DISSERTATION

CONTINUITY OF OBJECT TRACKING

Submitted by

Haney W. Williams

Department of Systems Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2022

Doctoral Committee:

Advisor: Steven J. Simske

Mahmood R. Azimi-Sadjadi
Edwin K. P. Chong
J. Ross Beveridge

ABSTRACT

CONTINUITY OF OBJECT TRACKING

The demand for object tracking (OT) applications has been increasing for the past few decades in many areas of interest: security, surveillance, intelligence gathering, and reconnaissance. Lately, newly-defined requirements for unmanned vehicles have enhanced the interest in OT. Advancements in machine learning, data analytics, and deep learning have facilitated the recognition and tracking of objects of interest; however, continuous tracking is currently a problem of interest to many research projects. This dissertation presents a system implementing a means to continuously track an object and predict its trajectory based on its previous pathway, even when the object is partially or fully concealed for a period of time. The system is divided into two phases: The first phase exploits a single fixed camera system and the second phase is composed of a mesh of multiple fixed cameras. The first phase system is composed of six main subsystems: Image Processing, Detection Algorithm, Image Subtractor, Image Tracking, Tracking Predictor, and the Feedback Analyzer. The second phase of the system adds two main subsystems: Coordination Manager and Camera Controller Manager. Combined, these systems allow for reasonable object continuity in the face of object concealment.

ACKNOWLEDGMENTS

I would like to thank Dr. Steven Simske, professor, mentor, friend and dissertation advisor extraordinaire. Without his help, advice, expertise and encouragement this research and dissertation would not have happened. I also would like to thank the members of my dissertation committee: Dr. Mahmood Azimi-Sadjadi, Dr. Edwin Chong and Dr. Ross Beveridge. Their insight, feedback and advice were influential and essential throughout my dissertation process.

A debt of gratitude to my wife, my daughter, my father, my late mother who passed away earlier this year, my sisters and my family in-law who encouraged me throughout the toughest times to continue my education and reach my educational dreams. They also were able to put up with a busy and absentee husband, father and son most of the time throughout the program.

I would like to thank my friends, coworkers and especially Father Gregory Bishay for guiding me with ideas throughout this dissertation. Without the support of all the people mentioned above none of my accomplishments would have happened.

I appreciate Colorado State University Systems Engineering department for offering this remote PhD program which allowed professional individuals like myself to reach their educational goals. A special thanks to Ingrid Bridge for all the help she offers to all the students with the process.

Last but not least, I would like to thank the Boeing Company for offering the learning together program to their employees to reach their dream.

TABLE OF CONTENTS

ABSTRACT ii
ACKNOWLEDGMENTS iii
LIST OF TABLES ix
LIST OF FIGURES x
LIST OF KEYWORDS xiii
CHAPTER 1 – SYSTEM ENGINEERING METHODOLOGY 1
Introduction 1
Needs Analysis 1
Operational Analysis 2
Analyze Projected Needs 2
Define the Operational Approach: 3
Functional Analysis 4
Translation into Functions 4
Allocation of Function into Subsystems 4
Feasibility Definition 6
Visualize Subsystem Technology 6
Define Feasibility Concept 9
Needs Validation 10
Design Effectiveness Model 10
Validate Needs and Feasibility 11
Concept Exploration 12
Operational Requirement Analysis 12
Analyzing Operation Objectives 12
Refining Operations Requirements 13
Performance Requirement Formulation 13
Derive Subsystem Functions 13
Formulate Performance Parameters 15
Implementation of Concept Exploration 17
Explore Implementation Concepts 17

Define Performance Characteristics 17
Performance Requirements Validation 18
Integrate Performance Characteristics 19
Validate Performance Requirements 19
Concept Definition 19
Performance Requirements Analysis 19
Analyze Performance Requirements 20
Refine Performance Requirements 20
Functional Analysis and Formulation 20
Define Components Functions 20
Formulate Functional Requirements 21
Concept Selection 22
Synthesize Alternative Concepts 22
Select Preferred Concept 23
Concept Validation 24
Conduct System Simulation 24
Validate Selected Concept 24
Risk Management 24
Project Management Related Risks 24
Qualitative Risk Assessment 25
Advanced Development 27
Requirements Analysis 28
Analyze System Functional Specs 28
Identify Immature Components 29
Functional Analysis and Design 29
Identity Functional Performance Issues 29
Resolve Issues Design Software 30
Prototype Development 30
Identify Unproven Technology 30
Design and Build Critical Components 32
Development Testing 32
Build Test Setup 33

Evaluate Test Results 33
Software Systems 35
Engineering Design 37
Integration and Evaluation 38
Production 44
Operation and Support 44
CHAPTER 2 – SYSTEM OVERVIEW AND ARCHITECTURE 46
Introduction 46
System Overview 47
Image Processor Subsystem 49
Background Subtraction Subsystem 50
Detection Subsystem 51
Object Tracking Subsystem 52
Trajectory Predictor Subsystem 53
Feedback Analyzer Subsystem 54
CHAPTER 3 – OBJECT RECOGNITION USING DEEP NEURAL NETWORK 55
Introduction 55
Stanford Cars Dataset 57
Approach 58
Image Processing and Enhancement for Training 58
Data Augmentation 60
Learning Algorithm Architecture 60
VGG16 Convolutional Neural Network 63
VGG16 Architecture 63
VGG16 Results 66
VGG19 Convolutional Neural Network 67
VGG19 Architecture 67
VGG19 Results 68
DenseNet201 Convolutional Neural Network 69
DenseNet201 Architecture 69
DenseNet201 Results 71
NASNetLarge Convolutional Neural Network 72

NASNetLarge Architecture 72
NASNetLarge Results 75
InceptionV3 Convolutional Neural Network 76
InceptionV3 Architecture 76
InceptionV3 Results 79
Xception Convolutional Neural Network 80
Xception Architecture 80
Xception Results 82
InceptionResNetV2 Convolutional Neural Network 83
InceptionResNetV2 Architecture 83
InceptionResNetV2 Results 85
MobileNetV2 Convolutional Neural Network 86
MobileNetV2 Architecture 86
MobileNetV2 Results 89
ResNet50V2 Convolutional Neural Network 90
ResNet50V2 Architecture 90
ResNet50V2 Results 93
ResNet152V2 Convolutional Neural Network 94
ResNet152V2 Architecture 94
ResNet152V2 Results 95
ResNeXt50 Convolutional Neural Network 96
ResNeXt50 Architecture 96
ResNeXt50 Results 99
Networks Performance Metrics and Network Comparisons 99
Networks Collaborative Approach 105
Equal Voting 105
Selective Voting 106
CHAPTER 4 – COMMON COORDINATE SYSTEM 112
Surveyed Positions Solution 113
Distant Point Calibration Solution 114
Point Correspondence Solution 118
Point Correspondence Solution Data Gathering 120

Point Correspondence Solution Detailed Algorithm 120
Scene Reconstruction and Testing 122
Error Estimation 124
Future Enhancements 127
CHAPTER 5 – CAMERA POSITION OPTIMIZATION 128
Introduction 128
Theory 129
Cepstral Analysis for Translation Estimation 131
Cepstral Analysis for Rotation and Scaling Estimation 132
Results 133
CHAPTER 6 – SUMMARY 139
BIBLIOGRAPHY 143
APPENDIX 1 147
Python Version List 147

LIST OF TABLES

Table 1 - System Functions 4
Table 2 - Subsystems 5
Table 3 - Camera Subsystem 6
Table 4 - Power Subsystem 7
Table 5 - Drive Train Subsystem 8
Table 6 - Communication Subsystem 9
Table 7 - Object Recognition 9
Table 8 - Alternative System Performance Characteristics 18
Table 9 - Subsystems Functional Definition 21
Table 10 - Subsystems Functional Requirements 21
Table 11 - Risk Matrix 26
Table 12 - Unproven Technologies 31
Table 13 - Test Environment 34
Table 14 - Algorithm Image Sizes 59
Table 15 - VGG Convolutional Network Configuration 64
Table 16 - ResNet Various Layer Architectures 95
Table 17 - ResNet50 vs ResNeXt50 Architecture 98
Table 18 - CNN Architectures Performance Metrics 102
Table 19 - CNN Architectures Accuracy Metrics with Folds 102
Table 20 - Paired T-Test 103
Table 21 - Paired T-Test Interpretation 104
Table 22 - Best 4 CNN Architectures with Equal Voting 108
Table 23 - Best 9 Architectures with Equal Voting 108
Table 24 - Best 4 CNN Architectures with Selective Voting with Accuracy Weight Measurement 109
Table 25 - Best 4 CNN Architectures with Selective Voting with 1/error Weight Measurement 109
Table 26 - Best 9 CNN Architectures with Selective Voting with Accuracy Weight Measurement 110
Table 27 - Best 4 CNN Architectures with Selective Voting with 1/error Weight Measurement 111

LIST OF FIGURES

Figure 1 - Operational Approach Objectives 3
Figure 2 - OPM Diagram 14
Figure 3 - Functional Decomposition 20
Figure 4 - Preliminary Concept Diagram 23
Figure 5 - DSM Module 38
Figure 6 - (a) Performance of various sensors, (b) Sensors combined performance 46
Figure 7 - Three Design Phases 47
Figure 8 - System Block Diagram 49
Figure 9 - Image Processor Subsystem Block Diagram 50
Figure 10 - Background Subtraction Block Diagram 51
Figure 11 - Detection Subsystem Block Diagram 52
Figure 12 - Object Tracking Block Diagram 53
Figure 13 - Trajectory Predictor Block Diagram 54
Figure 14 - Feedback Analyzer Block Diagram 54
Figure 15 - Convolutional Neural Network 56
Figure 16 - Stanford Cars Dataset Images 57
Figure 17 - Example of Image Conditions 58
Figure 18 - Cyclical Learning Rate (a) Fixed Rate, (b) Linearly Decaying Rate, (c)Log Decaying Rate 62
Figure 19 - VGG16 Architecture 65
Figure 20 - (a) VGG16 Accuracy, (b) VGG16 Loss 66
Figure 21 - VGG19 Architecture 67
Figure 22 - (a) VGG19 Accuracy, (b)VGG19 Loss 69
Figure 23 - DenseNet201 Architecture 70
Figure 24 - (a) DenseNet201 Accuracy, (b) DenseNet201 Loss 72
Figure 25 - NASNetLarge Architecture 73
Figure 26 - Overview of Neural Architecture Search 74
Figure 27 - Scalable Base Blocks 74
Figure 28 - (a) NASNet Search Space, (b) Convolutional Cell Choices of Operation 75
Figure 29 - (a) NASNetLarge Accuracy, (b) NASNetLarge Loss 76
Figure 30 - InceptionV3 Architecture 77
Figure 31 - (a) Naive Inception Module, (b) Inception Module 78
Figure 32 - (a) Inception Module, (b) InceptionV3 Module, (c) InceptionV3 Module with expanded filter bank 78
Figure 33 - (a) InceptionV3 Accuracy, (b) InceptionV3 Loss 79
Figure 34 - Xception Architecture 80
Figure 35 - (a) Inception Module, (b) Simplified Inception Module 81
Figure 36 - (a) Reformulated Simplified Inception Module, (b) Xception Module 81
Figure 37 - Xception Architecture Flow 82

Figure 38 - (a) Xception Accuracy, (b) Xception Loss 83
Figure 39 - InceptionResNetV2 Architecture 84
Figure 40 - InceptionResNetV2 Structure 85
Figure 41 - (a) InceptionResNetV2 Accuracy, (b) InceptionResNetV2 Loss 86
Figure 42 - MobilenetV2 Architecture 87
Figure 43 - (a)-(d) Evolution of Separable Convolution Blocks 88
Figure 44 - (a) Residual Connection Block, (b) Inverted Residual Connection Block 88
Figure 45 - (a) MobileNetV2 Accuracy, (b) MobileNetV2 Loss 89
Figure 46 - ResNet50 Architecture 91
Figure 47 - (a) Deep Plain Network Training Error, (b) Deep Plain Network Test Error 92
Figure 48 - ResNet Building Block 92
Figure 49 - Network Architectures Comparison Between Plain Networks and ResNet 93
Figure 50 - (a) ResNet Building Block, (b) Bottleneck Building Block 93
Figure 51 - (a) ResNet50V2 Accuracy, (b) ResNet50V2 Loss 94
Figure 52 - (a) Performance of Plain Deep Network on ImageNet, (b) Performance of ResNet Network on ImageNet 95
Figure 53 - (a) ResNet152V2 Accuracy, (b) ResNet152V2 Loss 96
Figure 54 - (a) ResNet Building Block, (b) ResNeXt Building Block 97
Figure 55 - (a) ResNet50 vs ResNeXt50 Performance, (b) ResNet101 vs ResNeXt101 Performance 98
Figure 56 - (a) ResNeXt50 Accuracy, (b) ResNeXt50 Loss 99
Figure 57 - CNN Architectures Performance Metrics 101
Figure 58 - Enhanced System Block Diagram 112
Figure 59 - Surveyed Cameras Positions Solution Concept 113
Figure 60 - OOI Position Calculation 116
Figure 61 - Estimate Matching Features Subsystem 118
Figure 62 - Estimate Fundamental Matrix and Relative Pose Subsystem 119
Figure 63 - Synthesized Data Overview 120
Figure 64 - Correspondence Features 121
Figure 65 - Synthesized Data Overview 121
Figure 66 - Relative Position and Orientation 122
Figure 67 - Scene Reconstruction 122
Figure 68 - First Example (a) Remap onto Camera1 Scene, (b) Remap onto Camera2 Scene .. 123
Figure 69 - Second Example (a) Remap onto Camera1 Scene, (b) Remap onto Camera2 Scene 124
Figure 70 - (a) Template, (b) Remapped ROI onto Camera1 Scene, (c) Remapped ROI onto Camera2 Scene 125
Figure 71 - Cross-Correlation Result 125
Figure 72 - (a) Original Mapped Location onto Camera1 Scene, (b) Error Estimated onto Camera2 Scene 126
Figure 73 - (a) Original Mapped Location onto Camera1 Scene, (b) Error Estimated onto Camera2 Scene 126
Figure 74 - Test Scenario Trajectory 134

