or from the front at a steady speed until positioning over it. The gimbal slowly rotates (up-down) so that the camera is always focused on the non-moving or moving visual target.
CMT16	Reveal Shot	The camera shoots a scene with the gimbal stable and the visual target initially out of frame. The camera moves along a linear trajectory until the target becomes fully visible.
CMT17	Orbit	The camera shoots a scene with the gimbal slowly rotating so that the camera always focuses on the non-moving or linearly moving visual target. The camera moves in a (semi-)circular trajectory around the visual target, relative to the latter's straight-line trajectory (if any).
CMT18	Fly-by	The camera shoots a scene whilst approaching or intercepting the visual target from behind or from the front, passing by it to the left or right side and then moving away on a constant heading. The gimbal slowly rotates up or down so that the camera always focuses on the visual target.

Shot Id	Name	Description
CMT19	Chase/ Follow Shot	The camera shoots a scene following or leading the visual target from behind or from the front, matching its heading and speed if possible so that it remains at a constant distance. The gimbal is stable and the camera is always focused on the visual target.
CMT20	Bird's- Eye Shot	The camera shoots a scene whilst slowly flying up. The gimbal is stable and the camera faces vertically down (not focusing on a visual target).
CMT21	Moving Bird's- Eye Shot	The camera shoots a scene whilst slowly moving forward or backwards at steady altitude. The gimbal is stable and faces vertically down (not focusing on a visual target).
CMT22	Fly- Through Shot	The camera shoots a scene whilst moving forward and flying through an opening/gap/hole. The gimbal is stable and faces ahead, possibly focusing on a visual target.

B Recommended Drone Parameters for Typical Shot Types in a Cycling Scenario

Table 10: Recommended Drone Parameters for Typical Shot Types in a Cycling Scenario.

Shot type	Typical Target/Drone/Gimbal Movement	Parameters (cycling scenarios)
S2.0: ESTABLISH	$\mathbf{T}(t) = (0,0,0)$ $\mathbf{D}(t) = \left(\frac{t}{t_0}(x_e - x_s) + x_s, 0, \frac{t}{t_0}(z_e - z_s) + z_s\right)$ $\mathbf{G}(t) = \left(0°, \arctan\left(\frac{t(z_e - z_s) + t_0 z_s}{t(x_e - x_s) + t_0 x_s}\right), 0°\right)$	$z_s = 2.3H = 3.5m, z_e = 1.3H = 2m$ $(x_s = -70m, x_e = -12m)$
S2.1: CHASE S2.2: LEAD	$\mathbf{T}(t) = \left(\frac{t}{t_0}d_0, 0, 0\right)$ $\mathbf{D}(t) = \left(\frac{t}{t_0}(x_e - x_s) + x_s, 0, z_0\right)$ $\mathbf{G}(t) = \left(0°, \arctan\left(\frac{t_0 z_0}{t(x_e - x_s) + t_0 x_s}\right), 0°\right)$	$z_0 = 2H = 3m$ $(x_s = -50m, x_e = -10m)$
S2.3: FLYBY	$\mathbf{T}(t) = \left(\frac{t}{t_0}d_0, 0, 0\right)$ $\mathbf{D}(t) = \left(\frac{t}{t_0}(x_e - x_s) + x_s, y_0, z_0\right)$ $\mathbf{G}(t) = \left(0°, \arctan\left(\frac{t_0 z_0}{t(x_e - x_s) + t_0 x_s}\right), - \arctan\left(\frac{t_0 y_0}{t(x_e - x_s) + t_0 x_s}\right)\right)$	$z_0 = 2H = 3m$ $(x_s = -15m, x_e = 15m, y_0 = 6m)$
S3: ELEVATOR	$\mathbf{T}(t) = (0,0,0)$ $\mathbf{D}(t) = \left(x_0, 0, \frac{t}{t_0}(z_e - z_s) + z_s\right)$ $\mathbf{G}(t) = \left(0°, \arctan\left(\frac{t(z_e - z_s) + t_0 z_s}{t_0 x_0}\right), 0°\right)$	$z_s = 1H = 1.5m, z_e = 3H = 4.5m$ $(x_0 = 12m)$
S4: ORBIT	$\mathbf{T}(t) = \left(\frac{t}{t_0}d_0, 0, 0\right)$ $\mathbf{D}(t) = \left(r_0 \cos\left(\frac{2\pi t}{t_0}\right), -r_0 \sin\left(\frac{2\pi t}{t_0}\right), z_0\right)$ $\mathbf{G}(t) = \left(0°, \arctan\left(\frac{z_0}{r_0}\right), 180° - \frac{2\pi t}{t_0}\right)$	$z_0 = 2H = 3m$ $(r_0 = 7.5m)$

Above parameters suitable for a camera with focal length of 35mm and sensor size of 23.66mm x 13.3mm. The height of the cyclist (H) is around 1.5m. Reference videos available at https://multidrone.eu/multidrone-public-dataset/.

C Post Processing of Photogrammetry Models using Blender

3D Modelling software can be used to improve the final quality of an environment model before it is imported into Unreal Engine. Typical applications include the removal of artifacts and reducing distortions such as bumps and holes in flat surfaces (e.g. roads or water). Various modelling packages were examined and Blender (developed by the Blender Foundation) was found to have the required functionality for this purpose. Some of the main features of Blender include:

- Import and export models using industry standard file formats such as Filmbox (.fbx), Wavefront (.obj), STL (.stl) and Stanford (.ply).
- 3D modelling (e.g. creation of primitives such as planes, cubes and cylinders).
- Edit mode enables modification of the faces, edges and vertices of an object mesh.
- Sculpt mode provides easy to use tools to rapidly shape large areas to the required form.
- Texture Paint mode provides tools to paint directly on a 3D surface, automatically updating the associated Texture image.
- UV editing allows the modification of existing UV mappings and the definition of mappings for newly created geometry.
- Path tracing rendering engine (Cycles) supporting GPU rendering and the Open Shading Language.
- Open-source and free for commercial use.
- Wide platform support (e.g. Windows, Linux, macOS).

An environment model produced using 3DF Zephyr can be imported into Blender using the FBX import option. Detailed below are a number of techniques that can be used to fix particular problems with a model and improve the overall quality:

C.1 Removing Isolated Mesh Components

Models produced using photogrammetry may contain isolated floating artefacts, e.g. remnants of objects such a poles which have a thin dimension and for which there is not sufficient information to enable correct reconstruction (see Figure 64). To remove such an artefact the following procedure can be used:

1. Select a single vertex on the object.
2. Use the 'Select Linked' option in the Select menu to select the entire object (see Figure 65).
3. Delete all selected vertices. This will remove the object including all associated faces, edges and vertices.

It is also possible to select all 'floating' objects for removal by firstly selecting a vertex on the main surface and then using the 'Select Linked' command to select the entire surface. Finally the 'Invert' option can be used to deselect the main surface and select all unattached surfaces. (see Figure 66).

Figure 64: 'Floating' Artefacts in a Reconstructed Model.

C.2 Smoothing Flat Surfaces

A common problem with photogrammetry is for flat surfaces such as roads to be reconstructed with a bumpy or pitted appearance. This can be due to the surface having a near uniform texture which makes it difficult for the photogrammetry software to match surface points between images. Blender has a Sculpt mode within the 3D editor which allows large areas to be rapidly shaped to a required form. Of particular use is the 'Smooth' tool which will smooth out a surface simply by painting over it with a brush. Whilst using this tool it is preferable to set the display shading to 'Solid' so that its effect on the shape of the surface can easily be judged (see Figure 67 on page 147).

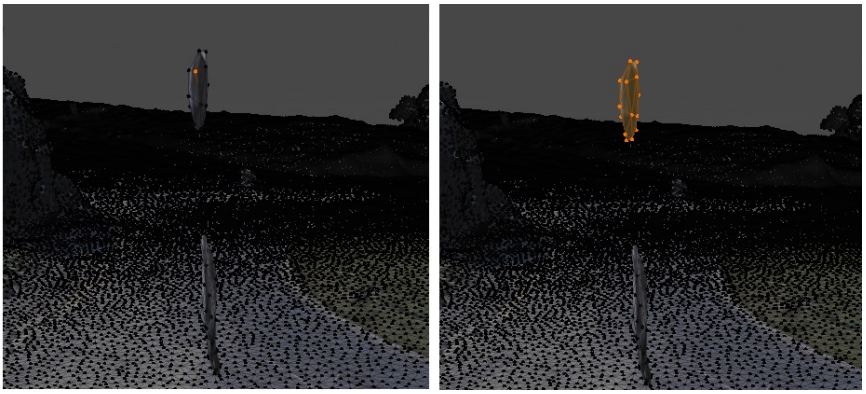


Figure 65: Selecting all Vertices in a 'Floating' Artefact. Select a vertex on the artefact (left) and then use the 'Select Linked' command (right).

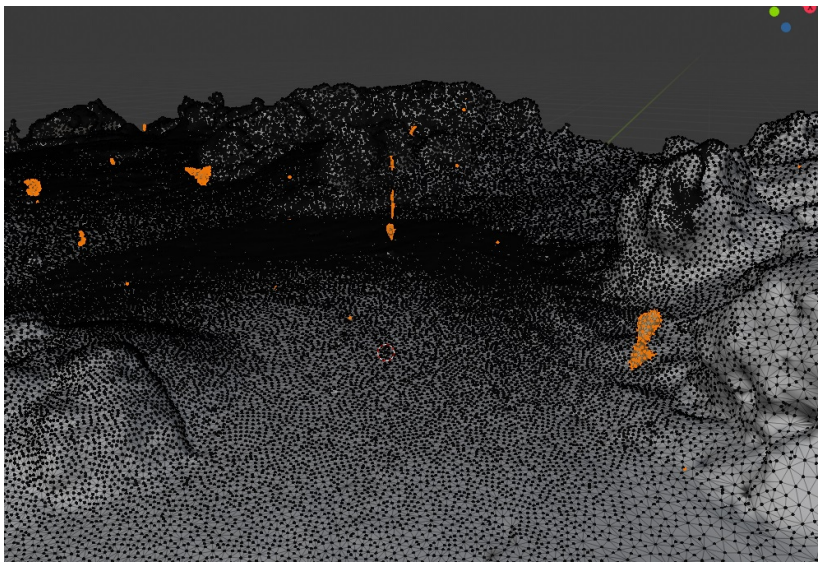


Figure 66: Selecting all 'Floating' Artefacts in a Reconstructed Model.

Other useful tools in Sculpt mode include 'Flatten' which can be used to bring vertices within the brush area towards the average vertex height and the 'Fill' and 'Deepen' tools which will modify vertices below the average height. 'Fill' will take vertices up towards the average and 'Deepen' will move them further downwards from the average.

The smoothing tools provided in Edit mode are more precise than those of Sculpt mode and can typically be used for small areas or for further surface refinement after an initial smoothing using Sculpt mode. The 'Smooth Vertices' command (in the Vertex menu) will smooth the surface defined by previously selected vertices, with a Smoothing factor and a Repeat number used to modify the degree of smoothing (see Figure 68).

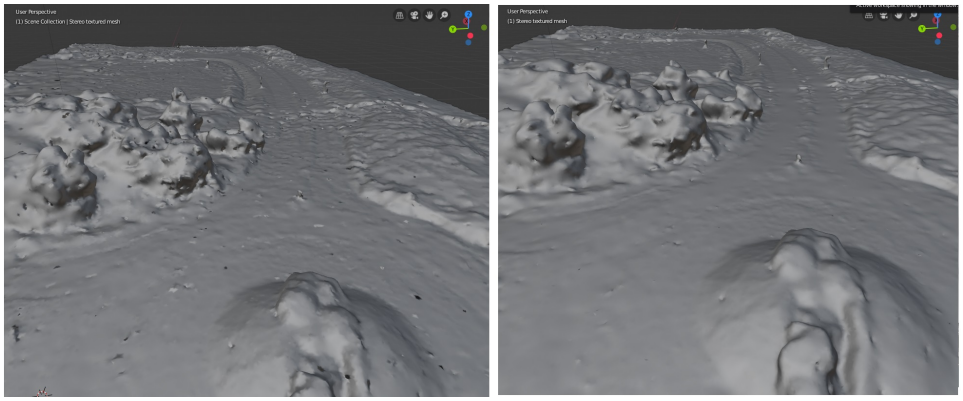


Figure 67: Use of the ‘Smooth’ Tool in Blender. Model of roundabout before use of the ‘Smooth’ tool (left) and after (right).

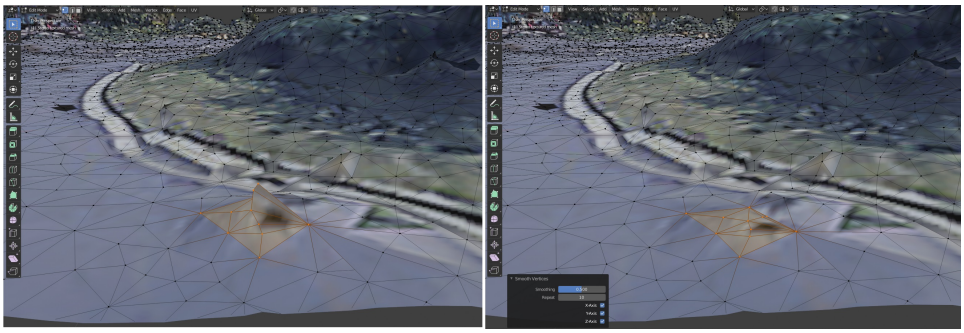


Figure 68: Use of the ‘Smooth Vertices’ command in Blender. Model before ‘Smooth Vertices’ operation (left) and after (right).

When using the ‘Smooth Vertices’ tool it is important to select all vertices within the area to be smoothed, but if there are deep pits or holes in the area this may be difficult using the area selection methods (e.g. circle or box select) whilst looking down on the surface from above. It may be necessary to change to a viewing position below the surface to select the vertices at the bottom of these deep holes (as shown in Figure 69).