Figure 75 - (a) I1/I1 Concatenation, (b) I1/I1 Concatenation Cepstrum 135
Figure 76 - (a) I1/I5 Concatenation, (b) I1/I5 Concatenation Cepstrum 136
Figure 77 - (a) I1/I10 Concatenation, (b) I1/I10 Concatenation Cepstrum, (c) I1/I10 Ceptrum Closeup 137
Figure 78 - a) I1/I15 Concatenation, (b) I1/I15 Concatenation Cepstrum, (c) I1/I15 Ceptrum Closeup 138

LIST OF KEYWORDS

Continuous Tracking
Object Recognition Tracking
Background Subtraction
Trajectory Prediction
Surveillance
VGG16 Convolutional Neural Network
VGG19 Convolutional Neural Network
DenseNet201 Convolutional Neural Network
NASNetLarge Convolutional Neural Network
InceptionV3 Convolutional Neural Network
Xception Convolutional Neural Network
InceptionResNetV2 Convolutional Neural Network
MobileNetV2 Convolutional Neural Network
ResNet50V2 Convolutional Neural Network
ResNet152V2 Convolutional Neural Network
ResNeXt50 Convolutional Neural Network
Collaborative Learning
Transfer Learning
Correlation of Camera Mesh Network
Camera Position Optimization
Cepstral Analysis
Systems Engineering Design Methodology
Standford Cars Dataset

CHAPTER 1 – SYSTEM ENGINEERING METHODOLOGY

Introduction

The scope of this section is to use the system engineering methodology to develop a system that continuously tracks an object of interest. The systems engineering methodology is adapted from the book "Systems Engineering Principal and Practice" by Alexander Kossiakoff, published in 2011. The methodology consists of the needs analysis; concept exploration; concept definition; risk assessment and mitigation; advanced development; software systems; engineering design; integration and evaluation; and the plan for production and operation support. Since the system is considered to be a complex system, the research will be divided into three phases: A single camera system performing object recognition and tracking, a multi-camera system, and lastly a multi-camera fixed on a moving platform such as a drone. This chapter will discuss the design aspect of the third phase as a whole; whereas, the following chapters will focus on the technical development of the more complex subsystems of the other phases.

Needs Analysis

In 2011, I was serving as a Search and Rescue (S&R) volunteer at the Orange County Sheriff Department (OCSD). During our S&R missions and disaster preparation training we were exposed to various protocols for disaster preparation, searching and tracking of individuals, and emergency response. The S&R team is composed of the search team and the rescue team. The search team is responsible to locate and track an object of interest, and the rescue team is responsible to develop a tactic to resolve the issue. In many cases, the search team discussed the need to continuously track an object of interest. Some examples include continuously tracking a sick or insured animal in a mountainous area, or continuously searching and tracking a fugitive.

Another example is the continuous monitoring of the rescue team during their mission to evaluate their tactical moves. In other scenarios, an automated system was needed to assess a disaster site, such as a collapsed building, or to track all the insured personnel and remap a site after it was reconfigured by the disaster. Therefore, I saw a need to develop a system to assist the S&R personnel to perform their job safely. The needs analysis is performed next to rationalize the market analysis of the proposed system.

Operational Analysis

The "Systems Engineering Principal and Practice" book divides the operations analysis process into two stages: (1) analysis of the projected needs, and (2) definition of the operational approach. These two steps will be discussed in detail in the next subsections.

Analyze Projected Needs

The customers' requirements were gathered by developing a user story board and user wish list on what helps them the most to perform the job safely. The list below shows the items that were requested by the customers.

- They require correlating different cameras that do not all have overlapping views.
- They require understanding the possible routes a fugitive can take and predict his/her next move.
- They require optimizing camera positions to continuously track an object of interest throughout a scene.
- They require deploying a swarm of autonomous vehicles to track an object of interest throughout a city.

- They require the ability to optimize the images in various environmental conditions.

Define the Operational Approach:

Figure 1 below describes the operational approach objectives. The primary objects are an autonomous network of cameras along with 3D scene reconstruction and proper classification and localization of the object of interest. The secondary objective of the classification and localization is applying these algorithms to real-time object recognition. The secondary objective for the 3D scene reconstruction is to parameterize the various possibilities for the scene and the camera tracking. This will facilitate the optimization of positioning the cameras. Lastly, the secondary objective of the autonomous network will allow a collective knowledge of the scene which will enable the global knowledge of the system.

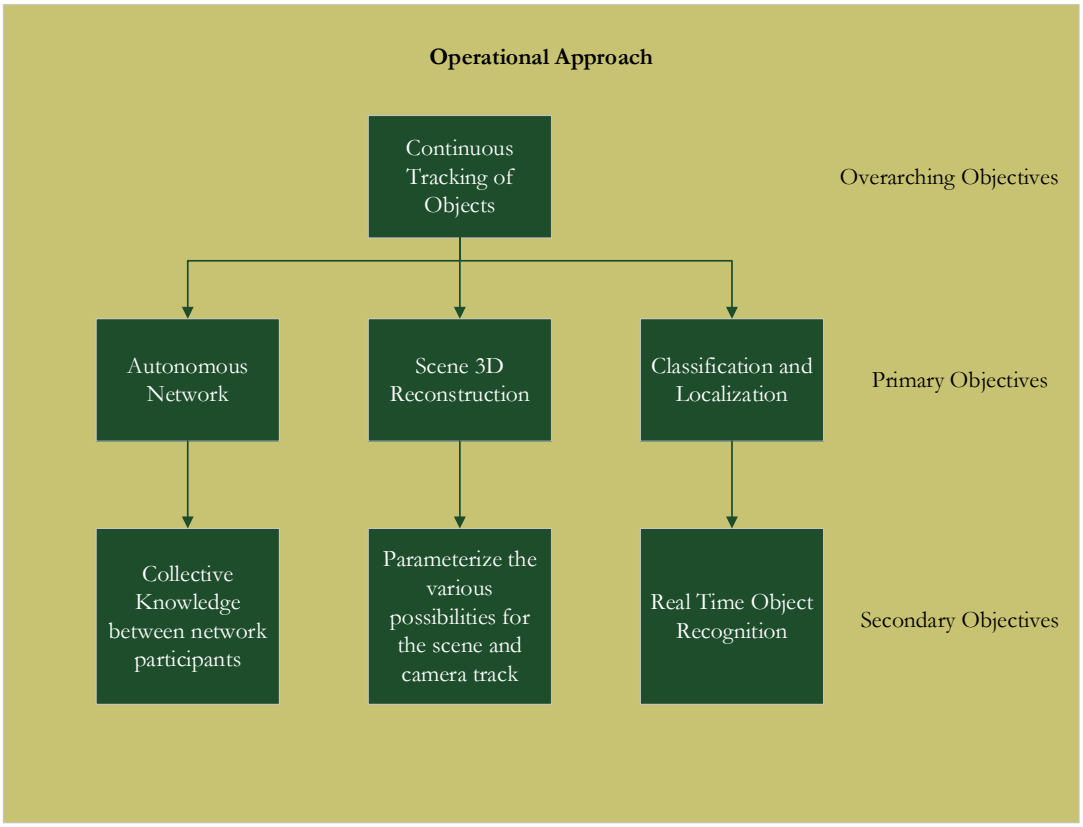


Figure 1 - Operational Approach Objectives

Functional Analysis

According to the Systems Engineering process as described above, this process comprises two steps: (1) translation into functions, and (2) allocation of functions to subsystems. The main purpose of the functional analysis is to decompose the system into processes and operations to achieve the designed operational needs.

Translation into Functions

The translation of the objects into main functions is described in Table 1 below.

Table 1 - System Functions

Function	**Description**
Image Enhancement / Processing	The ability to enhance the image (de-blurring, sharpening, removing background)
Real Time Object Recognition	This implies that the system will be able to classify and localize the Object Of Interest (OOI) in real time. It also means that the system shall have a pre-trained Convolutional Neural Network (CNN) already in place.
Networked system	All cameras have collective knowledge of each other and their location in the system
Central Knowledge.	This implies that all cameras communicate directly to a centralized computer that will have the ability to command and control each system asset.
3D Reconstruction	The ability to reconstruct the scene. This will allow better understanding of the options and the surroundings of the Object of Interest.
Moving Cameras	The ability to re-host for a later project the application onto a moving camera.
Camera Placement	The ability to place the moving cameras in the scene to minimize occlusion, and camera overlap and maximize the scene.

Allocation of Function into Subsystems

In this step, the previously identified functions, will be divided into subsystems. This initial modulation will help the system engineer further design the complex system. The

CONOPS of the system/subsystem will adopt the end-goal of this project and is defined by Table 2 below. The next chapters will subdivide the end-goal into several phases to ensure a successful completion of this project.

Table 2 - Subsystems

Candidate Subsystem	**Description**
Camera System	This subsystem will detail the minimum requirements for the cameras for the entire system.
Camera System Picture Resolution	The subsystem will detail the minimum resolution for the cameras for the entire system.
Camera Data Range	The camera system can be composed of various data ranges such as Visual (RGB), Thermal, LiDar and Radar for complete scene analysis. This is hyperspectral imaging.
Camera System Zoom functionality	This subsystem will detail the minimum requirement for the zoom level of the entire system.
Camera system Rotation	This subsystem will detail the requirements for camera rotation.
Drone Structure and support	The drone structure and support shall be ruggedized so it withstands the environmental conditions dictated by the application of the system
Power system	This subsystem will detail the entire power distribution minimum requirements to the drones and the cameras
Drive Train	This subsystem will detail the requirements for the drivetrain of the drone
Wireless Real-time Vision	This subsystem will detail the requirements of the real-time transmission of the data to a centralized computer.
Navigation and Steering	This subsystem will detail the requirements of the navigation system.
Sensory System	This subsystem will detail the requirements of the sensory system required to ensure proper operation of the system. Ex. Anti-crash system.
Wireless Command and Control	This subsystem will detail the manual command and control and/or the autonomous control to allow all drones to collaborate to continuously track an Object Of Interest (OOI)
Scene Reconstruction	This subsystem will detail the requirements for the scene reconstruction to properly track each camera position in a given environment
Object of Interest Recognition	This subsystem will detail the two different aspects of recognition which are: Localization and Classification.

Feasibility Definition

The feasibility definition process consists of two steps: (1) visualizing the subsystem technology, and (2) defining the feasibility concept. This allows the systems engineer to realize the system under-design and help him/her to perform the tradeoff analysis.

Visualize Subsystem Technology

The system will be mainly used for aforementioned applications; therefore, it needs to be a ruggedized system with all its subcomponents meeting the overarching environmental requirements of MIL-STD810G. To better design each of the subsystems, the subsystems described in the previous section will be decomposed and visualized by the following tables and discussions.

Table 3 - Camera Subsystem

Technology	**Advantage**	**Disadvantage**
Shutter Speed > 1/500 s	- Be able to take still images of fast-moving objects - Reduce motion blur	- More expensive - Underexposed photos
Aperture with f-stop at least 2.8	- Be able to capture faster moving objects - Able to capture greater details	- Overexposed photos for slower shutter speeds
Camera Weight < 1000g	- Less payload on the drone	- Inferior glass lens quality
High Definition Resolution	- Higher quality images - Ability to pick smaller details and features of the OOI	- Larger image to store - Larger image to process - Larger image to send real-time.
Zoom Functionality	- Ability to maintain higher altitude of during flight while registering the needed	- More expensive camera
Camera Rotation	- Ability to swivel the camera without changing the direction of the drone	- More mechanical parts
Camera Data Type: RGB	- High resolution - High marking detection - High range - Low sensory cost	- Poor Performance in bad weather conditions: fog, snow, rain

	- Small sensory size - Average velocity detection - Average glare resistance - Average distance from object	- Poor performance in sun Blinding - Poor in low light conditions
Camera Data Type: Thermal	- High performance in sun blinding condition - Glare resistant - High performance in low light conditions - Low sensor cost - Small sensor size - Average resolution	- Low performance in marking detection - Low performance in velocity detection
Camera Data type: LiDar	- High performance in low light conditions - Long distance from object - High performance in velocity detection - High Performance in glare condition	- Large sensor size - High cost - Low range - Low resolution - Low performance in bad weather conditions: rain, fog and snow - Low performance in marking detection
Camera Data Type: Radar	- High performance in poor weather conditions - High performance in sun-blinding conditions - Glare resistant - Low sensory cost - High performance in High Velocity detection - High performance in low light conditions - High range	- Low performance in marking detection - Low distance from the objects.