C.3 Directly Modifying the Object Mesh

An object mesh produced using photogrammetry may contain areas of high distortion and irregularity. Such areas will often contain very small faces or thin faces having one or two very small internal angles. Tools such as ‘Smooth Vertices’ may not work well in areas with these problems and it may be necessary to delete and rebuild the area manually using the Edit mode of the 3D Editor. Figure 70 on page 149 shows a water surface that has been smoothed but still contains spikes where the mesh is highly distorted.

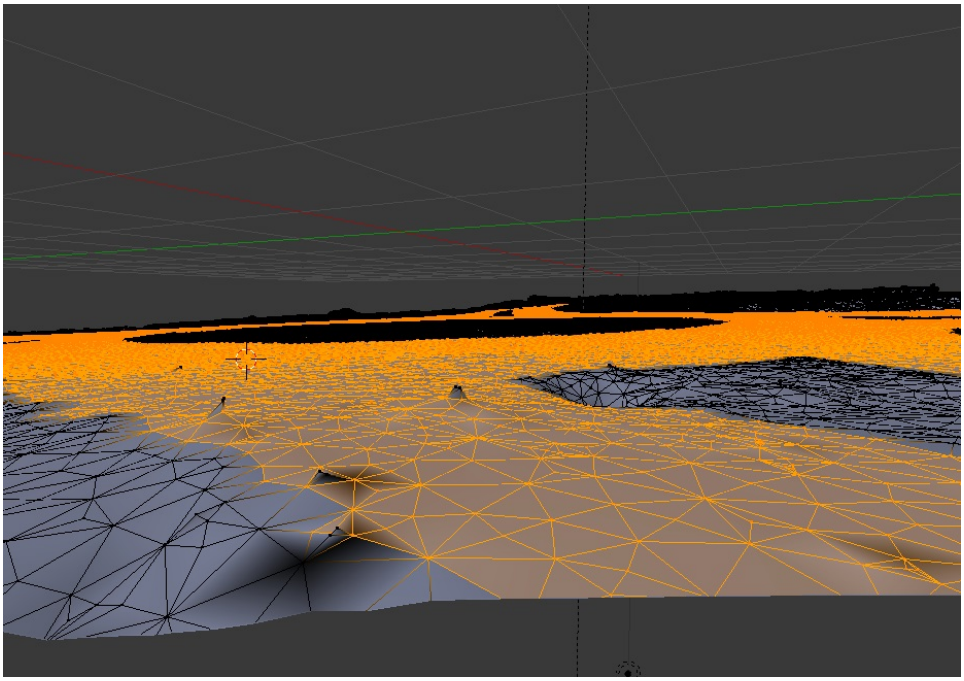


Figure 69: Deep Holes in a Reconstructed Model. Vertices in deep holes visible when looking from beneath the surface.

Figure 71 shows that after one of the spikes is deleted a hole is left in the water surface, which can be filled by merging vertices. Two 'Merge Vertices' operations (each with two selected vertices and merging at their centre) were carried out to fill the hole as shown in Figure 72 on page 150.

It would also be possible to fill the hole by creating faces using the 'Fill' command (from the Face menu). This method has the disadvantage that the UV mapping of this newly defined geometry will need to be created so that it can be painted with the correct colour using Texture Paint mode. For large areas merging may not be appropriate because the size of mesh faces may become too big to adequately model surface shape. In this case creating new geometry may give better results.

Other useful commands for modifying a selected portion of the mesh include 'Move', 'Rotate', 'Scale' and 'Push/Pull' (from the Mesh, Transform menu). 'Push/Pull' can be used to move selected mesh elements towards (Push) or away (Pull) from the centre point of the elements. The 'Knife' tool can be used to divide surfaces by splitting mesh faces along a line. There are also various extrude tools such as 'Extrude Region' which will create new geometry by offsetting a region and creating walls along the boundary to join it with the original geometry.

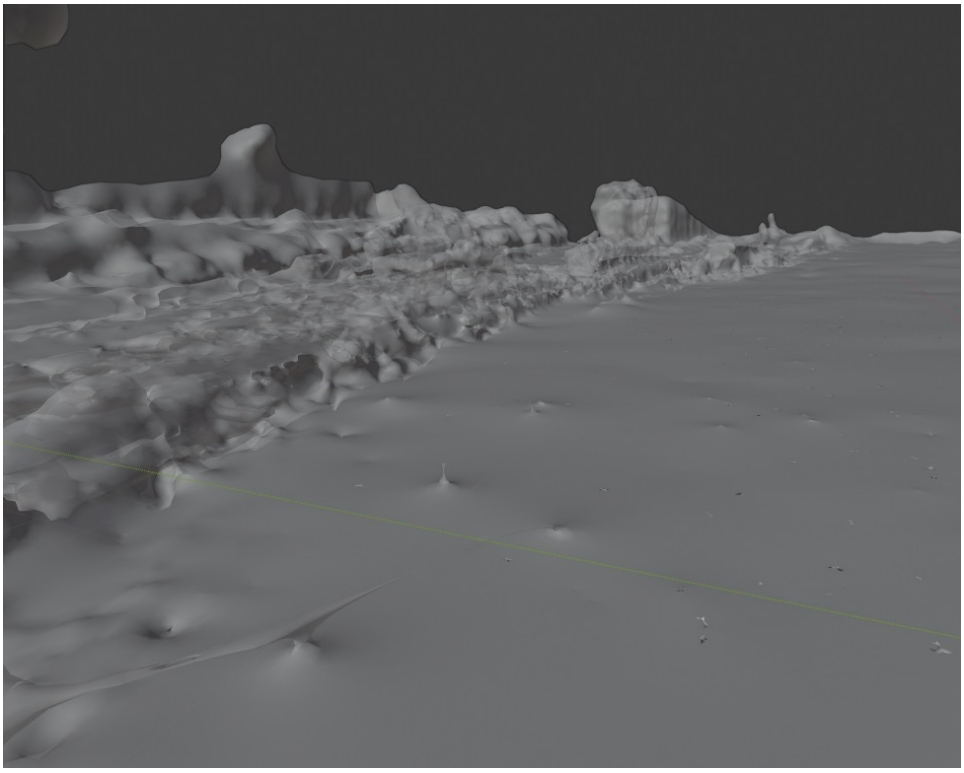


Figure 70: Distortions in Surfaces after Smoothing. Water surface containing distortions after smoothing operation.

.

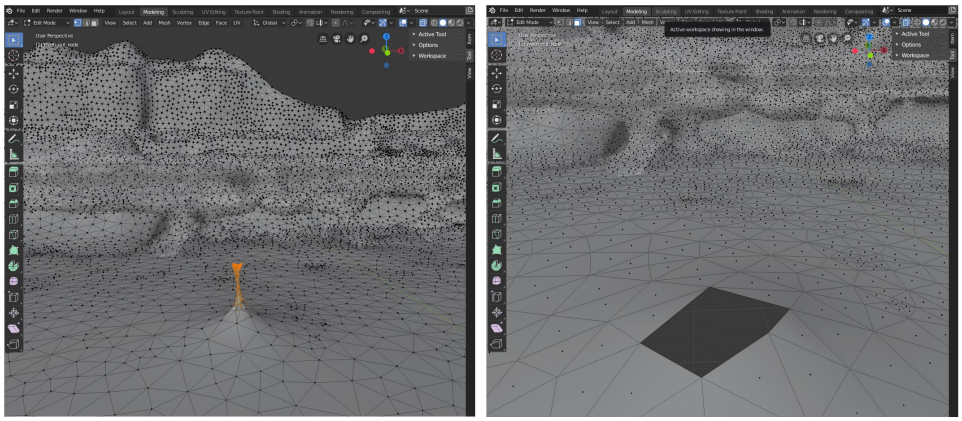


Figure 71: Deleting a Spike on a Surface. Selecting and deleting the vertices of a spike (left) leaves a hole in the surface (right).

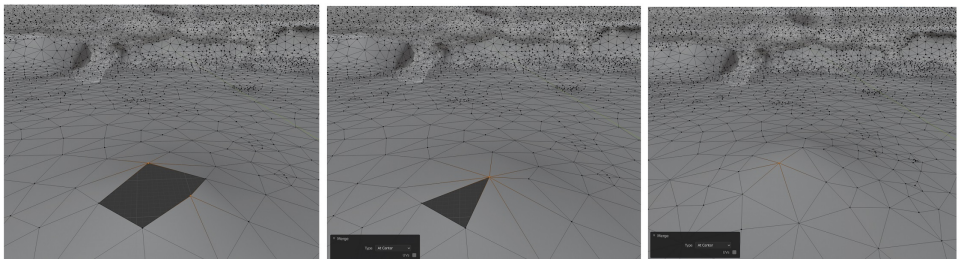


Figure 72: Merging Vertices to Fill a Surface Hole.

C.4 Modifying Surface Textures in Texture Paint Mode

Distortions in a model produced using photogrammetry are often most apparent when observing the surface texture rather than the overall surface shape. This is especially true for models of urban environments where distortions in road markings or in the texture applied to a building wall are very obvious because of viewer expectations. In contrast, a good quality texture projected onto moderately distorted geometry can help to mask the apparent degree of distortion in the surface shape. Correcting geometry (e.g. by surface smoothing) can increase the degree of apparent distortion in the texture, since the final mesh produced by the photogrammetry software will be textured to give results that closely match the 2D image data source with the uncorrected geometry.

The Texture Paint mode of the 3D Editor provides tools such as 'Draw', 'Soften', 'Smear' and 'Clone' which can be used to improve the quality of the texture applied on a object mesh. The 'Smear' tool can be used to give surfaces a more uniform texture and remove flaws. Care must be taken to ensure surfaces such as roads or grass do not become too uniform and hence less realistic. The 'Draw' tool can be used to paint directly on the surface with a user defined paint colour, which can be selected using a colour picker to match the colour at a particular surface point. Figure 73 shows the result of using this tool on the roundabout model to paint over the original distorted road markings and repaint new markings in their place. The shadows in the original texture have also been painted over. The remnants of the objects casting the shadows were deleted in Blender since they were not reconstructed correctly by Google Earth. In general all shadows (not just those from deleted objects) should be painted over since simulation environments such as Unreal Engine will usually (by default) generate shadows on objects according to the lighting setup.

The paint process automatically updates an internal image used to store the texture and this can be saved to a file using the Image Editor.

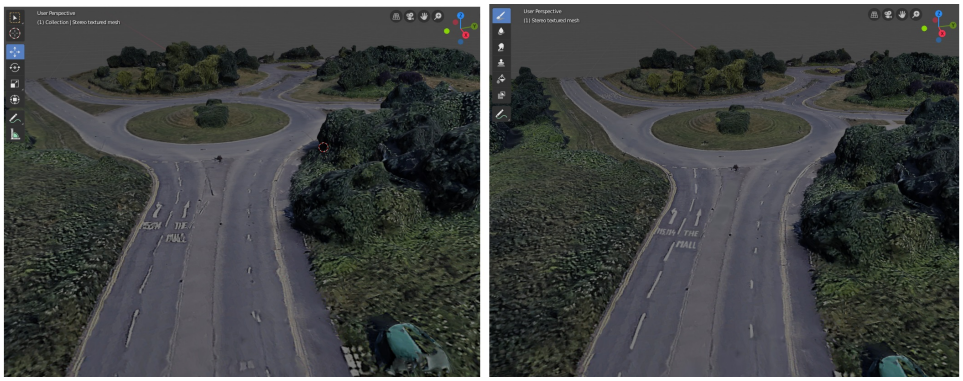


Figure 73: Painting on a Surface using the 'Draw' Tool. Original texture (left) and modified texture (right) with road markings repainted and shadows removed.

C.5 Editing the UV Map

In some circumstances (e.g. to use texture painting on newly created geometry) it may be necessary to edit the UV mapping for the object mesh. Figure 74 shows a model of the Wills Memorial Building produced using photogrammetry from Google Earth imagery. There are highly distorted areas on the main road due to moving vehicles (e.g. the selected portion of the mesh in the right-hand image). Figure 75 shows the mesh after one of these distorted areas has been deleted (left) and after it has been recreated using a 'Fill' operation in Edit mode (right).

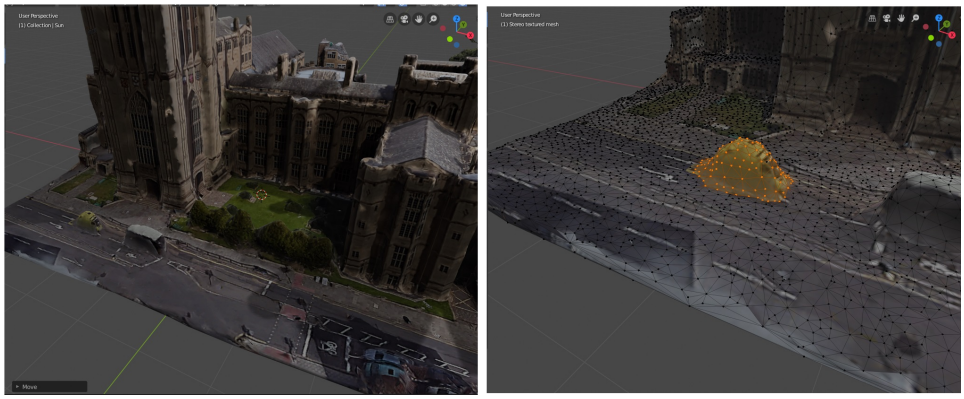


Figure 74: Distorted Mesh Due to a Moving Vehicle. Model of the Wills Memorial Building (left) and distorted mesh area due to a moving vehicle (right).