Table 4 - Power Subsystem

Technology	Advantage	Disadvantage
NiCa/NiCd Nickel Cadmium	- Harder to damage - Last longer, more charge cycles	- Don't last long in high draining systems - Higher cost - Very toxic.
NiMH Nickel Metal Hydride	- More affordable - Better capacity	- Short shelf life since they self discharge.

	- Lasts longer than NiCd.	- Not as durable as NiCa
Li-Ion Lithium Ion	- High energy - Slow discharge rate - Most energy capacity - Light weight	- Overlong time they lose their capacity permanently - Least durable - Most expensive.

Table 5 - Drive Train Subsystem

Technology	Advantage	Disadvantage
Rover drivetrain with chains	- More torque - Off-roading capability	- Debris can get stuck on the chain - Harder to suddenly rotate
Rover drivetrain with 4x4 drive free rotations	- More freedom to rotate - Less probability to get stuck - Easier to climb than chain	- Less torque
Drone drivetrain – quadcopter	- Faster to maneuver than rover - Easier to get birds eye view of the site	- Harder to maneuver in low light conditions - Less stable than the hexacopter.
Drone drivetrain – hexacopter	- More stable than quadcopter - Faster to maneuver than rover - Easier to get birds eye view of the site	- Harder to maneuver in low light conditions
Stepping robot	- Most torque - Easiest to climb	- Slow Speed

Table 6 - Communication Subsystem

Technology	Advantage	Disadvantage
WiFi IEEE 802.11 Encrypted Network	- Encrypted - Already developed/specified protocol	- Might have limitation - Cyber security breach - Relatively short range - Channel disturbance
RF Communication In house developed protocol	- More secure - Developed specifically for this application - Longer range	- More expensive to implement - Allocate certain frequencies for the system.

Table 7 - Object Recognition

Technology	Advantage	Disadvantage
Supervised Learning – Using Statistical analysis approaches	- Easy to implement - Faster to train	- Less accurate than CNN. -
Supervised Learning using Convolutional Neural Networks	- Highest accuracy - There are already developed trained using ImageNet - Works great in the image recognition application	- Slower than NN.
Unsupervised	- Fastest Algorithms	- Requires significant amount of time for verification.

Define Feasibility Concept

To meet all the requirements mentioned above, the optimum model shall be a hexacopter drone powered by Li-Ion batteries. The drone shall be equipped with various camera systems assigned for the object detection system. These various cameras data types are to be RGB,

Thermal, LiDar, and RADAR cameras to complement each other for the tracking purposes. The system shall be equipped with LiDar dedicated for object detection/avoidance. The RGB cameras shall record high resolution images with at least 1/500s shutter speed and f-stop at least 2.8. The cameras shall have a mass of less than 1000g each. The cameras shall be zoom capable up to 30x and fixed from rotation to facilitate the calculations since the drone is fully integrated with the recognition system. The Recognition subsystem shall be a supervised CNN model for optimum results. The cluster of drones shall be able to reconstruct the 3D scene.

Needs Validation

The needs validation process is composed of a design effectiveness model and will validate needs and feasibility. The objective of this process is to verify that the operational needs are being met.

Design Effectiveness Model

The Measure Of Effectiveness (MOE) is taken mainly out of the customer requirements documents. The main requirements state that the system shall work for 2 hours without charge, and that the drone agents shall calculate its way back to the recharge station. The drones shall send the current view to a centralized computer to calculate trajectory and upload a mission to each of the drones. The centralized computer shall be redundant. The centralized computer shall map the robots relative to each other to better calculate schedules. The swarm system shall have a global knowledge of the OOI at all times. The OOI recognition shall achieve accuracies above 90%. As far as the Measure Of Performance (MOP), it will be developed after refining the subsystems.

Validate Needs and Feasibility

The proposed feasible concept addresses the needs of the S&R division. It is able to satisfy the overarching objective described at the beginning of this section and meet all the operational requirements. These operational requirements are:

- The system shall work for 2 hours without charge

- It shall be capable of withstanding the extreme environmental conditions.

- It shall be able to meet all the MIL-STD-810G

- The drone agents shall calculate its way back to the recharge station.

- The communication to a centralized computer is achieved by sending the current OOI position independently.

- The centralized computer shall calculate the OOI trajectory and upload the information to the drones.

- The centralized computer shall be redundant.

- The centralized computer shall map the robots relative to each other to better calculate schedule, and reconstruct the scene.

- The drones shall be able to avoid obstacles

- The system shall be interrupted by a human user at any given time.

- The system shall be smart to detect the features of the OOI with a 90% accuracy. The Machine Learning (ML) preferred technique is the CNN or Convolutional Neural Network (CNN) which is used almost ubiquitously for image recognition and feature extraction.

- It shall be reliable for at least 10 years.

- It shall be maintainable. This is done by decomposing the system into subsystems.

Concept Exploration

The concept exploration stage of the systems engineering methodology is used to explore alternative systems to produce a more neutral solution. The process of the concept exploration is described by the diagram in Chapter 7 of Kossiaskoff's textbook (Figure 7.2), "Systems Engineering Principal and Practice".

The input of this phase of the process is the output of the needs analysis phase. To summarize the output of the previous phase, a system is needed to aid the S&R team to be able to track an object of interest throughout a scene from various cameras that are not all overlapping. The system shall be able to recognize an object of interest with at least 90% accuracy. It shall be able to localize it on a given camera and be able to correlate its position with the other camera systems. It shall be easily maintainable and have 10 years of reliability.

Operational Requirement Analysis

The Operational Requirement Analysis is mainly composed of two steps: Analyzing Operational Objectives and Refining Operational Requirements. The purpose of this step is to explore the different operational Requirements that may lead to alternative solutions.

Analyzing Operation Objectives

Iterating through the project objective analysis:

- They required correlation of different cameras that do not all have overlapping views. "**Situational Awareness, Asset Correlation, Continuity of Tracking"**

- They required understanding of the possible routes a fugitive can take and predict his/her next move. "**Situational Awareness, Trajectory Prediction, Continuity of Tracking"**
- They required optimization of the camera positions to continuously track an object of interest throughout a scene. "**Continuity of Tracking with minimum assets"**
- They required a deployment of a swarm of autonomous vehicles to track an object of interest throughout a city. "**Continuity of Tracking**, **Recognition and Localization, Multiple Assets**"
- They required the ability to optimize the images in various environmental conditions. "**Accuracy of Tracking through Unpredictable Conditions"**

Refining Operations Requirements

The more refined operational requirements demand that the system shall be able to deploy at least 50 drones simultaneously and be able to position them such that there is a minimum overlap between them. The positioning of the assets shall be according to the trajectory prediction sent from the centralized system.

Performance Requirement Formulation

The performance requirements formulation is composed of two steps: Derive subsystem functions and formulate performance parameters. The objective of this process is to define the subsystem functions so the MOP can be formulated.

Derive Subsystem Functions

This task is best visualized by the OPM diagram which is optimal in deriving subsystem functions. The OPM diagram is composed of mainly three sections: (1) the operands: this is where the subsystems are defined; (2) the value process: this is where the functional actions are

described; and (3) the instrument section: this is where the objects performing these functional actions are defined.

Figure 2 shows the OPM diagram that describes our system. Operands show the basic subsystems; the value processes show their function in the system and the instruments show the means that allow them to perform these value processes or functions.

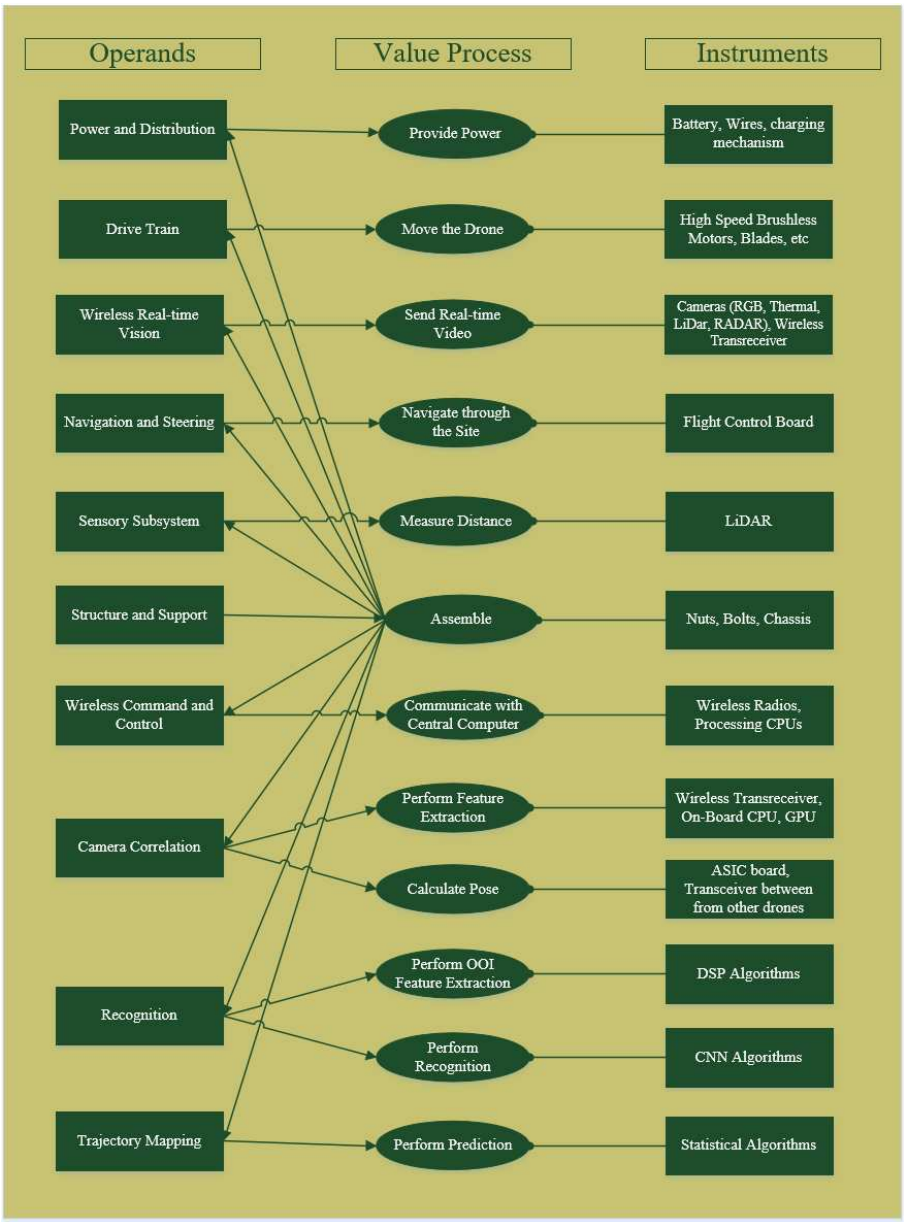


Figure 2 - OPM Diagram

Formulate Performance Parameters

The formulation of the MOP will be defined by the following requirements.

1) The system shall be ruggedized to withstand the environmental constraints. Therefore, the system shall meet MIL-STD810G. The MIL-STD810G defines the requirements of the following environmental conditions:

- Test Method 500.5 Low Pressure (Altitude)
- Test Method 501.5 High Temperature
- Test Method 502.5 Low Temperature
- Test Method 503.5 Temperature Shock
- Test Method 504.1 Contamination by Fluids
- Test Method 505.5 Solar Radiation (Sunshine)
- Test Method 506.5 Rain
- Test Method 507.5 Humidity
- Test Method 508.6 Fungus
- Test Method 509.5 Salt Fog
- Test Method 510.5 Sand and Dust
- Test Method 511.5 Explosive Atmosphere
- Test Method 512.5 Immersion
- Test Method 513.6 Acceleration
- Test Method 514.6 Vibration
- Test Method 515.6 Acoustic Noise
- Test Method 516.6 Shock

- Test Method 517.1 Pyroshock
- Test Method 518.1 Acidic Atmosphere
- Test Method 519.6 Gunfire Shock
- Test Method 520.3 Temperature, Humidity, Vibration, and Altitude
- Test Method 521.3 Icing/Freezing Rain
- Test Method 522.1 Ballistic Shock
- Test Method 523.3 Vibro-Acoustic/Temperature
- Test Method 524 Freeze / Thaw
- Test Method 525 Time Waveform Replication
- Test Method 526 Rail Impact.
- Test Method 527 Multi-Exciter
- Test Method 528 Mechanical Vibrations of Shipboard Equipment (Type I – Environmental and Type II – Internally Excited)

2) The system shall be operational for at least 2 hours without need to charge.
3) The system shall be able to correlate all the assets.
4) The centralized computer shall contain redundant systems that can take over in case of failures.
5) If drones need to be recharged, it shall return automatically to the initial deployment point for further handling.
6) The centralized computer shall send the assets the trajectory based on the data received from each asset.
7) The centralized computer shall deploy assets as needed based on their status.
8) The drones shall be able to avoid obstacles autonomously.