The UV Editor can be used to display the UV mapping of selected mesh areas. Figure 76 shows that the original geometry (i.e. the object mesh excluding the newly created fill area) does not use an area in the upper left

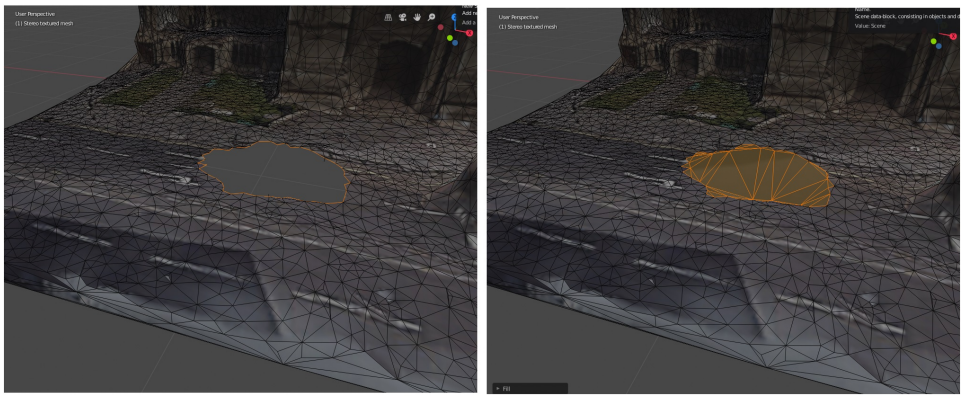


Figure 75: Repairing Distorted Geometry. Selected mesh deleted (left) and recreated using 'Fill' (right).

of the texture image for mappings, and hence this can be used for the UV map of the new fill geometry. The 'Unwrap' command of the UV Editor is used to create a UV mapping for the new geometry as shown in Figure 77, but by default this mapping will use nearly the entire texture, including areas used by the original geometry.

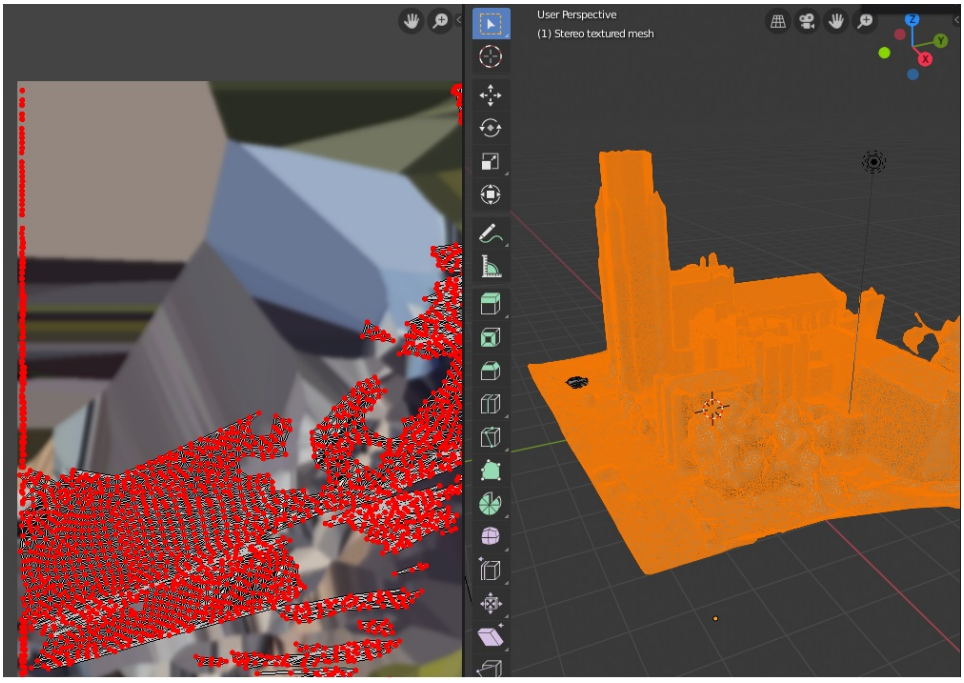


Figure 76: UV Maps. UV map (left) for original geometry of Wills Memorial Building (right).

By selecting the vertices defining the texture mapping in the UV Editor it can be scaled and moved to a part of the texture image that is not used for other mappings (see Figure 78). This will ensure that painting the new geometry will not affect other areas of the mesh. Figure 79 shows the textured mesh using the new UV map (left image) and the result of texture painting to fill in the missing areas of the road and pavement (right image).

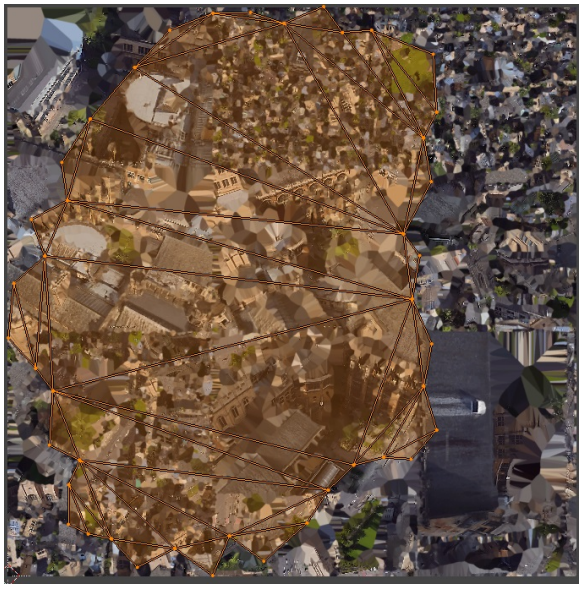


Figure 77: Creating a UV Map for New Geometry using 'Unwrap'. New geometry unwrapped onto texture image to create a UV mapping.

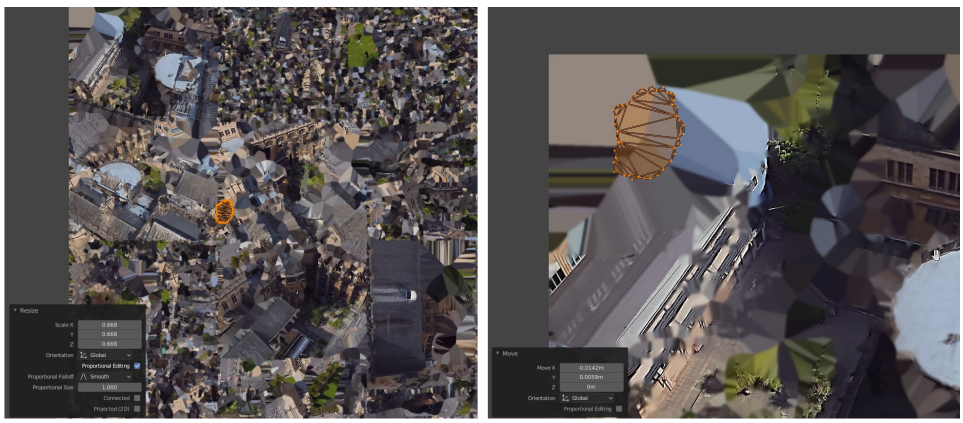


Figure 78: Editing a UV Map. Scaling and moving the UV map for new geometry to an unused part of the texture image.

Figure 79: Painting Newly Created Geometry. Textured mesh using edited UV Map for new geometry, before texture painting (left) and after (right).