9) The robotic agents shall be able to capture the image with various data types namely: RGB, Thermal, LiDar, RADAR.

10) The robotic agents shall be able to recognize a given OOI with accuracy of at least 90%.

Implementation of Concept Exploration

The implementation of concept exploration is composed of two steps: Explore Implementation of the concepts and define performance characteristics. The objective of this step of the process, is to evaluate alternative solutions; this is to explore the design space and allow the systems engineer to come up with a solution neutral system, doing so will eliminate focusing on only single point solutions.

Explore Implementation Concepts

The alternative concepts are:

- Rover that with a chain drive train

-Rover with drive 4 wheels drive train with full degree of rotation freedom

- Quadcopter drone

- Hexacopter drone

- Stepping drone

- Already existing cameras such as traffic cameras and ATM cameras

Define Performance Characteristics

In the previous section the key advantages and disadvantages of the alternative concepts were discussed. Table 8 below reiterates these design concepts for the drivetrain; however, the concept of utilizing the existing cameras such as traffic, security and ATM cameras is added.

Table 8 - Alternative System Performance Characteristics

Technology	Advantage	Disadvantage
Rover drive train with Chains	- More torque - Off-roading capability	- Debris can get stuck on the chain - Harder to suddenly rotate
Rover drivetrain with 4x4 drive free rotations	- More freedom to rotate - Less probability to get stuck - Easier to climb than chain	- Less torque
Drone drivetrain – quadcopter	- Faster to maneuver than rover - Easier to get birds eye view of the site	- Harder to maneuver in low light conditions - Less stable than the hexacopter.
Drone drivetrain – hexacopter	- More stable than quadcopter - Faster to maneuver than rover - Easier to get birds eye view of the site	- Harder to maneuver in low light conditions
Stepping robot	- Most torque - Easiest to climb	- Slow speed
Existing Cameras	- Cheaper to implement	- Fixed locations - Not having multiple sensors - Different resolutions

Performance Requirements Validation

The performance requirement validation is composed of two steps: (1) integrate performance characteristics and (2) validate performance requirements. The objective of this step is to validate that the performance requirements are met by exploring the various alternatives in this section.

Integrate Performance Characteristics

Most of the alternative designs above are feasible, and with currently available technology, it is possible to get many COTS subsystems. However, some of the features accompanying each alternative design brings an undesirable side effect. For instance, no rover robot will not be able to perform tracking efficiently as a drone. It is also hard to maneuver a driving object in such an environment. The stepping robot is undesirable because it is much slower and not efficient. Despite the fact that the existing cameras are appealing, it is not easy to optimize their positions to truly have a continuity of tracking. The network of hexacopter-drones present the best scenario for our needs because of the degree of freedom it presents as well as the speed and stability needed to keep the target in track.

Validate Performance Requirements

The performance requirements validation, a Model Based Systems Engineering (MBSE) model, will be created to model the integrated behavior and better understand the emergent behavior in different conditions. All basic requirements described in this section shall be met.

Concept Definition

The concept definition discussed in this section follows the systems engineering methodology described in Chapter 8 of the Kossiakoff textbook. The objective of this process is to further define the chosen system.

Performance Requirements Analysis

The performance requirement analysis is composed of two steps: Analyze performance requirements, and refine performance requirements. The objective of this section is further analyzing the system and developing more detailed requirements.

Analyze Performance Requirements

The system at this step has been well defined from the previous sections; however, each of the requirements can be further broken down to detailed requirements.

Refine Performance Requirements

The system has been adequately refined in the previous section; therefore, no further refinement is needed at this point.

Functional Analysis and Formulation

The functional analysis and formulation phase is composed of two steps: Define components functions, and formulate functional requirements. The objective of this design step is to formulate the design into functions that meet all the objective requirements

Define Components Functions

The first decomposition diagram is shown on Figure 3 below. It is composed of six main functions: Structure and Support, Power Supply and Distribution, Drivetrain, Vision, Navigation and Steering, and Wireless Transmission. The Objective of these functions will be detailed in Table 9 below.

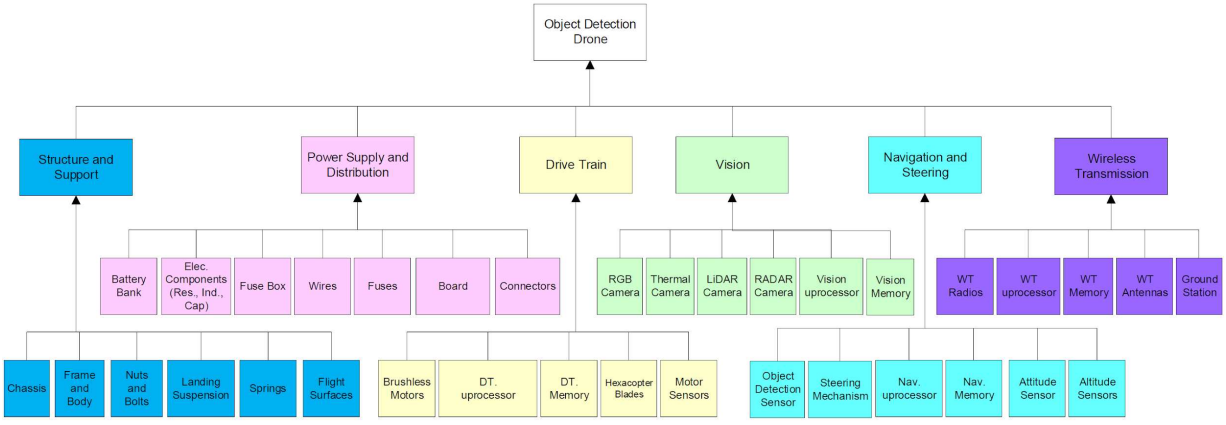


Figure 3 - Functional Decomposition

Table 9 below details each of the functions specified by the preliminary functional diagram discussed above.

Table 9 - Subsystems Functional Definition

Component	Function
Structure and Support	Provides the ruggedized structure and linkage of the system to each of its subcomponents.
Power supply and distribution	Provides power to the whole system. It provides means to recharge its on-board batteries.
Drive Train	Provide the means for the drone to fly.
Vision	Provides the drones with capabilities to capture images in various data formats: RGB, Thermal, LiDar, and RADAR. The system is also composed of the microprocessors that provide the means for the drone to perform the object recognition and tracking.
Navigation and Steering	Provides the means for the drones to be controlled in manual mode or submitted to its mission files. It also contains an object avoidance sensory system.
Wireless Transmission	Provides the means to transmit the mission information and flight profile from the centralized system

Formulate Functional Requirements

Table 10 below provides the requirements of the functional definition:

Table 10 - Subsystems Functional Requirements

Component	Function
Structure and Support	-It shall be ruggedized to withstand all the environmental conditions. -It shall provide an enclosure for the entire system that meets with MIL-STD810G.
Power supply and distribution	- It shall provide the system with the energy needed to properly operate the system. (this energy/hr will be defined at a later stage) - It shall provide the system with uninterrupted power for full operation for 2 hours. - It shall provide the system with power protection.
Drive Train	- It shall provide the drone the means to lift to an altitude that will be specified at a later stage. - It shall provide the drone the means to move at a different speed - It shall provide the drone to land

Vision	- The system shall contain multiple cameras to detect the OOI. The cameras can be of various data types such as Thermal, LiDAR, RGB, and RADAR - The system shall have high resolution through zoom and sensitive cameras to detect the OOI at an altitude that will be specified at a later stage. - The system shall be able to recognize and track a give OOI with an accuracy of at least 90% - The recognition SW shall take 100ms to scan the entire image. - The recognition SW shall take 1ms to recognize the OOI.
Navigation and Steering	- It shall allow the drone to receive navigation and steering commands manually - It shall provide the means for the drone to navigate autonomously. - The system shall provide the drone the means to avoid obstacles.
Receive Mission Information	- It shall provide the user with real-time feedback with at most 5 seconds delay. - It shall provide the centralized computer the position information of the OOI from each of the drones that has it on sight. - The drone system shall be able to receive information from the centralized computer to start a mission - The drone system shall be able to receive information from the centralized computer to update the current mission

Concept Selection

The concept selection is composed of two steps: synthesize alternative concepts and select preferred concepts. The objective of this section is to select the preferred method to start forming the architecture.

Synthesize Alternative Concepts

The system as was selected by the previous steps doesn't require further search for alternative concepts due to the stringent requirements given. Thus, the preferred concept that will meet all the requirements is composed of a mesh network of maximum 50 hexacopters drones that are equipped with several cameras to primarily object track certain OOI. The system is also

equipped with a centralized system that commands the drones with missions based on the trajectory calculation. The drone system is capable of avoiding obstacles. All camera systems are commonly coordinated so any of the drones has the OOI relative position.

Select Preferred Concept

Based on the analysis that has been performed by the concept exploration, the preferred concept is the hexacopter drone, as Figure 4 shows with the preliminary design below.

Figure 4 - Preliminary Concept Diagram

Concept Validation

The concept validation process is composed of two steps: conduct system simulation and validate the selected concept. The objective of this section is to validate the preferred concept meeting the primary objectives of the project.

Conduct System Simulation

Much research has been done in the area of object recognition using CNN. The best algorithm fit for the type of object, in our case is make and model of cars, needs to be determined after retraining the CNN algorithms. Chapter 3 of this dissertation discusses in detail the CNN algorithms that were trained using the transfer learning techniques and results achieved. Similarly, correlating the cameras in the network to a common coordinate system is discussed in detail with the simulation performed in Chapter 4 of this dissertation. The entire system will be integrated and simulated using the MBSE approach to simulate the entire system before prototyping it.

Validate Selected Concept

The validation of the concept performed in the previous section is valid up to this point of the design. No further validation is needed, as it showed previously. All primary objectives have been met using this approach.

Risk Management

Project Management Related Risks

There are several risks to this project, some of which are project management risks and others are risks associated with technical maturity and development.

1) TRL-7: Customer indicated that they require a Technology Readiness Level (TRL) of 7 before paying for the manufacturing of the product. This resembles a financial risk on the development of the product. TRL 7 means that the model is demonstrable in the operational Environment.
2) Authority to Operate (ATO) in the airspace: This item requires the Federal Aviation Authorization (FAA) approval before deploying these autonomous tracking vehicles.
3) Due to COVID-19 the supply of memory and chips is very short and has a long lead time.
4) Object detection algorithm: Developing a custom object detection algorithm for a specific application, such as cars, could be challenging due to the fact not many datasets are available and the class features are very closely correlated.
5) Camera coordination: Correlating all the cameras to a common coordinate system has not been a top priority in object tracking; thus, developing an algorithm that can correlate all the cameras in a given network could be challenging.
6) Camera position optimization: A nested risk in developing a camera correlation algorithm is optimizing the camera positions to cover a given scene.
7) Drone Integration: Integrating the object detection, tracking the camera correlation, and determining the object trajectory are challenging tasks. In particular, the movement space has 6 degrees of freedom.
8) Allocating a radio frequency: Allocating a radio frequency to command and control the drone as well as send a receive mission file and high definition images can be challenging

Qualitative Risk Assessment

The risk assessment was assessed on Table 11 which is 3x3 matrix; it can be easily translated to a 5x5 matrix.

Table 11 - Risk Matrix

	Consequences			
Likelihood		Low	Medium	High
	High		**6**	**1 ,7, 8**
	Medium		**4,5**	**2**
	Low		**3**	

For item #1, the likelihood of this happening is high due to the fact the customer has stated that they don't invest in any new technology less than TRL-7. The consequences of this risk are high due to the fact it will cost the company a large amount of money in research, modeling, developing prototypes, and testing without aided investment. The risk has been accepted and the proposed solution is to find investors for this product.

For item#2, the likelihood of this happening is high due to the fact the FAA has restricted the airspace for the desirable altitude for autonomous vehicles. The consequences of this risk are medium because it is necessary to test the drone in a controlled environment before deployment to ensure all the safety standards are met. This risk will be transferred to the customer to seek the appropriate approvals through the FAA.

For item #3, the likelihood is low because currently a lot of car manufacturing and computer manufacturing are waiting for microchips and memory. By the time this product is fully designed and ready for production, this limited supply might be exhausted. The consequences are medium because ordering the necessary parts can be well coordinated with the supply chain organization.

For item #4, the likelihood is medium because this is an area this dissertation needs to investigate and resolve and the consequences are medium as well because finding a promising object detection algorithm can be challenging.

For item #5, the likelihood is medium because this is an area this dissertation needs to investigate and resolve and the consequences are medium as well because finding a promising camera correlation algorithm can be challenging.

For item #6, the likelihood is high because this is an area this dissertation needs to investigate and resolve and the consequences are medium as well because finding a promising and cheap way to optimize the camera position can be challenging yet not mandatory.

For item #7, the likelihood is high because integrating autonomy, object detection and tracking and a 6-degrees of freedom device is challenging in its nature. The consequences are also high because this is what most of the project needs to address.

For item #8, the likelihood is high because there are some drones that currently are commercially available that send and receive the required information; however, further research is necessary to ensure that these frequencies can cover the altitude and distance required by the customers' needs. The consequences are high because if the currently developed frequency ranges do not work further research is required to establish the proper bandwidth and accepted amplification.

Advanced Development

The advanced development follows the process diagram in Figure 10.2 of the "Systems Engineering Principal and Practice" book. The purpose of the advanced development is to

reexamine the validity of the system functional specifications developed and identify the components that are not fully matured.

Requirements Analysis

The requirement analysis is composed of analyzing system functional specifications and identifying immature components that require further study and mitigation.

Analyze System Functional Specs

The preliminary functional requirements were specified by Table 10. The main components of the systems were: structure and support, power supply and distribution, drive train, wireless vision, navigation and steering, sensory system, receive mission information, camera system and recognition and tracking. These preliminary requirements were inserted to guarantee that the system will meet its primary objective in the rough environmental conditions specified. In this section we will analyze the functional requirements specified on the aforementioned table and identify redundant systems. Please note that these requirements can be greatly explored on a much finer granularity level. The most important requirement identified by the customer is the accuracy of tracking. Thus, building a detailed model that shows the expected accuracy is necessary. The second most important requirement identified by the customer is to meet all the environmental constraints specified by MIL-STD810G. The customer specified to ensure that the drone is fully operational for 2 hours to meet the field needs. Thus, A detailed MBSE model is needed to estimate the energy needed and the power budget required to meet the customer's need. The system shall provide the user with real-time feedback with a maximum 5 for upload and download data on a private frequency range. The drones shall be able to communicate constantly with the centralized computer. Lastly, all the drones shall have a

common and global knowledge of the OOI throughout the scene even if they don't have it on sight.

Identify Immature Components

There are several components that require immediate assessment to ensure the feasibility of the project. The first one is that the recognition and the object tracking must achieve at least 90% accuracy. The second is to ensure that all of the cameras achieve a common coordinate system so they can reference a point in space regardless if they have it in sight. The third is to ensure that the radio frequencies required to upload and download the data are suitable for real-time high definition feed. The fourth is to ensure that the proper estimation of weight and structure of the drone. Lastly, power needs to be modeled to ensure the proper estimation of the power budget. These tools will include: Matlab, Simulink, Python, Pspice, CATIA, SketchUP, CAMEO and Capella.

Functional Analysis and Design

Identity Functional Performance Issues

Each of these components present a critical characteristic that needs further identification. Table 10.3 of Kossiakoff best describes the functional elements and their related critical characteristics. In regards to the structure and support, the functional requirements that were described previously stated that the structure and support shall be ruggedized to withstand all environmental constraints described by the MIL-STD810G. The critical characteristics that need further specifications to meet such requirements can stress the strength and the versatility of the materials as well as the form and the join material should be designed in accordance with the total weight, capacity of the body. That is why a prototype is essential for such a purpose. Power supply and distribution requirements are: the system shall provide uninterrupted power for 2

hours straight, and it shall provide enough energy to operate the vehicle, and lastly provide power protection. In this case a full simulation and later verifying it on the prototype is needed. The coordinated effort of the tracking of OOI is a new technology that needs to be proven in various environmental conditions. The wireless communication needs to be verified in such environmental conditions and ensured at the desired altitude of operation.

Resolve Issues Design Software

Most of the design issues can be resolved using simulation tools for MBSE as well as functional simulations using Matlab, Python and CAD modeling tools. However, to validate the performance specifications that are required to meet certain environmental conditions, simulation development benches need to be built.

Prototype Development

In this section, the systems engineer will identify the subsystems and the components that will present a risk to the development of the project and will require the advanced development of the subsystems and /or the components. In this section, the systems engineer asks mainly three questions as the book suggests:

1) What things could go wrong?
2) How will they first manifest themselves
3) What could then be done to make them right?

Identify Unproven Technology

There are numerous unproven technologies. Table 12 below lists all the issues that need to be developed by the advanced development team.

The objective of this exercise is to design a system that is failsafe to the user and the objects in-site.

Table 12 - Unproven Technologies

Unproven Technology	**What things could go wrong**	**How will it first manifest itself**	**What could be done to make it right?**
Chassis and structure	Not properly ruggedized due to the environmental challenges.	The electronic components and the integrity of the product will be compromised	Materials for the ruggedization shall be identified to withstand the environmental challenges given in a disastrous site. Structural design shall be done to guarantee the stress load on the drone given that it will be in harsh conditions.
Power	Not enough power Not well-designed heat dissipation. Not well shielded wiring harnesses. Not well power distribution and protection	Assets will not work at its full potential. The battery will die in a much shorter time. Overheat of components which may cause fire. Electro-Magnetic Interference(EMI) with other radios or components which will decay or distort the signal with a significant noise.	Simulate the power subsystem by using electrical simulation programs such as pspice or matlab Simulink. Generate the proper modeling for the heat generated by all the components and identify the cooling method and the locations of the vents and how heat will dissipate from the enclosure.
Networking	Not able to communicate with the centralized computer due to	The drone will be completely disconnected from the	Verify that radio frequency and the technology will work in a given

	noise and blocked signal. Not able to communicate with the operator and send visual feedback	scheduling and feedback system.	scenario and at the specified altitudes.
Object Recognition and tracking	Not able to visualize the OOI due to environmental conditions	The OOI will not be lost from all cameras sight	Ensure that the other camera data types will remain locked on target and measure the effectiveness of having a complementing system.
Centralized computer	Not be able to process all the data due to high volume of requests and scheduling	Drone positioning might not be optimized.	Maximize its memory and CPU. Also, create the code to be as highly efficient as possible.

Design and Build Critical Components

In this section, the advanced development team will start simulating the identified items above, as well as they will start developing the prototype for this project.

Development Testing

In this section, the advanced development team along with the systems test engineer will be developing the test cases that will determine that all of the design issues identified during the advanced development phase have been satisfactorily resolved by doing a simulation, or building a prototype. Another objective for this development testing is to assure that the subsystem interfaces are properly selected. During this testing phase, the system will be tested against the upper and the lower limits. This will give the team a high confidence that the system will properly work in the specified environment. One of the more important reasons for the advanced development team to build the prototype is to determine the cost and the manpower required to

build and test the system. Lastly, this step will help the systems engineers, and the test engineers developed a detailed TEMP document which will be described in section 6.0 below. Kossiakoff stated that a well-planned development test program generally requires the following procedures.

1) development of a test plan, test procedures, and test analysis plan;

2) development or acquisition of test equipment and special test facilities;

3) conduct of demonstration and validation tests, including software validation;

4) analysis and evaluation of test results; and

5) correction of design deficiencies.

This test is normally done by the developers along with some test engineers and normally it is done at the unit level then at the subsystem level, in other words, there are unit tests followed by integration tests that validate that all components are working properly together.

Build Test Setup

In this section, the test cases for the advanced development team will be created to verify the risk presented above has been addressed with high confidence. Table 13 below shows the test environment needed to meet all the customer requirements.

Evaluate Test Results

In this section, the tests will be evaluated and reported upon completing all the simulations. The evaluation is normally documented to ensure the proper traceability of all the risks presented at the beginning of this phase. For any deficiencies, the advanced development team will suggest corrective actions that can be re-prototyped or simulated to close the risk items.

Table 13 - Test Environment

Subsystem/component	Equipment Facilities needed	Test functions
Chassis and Structure	Stress facility High-temperature chamber Low-temperature chamber Vibration benches Explosive chamber High humidity chamber Low humidity chamber Electrical shock bench Electro-Magnetic Interference chamber Salt and humidity environment Hydrostatic chamber Radiation chamber Shock benches	This test will verify that the structure and enclosure will be able to withstand the harsh environmental conditions.
Power	Electrical shock bench Stress test	This test will allow the system to meet the electrical shock test in MIL-STD810G. The stress test will operate the prototype for an extended period of time on a mission case for testing purposes. This test will verify the battery life and the functionality over a long period of time.
Networking	Restrictive chambers and facilities to radio signals that mimic various altitude and. combined environmental challenging conditions.	This will verify that the networking using the proposed Radio frequencies will meet the mission demands.
Object Recognition and tracking	Simulated OOI site condition at various speeds and directions.	This test will verify that the object detection and tracking system will properly operate in the specified conditions.
Centralized computer	ATP for request submittals to simulate drone requests and simultaneous uploads	This test will allow testing the centralized computer under a stress test.

Software Systems

The software system for this project is composed of:

1) Hardware embedded software and firmware
2) Real-time Operating System
3) Application program and backend software
4) Data structures and data storage
5) User Interface.
6) Documentation

The application systems engineer will be responsible to create the software requirements whereas the embedded software and hardware firmware will define the protocol for motors, sensors, on-board processor, and on-board memory to process the data for autonomous capability. The coding language recommended is C since it is a universal language; thus, it can be portable and the drones' capabilities can be upgraded later.

There are two operating systems for this project. The first operating system is the on-board Real Time Operating System (RTOS). This operating system is the one that is responsible to handle all the autonomous scheduling between all the components of the drone. It also receives and schedules operations from the centralized computer. The second type of operating system is a distributed operating system. This is the operating system that is running on the centralized computer where it allows each drone to have its own RTOS while scheduling the main tasks and receives and handles all the data coming from the field assets. The recommended RTOS is: RTLinux since it is a microkernel that runs the entire Linux OS as a fully preemptive process.

The application software and the backend. Is the software that is running in the background that handles all the communications schemas. The recommended language for this backend software is C++ since it is one of the mostly efficient languages. Also, since the front end, which will be discussed shortly, needs to communicate with the centralized computer. A good web request can be done using the Simple Object Access Protocol (SOAP). To build the SOAP functions, an automatic translation from the schema can be done using the C++ language.

The data structures and data storage are needed to handle all the submitted data from all the drones, which will allow the centralized computer to re-task the drones properly based on the predictive analysis of the OOI. The recommended data structures are Network Attached Storage (NAS) devices as well as databases that associate the location on the map with the path to the files.

The user interface is what allows the user to see the latest site, print it, see what the drone is seeing real-time, and command it to perform certain actions. Upon requesting the drone to perform a certain action, this command will be delivered to the centralized computer to properly schedule the drone to perform the action. This allows for data coherency of the whole system. The recommended user interface is Hyper Text Markup Language -5 (HTML-5) which allows the developer to update the content and the look and feel of the user interface more easily.

The documentation shall use the UML tools to explain the use cases, the structure of the software, the behavioral and the interactional cases with the software.

The software shall use the agile life-cycle which allows the developers to release the software on an incremental basis fully tested and qualified. The duration for each scrum is recommended to be 2 weeks and a release product every 3 scrums. The product owner and the

scrum masters are encouraged to change the scrum period. Some of the software systems will be prototyped in the next chapter.

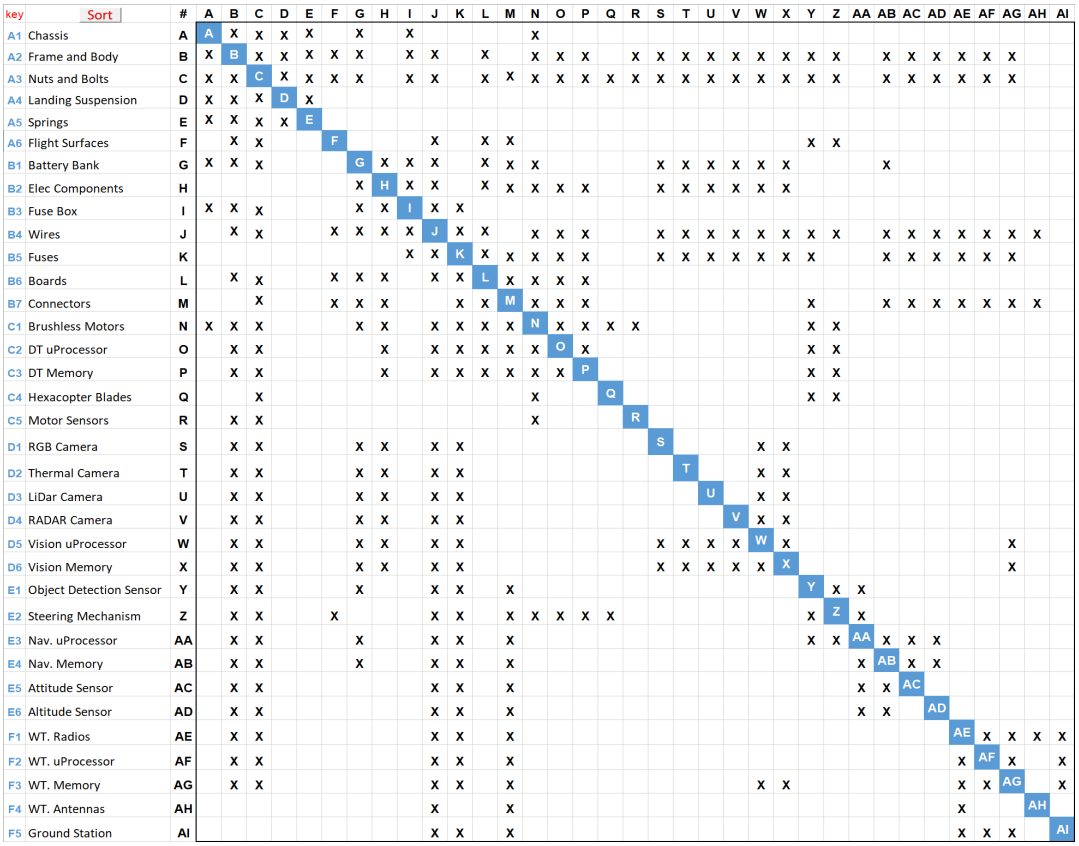
Engineering Design

The objective of this phase is to design systems components to the desired performance, cost and schedule per the requirements. In this phase all interfaces and interactions are well identified and defined. The system is configured on a modular base where it groups the tightly coupled components together in a single module. In this phase all components are designed and prototyped. During this phase there are two major design reviews: Preliminary Design Review (PDR) and Critical Design Review (CDR)

In this step all the interfaces and the interactions are identified. Normally the design starts with a preliminary concept diagram as was shown on Figure 4 in the previous section. Then the interfaces and interactions are identified on the block diagram. The obvious external interface is the application specific radio, which allows the communication with the central computer, or sends visual real-time feedback to the user. The second obvious external interface is the camera, and sensory components. This allows the drone to gather data and imagery about the OOI. There are numerous internal interfaces, since the objective is to modularize this system for maintainability. Also, modularizing the drone will increase the reliability since the system will be partially mission-capable upon a failure of a component. These interfaces are indicated by the arrows to each component listed below. Some of these interfaces include Ethernet connectors, J-type connectors and wireless interfaces.