D Scan Optimization Class Methods and Program Functions

D.1 Class Methods

D.1.1 Class Model_Object

__init__(self, blend_obj): This function is automatically called on object creation. It initialises the object using data from the Blender object given by parameter blend_obj.

cull_backface_polygons(self, cam_vec): The list of front-facing polygons for the object, *vis_polygons*, is updated, using a camera direction specified by parameter cam_vec.

D.1.2 Class Camera

__init__(self, x_pos, y_pos, z_pos, pitch, yaw, fl, sx, sy): Initializes camera properties. The function is automatically called on object creation.

set_params(self, fl, sx, sy): Parameters fl, sx and sy are used to set the focal length and screen size for the camera.

move_inc(self, x_inc, y_inc, z_inc): The camera moves distances in the x,y and z directions given by parameters x_inc, y_inc and z_inc. The transformation matrix property of the camera is updated.

move_to(self, x_pos, y_pos, z_pos): Moves the camera to the global coordinate position given by parameters x_pos, y_pos and z_pos. The transformation matrix property of the camera is updated.

set_rotation(self, pitch, yaw): Parameters pitch and yaw are used to set the desired camera angle. The transformation matrix and camera axes properties of the camera are updated.

visible_on_screen(self, obj, idx): The function returns a value of True if the vertex with index idx, in object obj, is visible on the camera screen, otherwise returning a value of False.

D.1.3 Class Hemisphere

__init__(self, vert_co, norm): Initializes a *Hemisphere* object for a vertex with coordinates and normals given by vector parameters vert_co and norm. An array of points, *hemi_locs*, uniformly distributed over the hemisphere is created and all coverage data in *hemi_values* is initialized to a value of 0.

set_hemi_values(self, cam_pos): Updates the *Hemisphere* coverage data for a view from a camera with a position given by vector parameter cam_pos. Sampling points in *hemi_locs*, within the disk centred on the projection of cam_pos on to the hemisphere, with size defined by equation 44 (given on page 88), have their corresponding coverage value in array *hemi_values* set to 1.

calc_hemi_metric(self): Returns a value estimating the quality of reconstruction for the vertex corresponding to the *Hemisphere* object. The function returns the (un-scaled) coverage for a vertex j as given by equation 47 (given on page 89), with the $v_j(\mathbf{h}_k)$ for each sampling point k on the hemisphere stored in coverage array *hemi_values*:

$$F(C_j) = \sum_{k=1}^{K} w_j(\mathbf{h}_k)\, v_j(\mathbf{h}_k)$$

reset_hemi_values(self): Resets the coverage data for sampling points on the *Hemisphere*. All elements of array *hemi_values* are set to a value of 0. This is typically used before performing a new scan to initialise the viewpoint coverage information.

D.2 Main Program Functions

Global parameter *model_objs* is a list of objects of type *Model_Object*, each of which corresponds to a Blender object of type 'MESH' (used to model the environment).

calc_poly_visibility(obj_list, pos): Parameter pos is a vector defining the camera position and obj_list is a list of *Model_Object*. A back-face culling algorithm is used to update the list of front facing polygons (stored in *Model_Object* class property *vis_polygons*) for each *Model_Object* in obj_list.

vertex_visible(obj, idx, cam_pos): Parameter obj is an object of class *Model_Object*, idx is the index of a vertex in obj and cam_pos is a vector defining a camera position. The function returns a value of 'True' if the

vertex is visible from a camera positioned at cam_pos, otherwise returning 'False'. For each environment object in list *model_objs*, all front facing polygons of the object (stored in list *vis_polygons*) are checked to determine if the plane of the polygon intersects between the vertex and the camera. If there is such an intersection the polygon is checked to see if it overlaps the vertex, in which case the function returns 'False'. If no polygons overlap the vertex the function returns 'True'.

update_coverage(obj_list, cam):
Parameter obj_list is a list of *Model_Object* and cam specifies a *Camera* object. The function updates the accumulated view coverage data for the vertices of objects in list obj_list, adding extra coverage due to cam. The function will call the **vertex_visible** function for each vertex of every object in obj_list. If a vertex is visible from the position of cam it will call the **visible_on_screen** function of cam to determine if the image of the vertex falls within the limits of the camera screen, and if so update the coverage for the vertex by calling the **set_hemi_values** function of the corresponding *Hemisphere*.

calculate_metric(obj_list): Parameter obj_list is a list of *Model_Object*. The function calculates the average reconstruction metric for all vertices of objects in obj_list. The metric used is a scaled version of the coverage metric used by Roberts *et al.* [60], given in equation 47 (see page 89). The scale is chosen so that the maximum metric value of 1.0 represents every vertex having coverage from all possible directions. To calculate the metric the function will call the **calc_hemi_metric** function for all *Hemisphere* objects belonging to the objects in list obj_list.

scan_x_dir(cam, x_start, x_end, x_steps, y_start, y_end, y_steps, z): The function performs a rectangular scan at a height z, with scan lines directed alternately in the +x and -x directions. The scan starts from coordinate (x_start, y_start) and ends at (x_end, y_end). The cross-track separation distance is given by (y_end − y_start)/y_steps and the in-track distance (between successive camera shots) is given by (x_end − x_start)/x_steps. After each movement of the camera the functions **calc_poly_visibility** and **update_coverage** are called to update the list of front facing polygons and recalculate the accumulated coverage.

set_best_hemi_values(obj_list):
Parameter obj_list is a list of *Model_Object*. This function is called during the optimization of a level when the reconstruction metric resulting from a scan exceeds the previous highest value. For all *Hemisphere* objects belonging to objects in obj_list, the data in *hemi_values* is copied to *hemi_values_best*.

save_best_hemi_values(obj_list):
Parameter obj_list is a list of *Model_Object*. The function is called after the optimization of a level to save the optimum coverage data. For all *Hemisphere* objects belonging to objects in obj_list, the data in *hemi_values_best* is copied to *hemi_values_saved*.

set_to_saved_hemi_values(obj_list):
Parameter obj_list is a list of *Model_Object*. The function is called at the beginning of each trial scan during level optimization to initialize the coverage data to that at the optimum of the previous level. For all *Hemisphere* objects belonging to objects in obj_list, the data in *hemi_values_saved* is copied to *hemi_values*.

References

[1] A. Charlton, *Sochi Drone Shooting Olympic TV, Not Terrorists*, Associated Press, Feb. 2014. [Online]. Available: http://wintergames.ap.org/article/sochi-drone-shooting-olympic-tv-not-terrorists.

[2] Live Production, *Review Sochi 2014: Broadcasting the Magic of the Games Across the World*, Feb. 2014. [Online]. Available: http://www.live-production.tv/news/sports/review-sochi-2014-broadcasting-magic-games-across-world.html.

[3] K. Gallagher, *How Drones Powered Rio's Olympic Coverage*, The Simulyze Blog, Aug. 2016. [Online]. Available: http://www.simulyze.com/blog/how-drones-powered-rios-olympic-coverage.

[4] Max Goldbart, *Attenborough: Drones are 'game-changers'*, Feb. 2019. [Online]. Available: https://www.broadcastnow.co.uk/bbc/attenborough-drones-are-game-changers/5137078.article.

[5] NBC NEWS, *New Drone Video Captures Scale of Haiti Hurricane Damage*, Oct. 2016. [Online]. Available: http://www.nbcnews.com/video/new-drone-video-captures-scale-of-haiti-hurricane-damage-784114243853.

[6] MultiDrone. (2020). MultiDrone - Using multiple drones for media production, [Online]. Available: https://multidrone.eu/.

[7] Red Dot Drone PTE LTD. (2021). Red Dot Drone, [Online]. Available: https://reddotdrone.com/.

[8] S. Boyle, F. Zhang, and D. R. Bull, "A Subjective Study of the Viewing Experience for Drone Videos," in *2019 IEEE International Conference on Image Processing (ICIP)*, (Taipei, Taiwan, Sep. 22–25, 2019), IEEE, 2019, pp. 1034–1038. DOI: 10.1109/ICIP.2019.8803747.

[9] S. Boyle, M. Newton, F. Zhang, and D. Bull, "Environment Capture and Simulation for UAV Cinematograhy Planning and Training," in *European Signal Processing Conference, Satellite Workshop: Signal Processing, Computer Vision and Deep Learning for Autonomous Systems*, 2019. [Online]. Available: https://research-information.bris.ac.uk/ws/portalfiles/portal/199953761/Environment_Capture_and_Simulation_for_UAV_Cinematography_Planning_and_Training.pdf.

[10] F. Zhang, D. Hall, T. Xu, S. Boyle, and D. Bull, "A Simulation Environment for Drone Cinematography," *IBC Technical Papers*, 2020. [Online]. Available: https://www.ibc.org/technical-papers/a-simulation-environment-for-drone-cinematography/6747.article.

[11] J. V. Mascelli, *The Five C's of Cinematography*. W. Hollywood, California, USA: Silman-James Press, 1965.

[12] Film Riot. (2021). Basic Camera Shots to Improve Your Films, [Online]. Available: https://www.youtube.com/watch?v=-s57Na6Nx_c.

[13] ——, (2011). Camera Techniques for Better Filmmaking! [Online]. Available: https://www.youtube.com/watch?v=CYPrtXZ7HVE.

[14] D. Arijon, *Grammar of the Film Language*. London, UK: Focal Press, 1976.

[15] Film Riot. (2014). Quick Tips: Understanding the 180 Degree Rule! [Online]. Available: https://www.youtube.com/watch?v=Bba7raSvvRo.

[16] Learn Online Video. (2017). The Rule of Thirds | What is it? [Online]. Available: https://www.youtube.com/watch?v=A7wnhDKyBuM.

[17] T.Nägeli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, "Real-time Motion Planning for Aerial Videography with Dynamic Obstacle Avoidance and Viewpoint Optimization," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1696–1703, 2017.

[18] C. Gebhardt, B. Hepp, T.Nägeli, S.Stevsić, and O.Hilliges, "AirWays: Optimization-Based Planning of Quadrotor Trajectories according to High-Level User Goals," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, May 2016, pp. 2508–2519, ISBN: 9781450333627. DOI: 10.1145/2858036.2858353.

[19] L. He, M. Cohen, and D. Salesin, "The Virtual Cinematographer: A Paradigm for Automatic Real-Time Camera Control and Directing," in *SIGGRAPH '96 Proceedings of the 23rd annual conference on Computer Graphics and Interactive Techniques*, New York, NY, USA: ACM, 1996, pp. 217–224, ISBN: 0-89791-746-4. DOI: 10.1145/237170.237259.

[20] COPTRZ. (2021). Drones in Filmmaking – The best drones for the job, [Online]. Available: https://coptrz.com/drones-in-filmmaking-the-best-drones-for-the-job/.

[21] Los Angeles Times. (2015). Drones are providing film and TV viewers a new perspective on the action, [Online]. Available: https://www.latimes.com/entertainment/envelope/cotown/la-et-ct-drones-hollywood-20151008-story.html.

[22] N. Joubert, M. Roberts, A.Truong, F. Berthouzoz, and P. Hanrahan, "An Interactive Tool for Designing Quadrotor Camera Shots," *ACM Transactions on Graphics*, vol. 34, no. 6, pp. 1–11, 2015.

[23] N. Joubert, J. L. E, D. B. Goldman, F. Berthouzoz, M. Roberts, J. A. Landay, and P. Hanrahan, "Towards a Drone Cinematographer: Guiding Quadrotor Cameras using Visual Composition Principles," *ArXiv e-prints*, Oct. 2016. arXiv: 1610.01691 [cs.GR].

[24] M. Christie, P. Olivier, and J.-M. Normand, "Camera Control in Computer Graphics," *Computer Graphics Forum*, vol. 27, no. 8, pp. 2197–2218, 2008. DOI: 10.1111/j.1467-8659.2008.01181.x.

[25] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, UK: Cambridge University Press, 2003, ISBN: 978-0-521-54051-3.

[26] SZ DJI Technology Co. (2019). Film Like a Pro: How to Create Cinematic Orbit Shots, [Online]. Available: https://store.dji.com/guides/film-like-a-pro-how-to-create-cinematic-orbit-shots/.

[27] *Unreal Engine 4 Documentation*. Epic Games. [Online]. Available: https://docs.unrealengine.com/latest/INT/.

[28] F. Moss, K. Wang, F. Zhang, R. Baddeley, and D. Bull, "On the Optimal Presentation Duration for Subjective Video Quality Assessment," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 11, pp. 1977–1987, 2016.

[29] DJI. (2020). DJI Flight Simulator, [Online]. Available: https://www.dji.com/uk/simulator.

[30] Microsoft Research. (2018). Welcome to AirSim, [Online]. Available: https://microsoft.github.io/AirSim/.

[31] Microsoft Corporation. (2020). Microsoft Flight Simulator, [Online]. Available: https://www.xbox.com/en-GB/games/microsoft-flight-simulator.

[32] Google. (2020). Google Earth Studio, [Online]. Available: https://www.google.com/intl/en_uk/earth/studio/.

[33] Dillon Skiffington. (2020). A Complete List of All Photorealistic Cities in Microsoft Flight Simulator 2020, [Online]. Available: https://www.fanbyte.com/guides/a-complete-list-of-all-photorealistic-cities-in-microsoft-flight-simulator-2020/.

[34] BBC. (2021). Israel-Gaza: Why is the region blurry on Google Maps? [Online]. Available: https://www.bbc.co.uk/news/57102499.

[35] OpenStreetMap Foundation. (2020). OpenStreetMap, [Online]. Available: https://www.openstreetmap.org/.

[36] OSM Buildings. (2020). OSM Buildings, [Online]. Available: https://osmbuildings.org/.

[37] prochitecture. (2020). blender-osm: OpenStreetMap and Terrain for Blender, [Online]. Available: https://gumroad.com/l/blosm.

[38] Ordnance Survey Limited. (2020). Ordnance Survey — See A Better Place, [Online]. Available: https://www.ordnancesurvey.co.uk/.