A great tool that I learned from the systems engineering certificate I achieved from MIT is to use the Design Structure Matrix (DSM). This tool allows the system engineer to visualize

and to modularize the subsystems into smaller modules, as well as the interfaces between the modules. The DSM for this project is shown on Figure 5 below.



key	Sort	#	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI
A1	Chassis	A	A	X	X	X	X		X		X					X																					
A2	Frame and Body	B	X	B	X	X	X	X	X		X	X		X		X	X	X		X	X	X	X	X	X	X	X	X		X	X	X	X	X	X		
A3	Nuts and Bolts	C	X	X	C	X	X	X	X		X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X		
A4	Landing Suspension	D	X	X	X	D	X																														
A5	Springs	E	X	X	X	X	E																														
A6	Flight Surfaces	F		X	X			F				X		X	X												X	X									
B1	Battery Bank	G	X	X	X				G	X	X	X		X	X	X					X	X	X	X	X	X				X							
B2	Elec Components	H							X	H	X	X		X	X	X	X	X			X	X	X	X	X	X											
B3	Fuse Box	I	X	X	X				X	X	I	X	X																								
B4	Wires	J		X	X			X	X	X	X	J	X	X		X	X	X			X	X	X	X	X	X	X	X		X	X	X	X	X	X	X	
B5	Fuses	K									X	X	K	X	X	X	X	X			X	X	X	X	X	X	X			X	X	X	X	X	X		
B6	Boards	L		X	X			X	X	X		X	X	L	X	X	X	X												X	X	X	X	X	X		
B7	Connectors	M			X			X	X	X			X	X	M	X	X	X									X			X	X	X	X	X	X	X	
C1	Brushless Motors	N	X	X	X				X	X		X	X	X	X	N	X	X	X	X							X	X									
C2	DT uProcessor	O		X	X					X		X	X	X	X	X	O	X									X	X								
C3	DT Memory	P		X	X					X		X	X	X	X	X	X	P									X	X								
C4	Hexacopter Blades	Q			X											X			Q								X	X								
C5	Motor Sensors	R		X	X											X				R																
D1	RGB Camera	S		X	X				X	X		X	X								S				X	X										
D2	Thermal Camera	T		X	X				X	X		X	X									T			X	X										
D3	LiDar Camera	U		X	X				X	X		X	X										U		X	X										
D4	RADAR Camera	V		X	X				X	X		X	X											V	X	X										
D5	Vision uProcessor	W		X	X				X	X		X	X								X	X	X	X	W	X									X	
D6	Vision Memory	X		X	X				X	X		X	X								X	X	X	X	X	X									X	
E1	Object Detection Sensor	Y		X	X				X			X	X		X												Y	X	X							
E2	Steering Mechanism	Z		X	X			X				X	X		X	X	X	X	X								X	Z	X							
E3	Nav. uProcessor	AA		X	X				X			X	X		X												X	X	AA	X	X	X				
E4	Nav. Memory	AB		X	X				X			X	X		X													X	AB	X	X					
E5	Attitude Sensor	AC		X	X							X	X		X													X	X	AC						
E6	Altitude Sensor	AD		X	X							X	X		X													X	X		AD					
F1	WT. Radios	AE		X	X							X	X		X																	AE	X	X	X	X
F2	WT. uProcessor	AF		X	X							X	X		X																	X	AF	X		X
F3	WT. Memory	AG		X	X							X	X		X										X	X							X	X	AG		X
F4	WT. Antennas	AH										X			X																				AH		
F5	Ground Station	AI										X	X		X																			X	X	X		AI

Figure 5 - DSM Module

Integration and Evaluation

The objective of the integration and evaluation phase is to test and evaluate the system performance and the system-emergent behavior meets all the operational requirements. During this step the system will be validated and verified. In the validation step the system will be tested against the requirements of the customer to guarantee that it meets the mission needs of the customer. In other words, a given scenario will be built with a known ground truth table provided then a quantitative accuracy measurement will be performed against all system components such as the user interface, the centralized computer, the drones data gathered and their prediction on

all the measurements against the ground truth. This step will involve acceptance and suitability with external customers. In the verification step the whole system will be tested against the requirements as a formal test with the presence of the Quality Assurance team. The test procedure will be written by the systems test engineers according to the requirements document. Each requirement will be mapped to a test case. During this phase the Test and Evaluation Master Plan TEMP document will dictate the plan for the testing criteria. A simplified TEMP document to show the sections and the potential content of each section.

- **System Introduction**

The network of autonomous drones will assist the S&R team with the aforementioned applications. The drones also will send visual feedback to the user.

- **Mission Description**

  The mission for this document is to describe what is planned to be done at each stage of the development of the system.

- **Operational Environment**

  The Operational Environment will be described in this section where the normal operation will be specified. However, the environmental testing will be completely outside of the operating range. Normally, it specifies a standard to meet, such as MIL-STD810G as described by the requirements stated above.

- **Measure of Effectiveness and suitability.**

  The measure of effectiveness and suitability is described by the proper operation of data retention and recording capability during all the extreme testing described

by MIL-STD810G, as well as the ability to communicate real time with the centralized computer.

- **System Description**

  In this section is described the functional and operational system description. As well as the functional decompositions of the subsystems.

- **Critical Technical Parameters.**

  This section will identify all the critical technical parameters that the system needs. Such as the number of channels, input, output, buffer range, expected message flow in the worst case, restricted access and loss of communication with the centralized computer.

- **Integrated Test Program Summary**

  - **Test Program Schedule Management**

    In this section the schedule will be set for all the testing procedures. In this section it will indicate how it will be weaved into the development process. For instance, all the unit testing will be done after some sprints in case the development is following an agile process. Then some integration testing will be done at every major release. Going back to the project on hand, it is following an agile process with every other week we have an iteration. During this iteration, a hardware system is built, unit tested and the associated software is built and unit tested. Every 3 iterations we have a major release. In this release a major functionality of the system is integrated and tested. In a way, the testing is following the right

branch of the V model. During the integration testing, the unit undergoes environmental testing according to MIL-STD810G and functional testing of the whole functionality. Any undeveloped software or hardware driver/stubs will be created to emulate the undeveloped sections of the project.

- **Participating Organizations**

  In this section, all the participating organizations will be specified. This includes the testing team, the validation team, the simulation team, the systems engineering team including testing the hardware team, the software team, the human factors team, and safety and reliability and quality assurance (QA) team.

- **Developmental Test and Evaluation**

  - **Method of Approach**

    In this section, the method of testing will be developed. For instance, some of the algorithms will be simulated or emulated using driver and stubs software because the hardware sometimes is not in line with the software. Therefore, the software emulators will be used to temporarily replace the hardware until it is done. Therefore, this section will be describing the methods that will be used.

  - **Configuration Description**

    In this section, the configuration will be specified for the test and evaluation for each release. This will hold the whole system configuration of the test. This is the configuration that was agreed on at the Critical Design Review (CDR).

- **Test Objectives**

  In this section we specify the objectives for this test, for instance, the test will be used for integration of the specific release, durability test, unit test. Each test section will be specified.

- **Event and Scenarios**

  This section describes the events used for each test. As well as scenarios in which the system will be properly operating outside the specified range, it has a failsafe in case of a specific event to not cause harm to the operator, or anything on-site.

- **Operational Test and Evaluation**

  - **Purpose**

    The purpose of this section is to specify the test needed to guarantee the operational requirements.

  - **Configuration Description**

    Similar as above but to guarantee the operational needs.

  - **Test Objectives**

    Similar as above but to guarantee the operational needs.

  - **Event and Scenarios**

    Similar as above but to guarantee the operational needs.

- **Test and Evaluation Resource Summary**

- **Test Articles**

  This section will specify the articles of the requirements that will be tested against. This is normally created and traced in doors.

- **Test Sites**

  This section will specify a special location that needs to be used such as a building that has a deep pool for compression and submersion testing, or another building for explosive testing, or specify a certain vibration bench, or in a certain location such as a simulated operational site.

- **Test Instrumentation**

  This section will specify the tools needed: such as a multimeter or spectrum analyzer.

- **Test Environment and sites**

  Similar as above in a greater level of detail.

- **Test Support Operations**

  Personal needs and support such as simulator support personal

- **Computer Simulations and models**

  In this section, we can specify the requirements for a given simulator that will allow us to properly test the system.

- **Special Requirements.**

In this section, we specify any special requirements for the testing such as humidity level, or any other condition needed.

Production

The production phase objective is to lay the groundwork to be able to reproduce identical hardware and software of the prototype developed for this project. The requirements for the tooling, facilities, and the technology that will be used for production are normally defined concurrently during the previous phases of the design. Other than the facilities needed for the testing, the facility for production will require a set of stations where the production can be pipelined to maximize the throughput. Each stage will be responsible for a subsection of the system all the way to final assembly and full integration phase. These stages are going to be defined after the simulation of the most efficient critical path. All the tooling and the equipment will be well defined after finalizing the configuration. All the parts will be taken from cutout trays to reduce any missing parts to the final product. A full design should have been done concurrently to the project and include in the design phases the appropriate facility and tooling experts to better define and simulate the production facility.

Operation and Support

The objective of this phase is to define the operation and support of the lifecycle of the product including sustainment and modernization. This product can be offered through three packages: the first package is the silver package where the customer buys the system and he/she is responsible to maintain the system and operate it. The second package is the golden package where the customer buys the system with limited maintenance. The third package is the platinum package where the customer buys the system including full maintenance and full support, data analysis and upgrades by paying a monthly fee; in other words, it is a turn key

service. These packages are designed to attack the various levels of customers. For the platinum package, our company will be responsible for the operation and support including installation and assembly, test, in-service support and upgrades. During the fielding and sustainment of the product, a detailed analysis will be collected to incorporate the field issues in the second generation of the autonomous network of drones.

CHAPTER 2 – SYSTEM OVERVIEW AND ARCHITECTURE[1]

Introduction

Object tracking is an active research area in computer vision thanks to the increasing demands in the Intelligence, Surveillance and Reconnaissance (ISR) applications and the Autonomous Vehicles Systems (AVS). The tasks of computer vision object tracking consist of: Image sensing, image enhancement, background extraction, object classification, tracking of the object of interest and feedback analyzer. To facilitate the development process, a visual sensing system is used; however, it is recommended to use a quadruple redundant system such that they complement each other. This quadruple redundant sensory system is composed of LiDAR, Visual Camera (RGB), and Thermal Camera, and RADAR sensor the performance of each system is shown on Figure 6a. The Web-graph on Figure 6b shows the combined performance of the four sensors and shows how they complement each other.

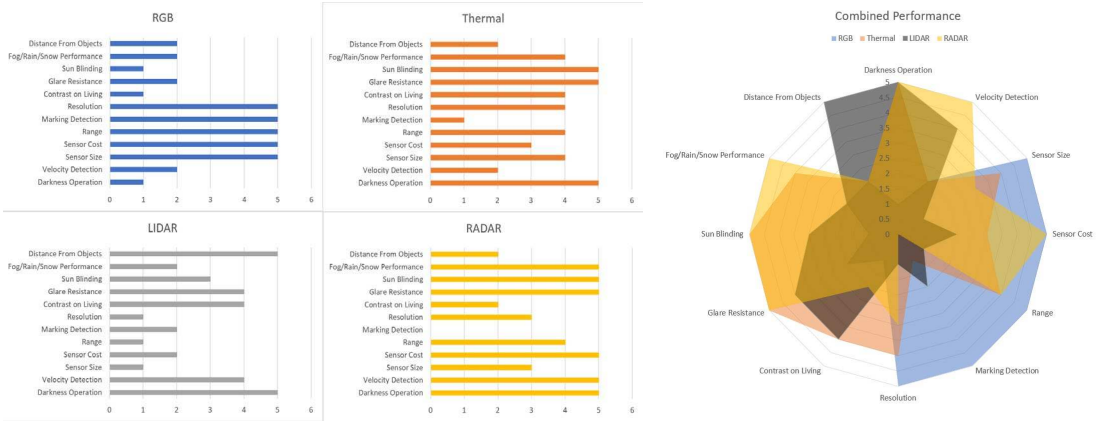


Figure 6 - (a) Performance of various sensors, (b) Sensors combined performance

[1] H.Williams and S.Simske, "Object Tracking Continuity through Track and Trace Method," *IS&T,* 2020

System Overview

The proposed system will span across three phases as listed below; however, this dissertation will mainly cover the first two phases of the design.

- Phase-1: A single static camera that observes the seen,
- Phase-2: Multi static cameras that slightly overlap their fields of view,
- Phase-3: A Moving camera, where the system controls the 6 degrees of freedom of the camera source assuming it is fixed on a rigid body as shown on Figure 7 below.

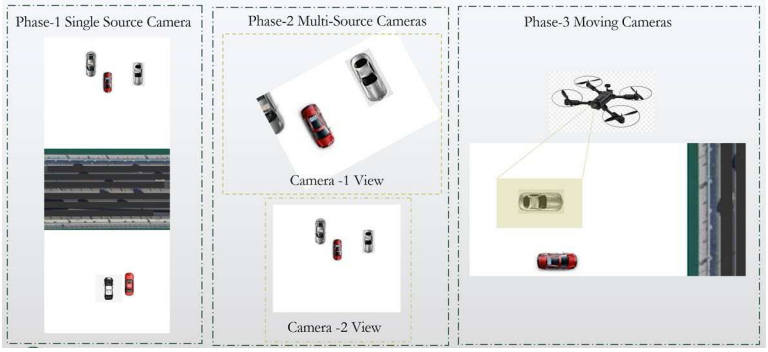


Figure 7 - Three Design Phases

The modules are divided as the tasks mentioned above to facilitate the enhancement of any of the subsystems independently.

- The image sensing requirements dictates the camera technology to be used. The camera technology is not limited to only the resolution and the field of view of the camera but also the frequency range for example the frequency requirements could be outside of the visual range such as in the microwave and infrared range in the military application and in the x-rays or higher in the medical application.
- The image enhancement also referred to as the image processing module is responsible for noise removal, sharpening, deblurring, and normalizing the image.

- The background extraction can be achieved with background subtraction; however, a more elaborate subsystem needs to be in place to account for the dynamic scene changes for instance, changing in illumination, shadow casting. In static the cameras, the system shall account for subtle changes as part of the background such as flying flag or tree branches moving; however, moving camera, on design phase-3 the system shall account for moving background.
- The object classifier is responsible to detect and recognize the object of interest with a machine learning algorithm. These algorithms can be chosen from many different algorithms such as Convolutional Neural Networks (CNN), Support Vector Machines (SVM), or statistical based models such as the likelihood ratio.
- The tracking subsystem is responsible for predicting the next location of the Object Of Interest (OOI) based on the previous trajectory. Part of the tracking subsystem is gathering enough data from the camera mesh system to simultaneously localize and map using a SLAM (Simultaneous Localization and Mapping) system. This will enable the devices to properly route track the OOI over the geographical maps which in return will enhance the overall route prediction. The drones SLAM function will also enhance the photogrammetric quality of the reconstructed 3D scene.
- The feedback analyzer assigns the figure of merit to the system.
- The camera controller decides which camera to turn on to keep OOI in view for phase-2 and controls the vehicle in phase-3.
- Lastly, the camera-correlator performs the affine transformation between the various cameras field of view in Phase-2.

Figure 8 shows the system overview of system for Phase-1 which is discussed in this paper. To better control the scenario / testing the scene was synthesized. This paper discusses the related work, the theory of each subsystem, then follows with the results.

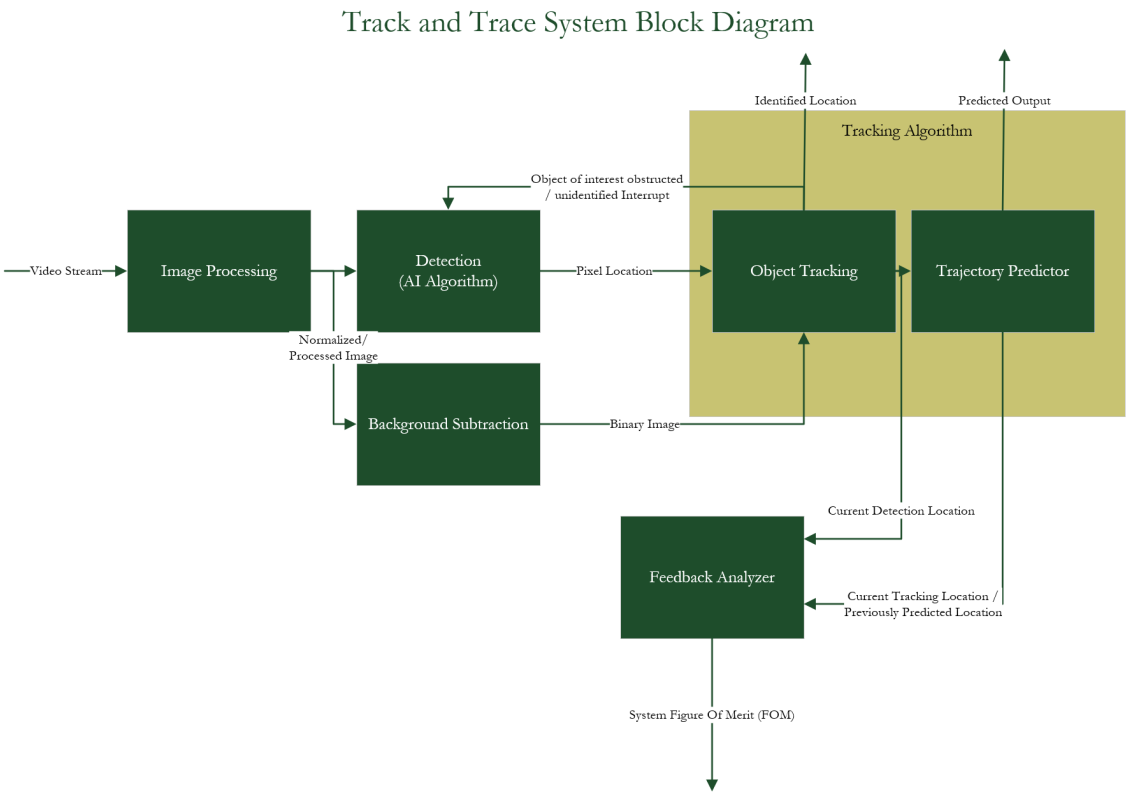


Figure 8 - System Block Diagram

Image Processor Subsystem

The input to the image/video processing subsystem is a video steam. The input is separated into image frames, where each frame is normalized, histogram equalized, deblurred and sharpened. Then each frame is assigned to a red, green and blue channel as well as the hue, saturation and intensity channels. This allows for a custom usage of each channel in the detection and the tracking subsystem. Thus, the output of this subsystem is the processed RGB and HSI

subframes channels as well as an image enhanced grayscale. Figure 9 shows the block diagram of the image processor subsystem.

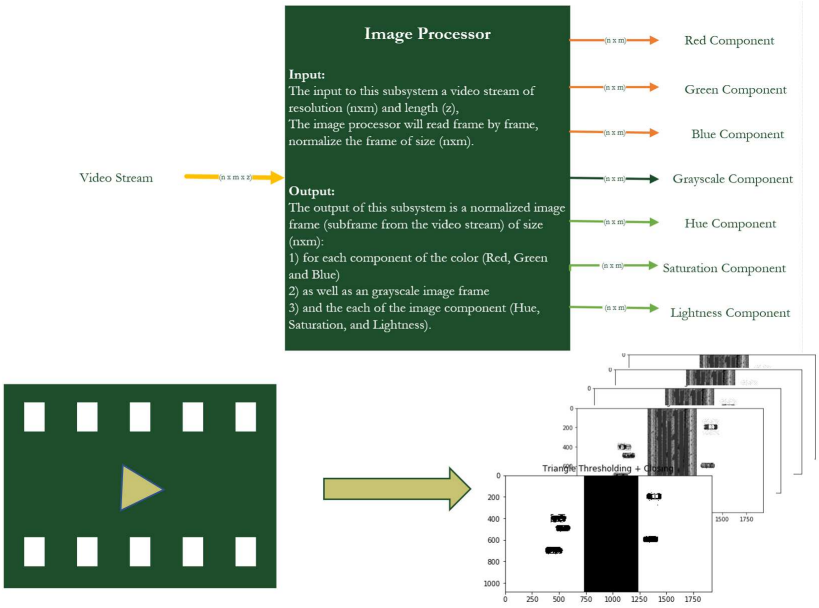


Figure 9 - Image Processor Subsystem Block Diagram

Background Subtraction Subsystem

The Background Subtraction[2] subsystem takes the output of the prior subsystem which consists of the RGB and the HSI subframes and performs edge detection, and mathematical morphology functions and thresholding also known as opening, closing, erosion and dilation on each subframe to extract the object in the scene. This subsystem also performs segmentation such as watershed function to separate multiple overlapping objects. The result is then subtracted from the base image to extract the moving objects from the fixed ones. At a later stage, another function will be added to remove subtle movements that are based on repetitive temporal-spatial characteristics such as flying flag, or tree branches moving to better perform in the outdoor

[2] L. Maddalena and A. Petrosino, "A Self-Organizing Approach to Background Subtraction for Visual Surveillance Applications," *IEEE,* vol. 1057, no. 7149, pp. 1169-1177, 2008.

environment. Figure 10 below shows the output of this subsystem which is a frame that consists of only the moving objects.

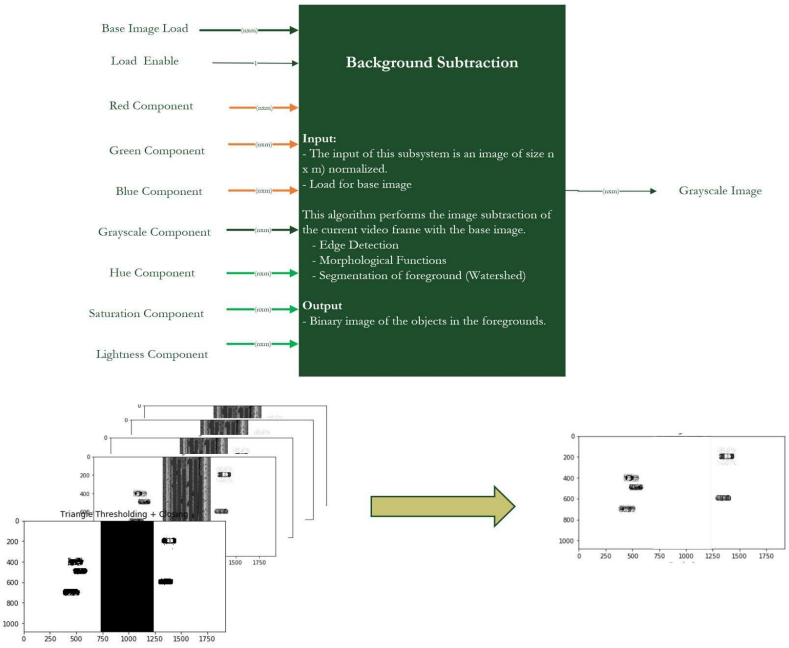


Figure 10 - Background Subtraction Block Diagram

Detection Subsystem

The input to the detection subsystem is the output of the image processor subsystem which consists of the RGB and the HSI subframes. This subsystem is responsible to locate the Object Of Interest (OOI) in the scene. Later, this block utilizes a variety of custom pre-trained learning algorithms that the user can select; however, up to this point a Convolutional Neural Network (CNN) was developed and is presented in this paper. This subsystem outputs the centroid of the OOI. The centroid is calculated based on the binary image of the OOI shape; thus, it is morphology-based calculation of the centroid. This subsystem only executes at the beginning when the system gets powered up and locates the OOI or when an interrupt occurs. The interrupt occurs if the object tracking subsystem fails to locate the OOI due to obfuscation of the OOI in the scene. Figure 11 below shows the block diagram of the detection subsystem.