[39] OpenTopography.org. (2018). OpenTopography Website, [Online]. Available: https://www.opentopography.org/.

[40] USGS. (2018). EarthExplorer Website, [Online]. Available: https://earthexplorer.usgs.gov/.

[41] NASA. (2020). Landsat Science, [Online]. Available: https://landsat.gsfc.nasa.gov/.

[42] NTT Data Corporation. (2018). AW3D Website, [Online]. Available: https://www.aw3d.jp/en/.

[43] B. Sirmacek and C. Unsalan, "Urban-Area and Building Detection Using Sift Keypoints and Graph Theory," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 4, pp. 1156–1167, 2009.

[44] C. Lin and R. Nevatia, "Building Detection and Description from a Single Intensity Image," *Computer Vision and Image Understanding*, vol. 72, no. 2, pp. 101–121, 1998.

[45] M. Fradkin, H. Maıtre, and M.Roux, "Building Detection from Multiple Aerial Images in Dense Urban Areas," *Computer Vision and Image Understanding*, vol. 82, no. 3, pp. 181–207, 2001.

[46] V. Verma, R. Kumar, and S. Hsu, "3D Building Detection and Modeling from Aerial LIDAR Data," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, New York, NY, USA: IEEE Conference Publications, 2006, pp. 2213–2220, ISBN: 0-7695-2597-0. DOI: 10.1109/CVPR.2006.12.

[47] J. Shan, Z. Hu, P. Tao, L. Wang, S. Zhang, and S. Ji, "Toward a unified theoretical framework for photogrammetry," *Geo-spatial Information Science*, vol. 23, no. 1, pp. 75–86, 2020. DOI: 10.1080/10095020.2020.1730712. eprint: https://doi.org/10.1080/10095020.2020.1730712. [Online]. Available: https://doi.org/10.1080/10095020.2020.1730712.

[48] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004. DOI: doi:10.1023/B:VISI.0000029664.99615.94.

[49] R. Gherardi, R. Toldo, V. Garro, and A. Fusiello, "Automatic camera orientation and structure recovery with samantha," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXVIII-5/W16, pp. 261–268, 2011. DOI: 10.5194/isprsarchives-XXXVIII-5-W16-261-2011. [Online]. Available: https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XXXVIII-5-W16/261/2011/.

[50] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon, "Bundle Adjustment - A Modern Synthesis," in *Vision Algorithms : Theory and Practice : International Workshop on Vision Algorithms, Corfu, Greece, September 21-22, 1999 : Proceedings*, Springer, 1999, pp. 298–372. DOI: 10.1007/3-540-44480-7.

[51] 3Dflow. (2018). 3DF Zephyr Website, [Online]. Available: https://www.3dflow.net/.

[52] E. Galceran and M. Carreras, "A Survey on Coverage Path Planning for Robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013. DOI: 10.1016/j.robot.2013.09.004.

[53] L. Heng, D. Honegger, G. Lee, L. Meier, P. Tanskanen, M. Pollefeys, and F. Fraundorfer, "Autonomous Visual Mapping and Exploration with a Micro Aerial Vehicle," *Journal of Field Robotics*, vol. 31, no. 4, pp. 654–675, 2014. DOI: 10.1002/rob.21520.

[54] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding Horizon "Next-Best-View" Planner for 3d Exploration," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1462–1468. DOI: 10.1109/ICRA.2016.7487281.

[55] B. Charrow, G. Kahn, S. Patil, S. Liu, K. Goldberg, P. Abbeel, M. N., and V. Kumar, "Information-Theoretic Planning with Trajectory Optimization for Dense 3d Mapping," in *2015 Robotics: Science and Systems Conference, RSS 2015*, 2015. DOI: 10.15607/RSS.2015.XI.003.

[56] P. Wang, R. Krishnamurti, and K. Gupta, "View planning problem with combined view and traveling cost," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 711–716. DOI: 10.1109/ROBOT.2007.363070.

[57] M. Mauro, H. Riemenschneider, A. Signoroni, R. Leonardi, and L. Van Gool, "A unified framework for content-aware view selection and planning through view importance," *BMVC 2014*, 2014.

[58] A. Hornung, B. Zeng, and L. Kobbelt, "Image Selection for Improved Multi-View Stereo," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008. DOI: 10.1109/CVPR.2008.4587688.

[59] K. Schmid, H. Hirschmèuller, A. Dèomel, I. Grixa, M. Suppa, and G. Hirzinger, "View Planning for Multi-View Stereo 3d Reconstruction Using an Autonomous Multicopter," *Journal of Intelligent and Robotic Systems*, vol. 65, no. 1-4, pp. 309–323, 2012.

[60] M. Roberts, D. Dey, A. Truong, S. Sinha, S. Shah, A. Kapoor, P. Hanrahan, and N. Joshi, "Submodular Trajectory Optimization for Aerial 3D Scanning," *arXiv e-prints*, arXiv:1705.00703, arXiv:1705.00703, May 2017. arXiv: 1705.00703 [cs.CV].

[61] N. Smith, N. Moehrle, M. Goesele, and W. Heidrich, "Aerial Path Planning for Urban Scene Reconstruction: A Continuous Optimization Method and Benchmark," *ACM Transactions on Graphics*, vol. 37, no. 6, pp. 1–15, 2019. DOI: 10.1145/3272127.3275.

[62] T. Xu, "A simulation framework for drone cinematography," Master's thesis, University of Bristol, Sep. 2019.

[63] Edmund Optics. (2021). Understanding Focal Length and Field of View, [Online]. Available: https://www.edmundoptics.co.uk/knowledge-center/application-notes/imaging/understanding-focal-length-and-field-of-view/.

[64] N. Smith, N. Moehrle, M. Goesele, and W. Heidrich, "Aerial Path Planning for Urban Scene Reconstruction: A Continuous Optimization Method and Benchmark," *ACM Transactions on Graphics*, vol. 37, no. 6, pp. 1–15, 2018. DOI: 10.1145/3272127.3275010.

[65] J. Foster. (2016). Four Steps for Making an Excellent 3D Model With a Drone, [Online]. Available: https://medium.com/aerial-acuity/4-steps-for-making-an-excellent-3d-model-with-a-drone-25dc35f1df62.

[66] SPH Engineering. (2018). Comparing Precision of Autopilots for Survey Missions, [Online]. Available: https://www.ugcs.com/news-entry/comparing-precision-of-autopilots-for-survey-missions.

[67] diydrones. (2019). Comparing Precision of Autopilots for Survey Missions-The Results, [Online]. Available: https://diydrones.com/profiles/blogs/comparing-precision-of-autopilots-for-survey-missions-the-results.

[68] Blender Foundation. (2020). Tips and Tricks - Blender Python API, [Online]. Available: https://docs.blender.org/api/blender2.8/info_tips_and_tricks.html#bundled-python-extensions.

[69] Continuum Analytics. (2014). Writing Cuda Python, [Online]. Available: https://numba.pydata.org/numba-doc/0.13/CUDAJit.html.