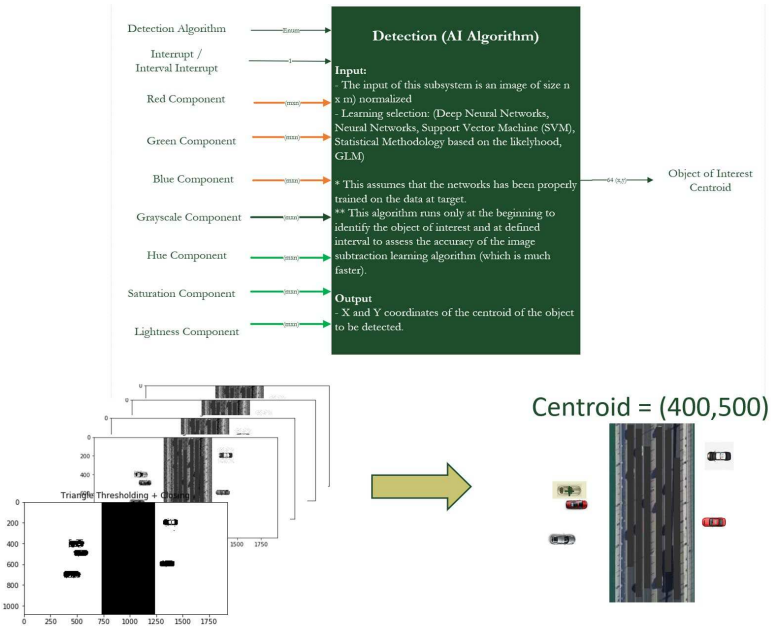


Figure 11 - Detection Subsystem Block Diagram

Object Tracking Subsystem

The input of the object tracking subsystem consists of the OOI centroid that was previously calculated by the detection module, the OOI valid bit, and the grayscale image output from the background subtraction module. The subsystem consists of multiple tracking algorithm that are native to openCV. Some of these algorithms are Boosting, Multiple Instance Learning (MIL), Kernelized Correlation Filter (KCF), Tracking and Learning Detection (TLD), CNN tracker (GOTURN), Minimum Output Sum of Squared Error (MOSSE)[3] and Discriminative Correlation Filter also known as DCF-CSR. All these algorithms and their performance will be compared and contrasted in a later paper. A compressed version of the OOI will be used to expedite the process. In this paper, this subsystem is not developed yet. The output of this subsystem consists of the binary centroid of the OOI and a validity bit that indicates that the object has been found by one of the algorithms stated above. The main difference of this module

[3] D. Bolme, R. Beveridge, et al. "Visual object tracking using adaptive correlation filters" *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2544-2550, 2010.

and the detection module is that this module is dependent on a temporal knowledge based on the multiple frames; thus, it performs faster than the detection module. Figure 12 below shows the block diagram of this subsystem.

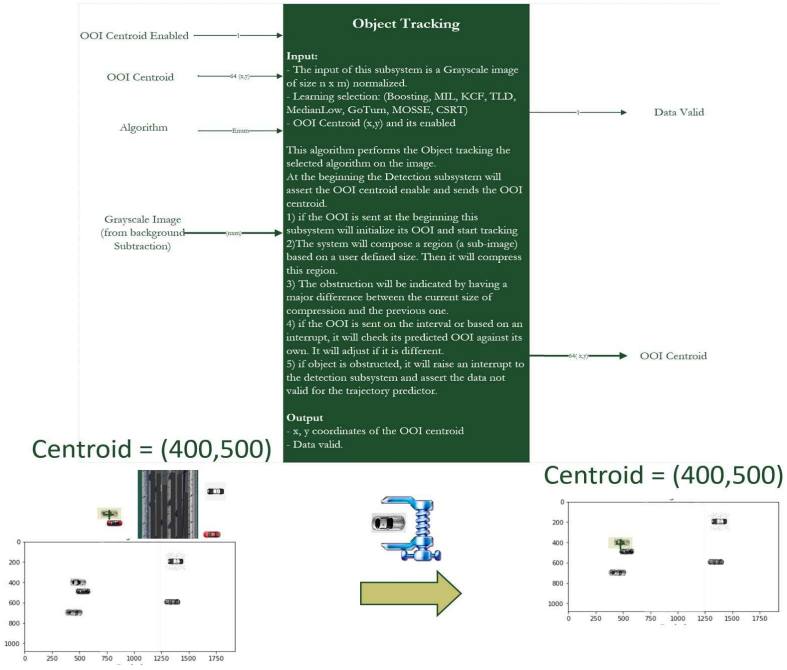


Figure 12 - Object Tracking Block Diagram

Trajectory Predictor Subsystem

The inputs to this module are the centroid of the OOI, the validity bit, and a user defined number that dictates the amount of time to extrapolate the trajectory path. This subsystem stores the discrete centroid location of all open hypotheses then performs a cubic spline interpolation to extrapolate the prediction to the amount of time requested by the user. This interpolation curve describes the characteristics and the behavior of the OOI which assist in building a model for the OOI. Figure 13 shows the block diagram of the trajectory predictor subsystem.

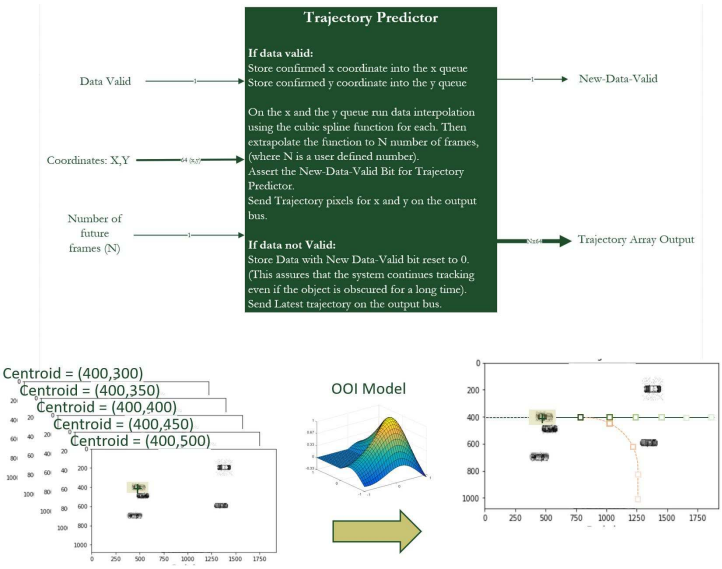


Figure 13 - Trajectory Predictor Block Diagram

Feedback Analyzer Subsystem

The input to this subsystem is the coordinates of centroid and the predicted trajectory calculated from the trajectory predictor module. In this subsystem, the accuracy of the overall system gets accessed by comparing the trajectory to the detected module, then a figure of merit gets assigned to each coordinate as Figure 14 shows below in the block diagram.

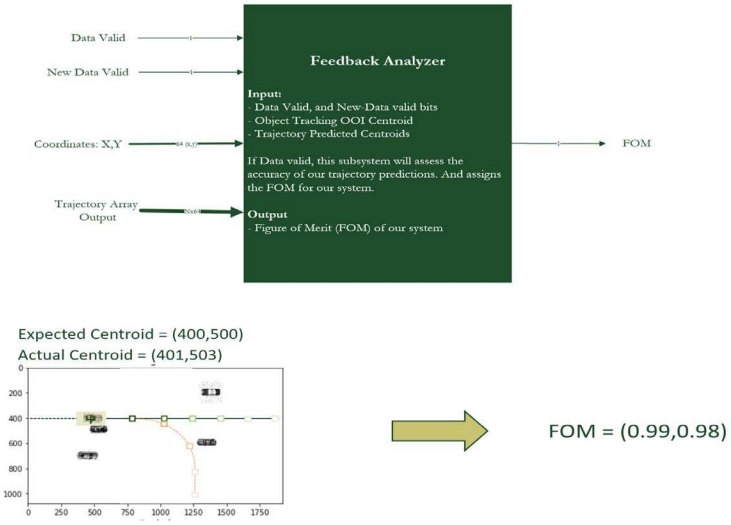


Figure 14 - Feedback Analyzer Block Diagram

CHAPTER 3 – OBJECT RECOGNITION USING DEEP NEURAL NETWORK

Introduction

Proper object recognition and classification is an essential task in object tracking. Traditionally, object recognition has been performed using three specific stages: The first is object localization which is done by a sliding multiscale window. Understandably, this is a computationally challenging task. The second stage is to perform feature extraction using Scale-Invariant Feature Transformation (SIFT), the Histogram of Oriented Gradients (HOG), a Hough Transformation or other digital signal/image processing techniques which mainly compares the most remarkable features to features in a known database from the supervised learning phase. Lighting, orientation and partial obfuscation pose challenges to these techniques. The last stage is to utilize Support Vector Machines (SVM) to make the representation more hierarchical and semantically descriptive. These traditional techniques are very successful in simplistic scenes; however, in more complex scenes – with lighting, shading, partial obfuscation, and orientation challenges – these techniques face multiple drawbacks. For this reason, more modern techniques have emerged. Convolutional Neural Network (CNN) is one of the modern techniques that showed remarkable results in the last decade. Some of the CNN algorithms combine the main three stages discussed above: localization, feature extraction and the classification within the algorithm itself such as RCNN. Other CNN algorithms combine only the latter two.

The Convolutional Neural Network is composed of the following functions and subfunctions as Figure 15 shows below: [4]

[4] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

- Feature Learning Section
    - Convolutional layers
    - Pooling layers
- Classification Section
    - Flatten Layer
    - Fully Connected Layer
    - SoftMax Layer

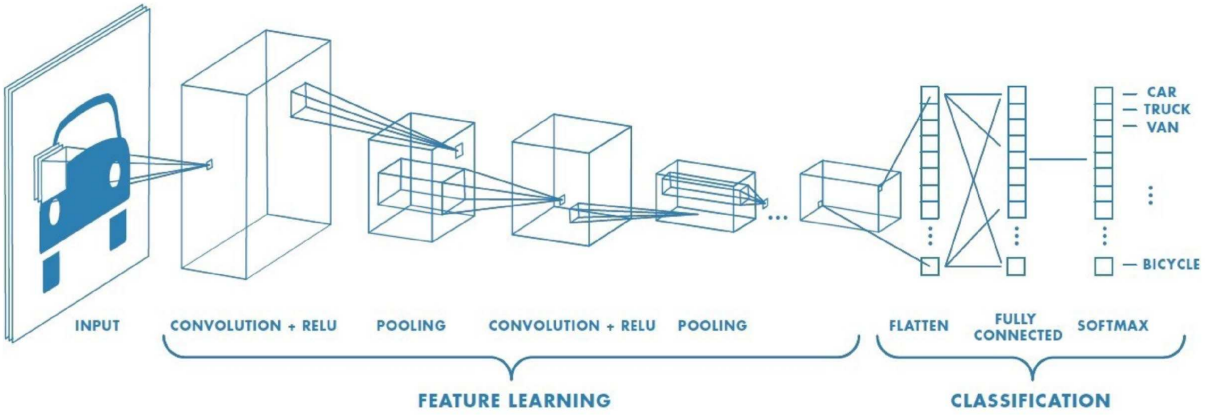


Figure 15 - Convolutional Neural Network

The Feature Learning Section is composed of a series of Convolution and Pooling Layers. The Convolution Layer is responsible for extracting the high-level features in the image such as the objects and shapes; this level of extraction is not normally done in one convolutional step; thus, an effective CNN is composed of multiple convolutional layers. The first layers are responsible for low-level features such as dots, lines, edges, color, orientation, then the more layers added, the CNN captures and extracts a more refined set of features referred to as high level features. The second main operation in the feature extraction is called the pooling layer. The pooling layer is responsible for reducing the spatial size of the features. This is mainly to reduce the computation complexity and extracts only the more dominant features through

dimensionality reduction techniques. In other words, pooling is responsible for noise suppression.

The Classification section is composed of a flattened layer, fully connected layer and a SoftMax layer. The flattened layer and the fully connected layer create a non-linear combination of the high-level features extracted by the prior section. Lastly, an activation function such as SoftMax is used as a logistic function to normalize the output of the network to a probability distribution over the various classes.

This chapter shows various CNN architectures and compares their performance; it also shows a technique in combining these architectures as an architectural collaborative approach to enhance the results. The dataset that is used to perform this benchmark is the Stanford Cars dataset, which is discussed in detail in the next section.

Stanford Cars Dataset

The Stanford Cars dataset is available to be used for research purposes only, similar to the ImageNet license. It comprises 16,185 images of 196 classes of cars as shown on Figure 16 below.

Figure 16 - Stanford Cars Dataset Images

These images are taken in various lighting conditions, backgrounds, size, orientation and even motion blur, all these conditions create a great representation when trying to resolve recognition in real world applications. Examples of these conditions are shown on Figure 17.[5]

Figure 17 - Example of Image Conditions

The dataset is composed of the aforementioned images, and development kit which contains a table with the following fields:

- File name,
- Corresponding class label
- Bounding box for the car in the image.
- Training/Testing label suggestion field.

Approach

Image Processing and Enhancement for Training

To prepare the image for the CNN training, each of the images was cropped according to the bounding box as specified in the dataset development kit. This allows the training algorithm to properly learn the object of interest features of each class without contaminating the features

[5] http://ai.stanford.edu/~jkrause/cars/car_dataset.html

with objects in the background or the surrounding including features of other cars in the background.

Since the learning used in this approach was the transfer learning methodology, the size of the images had to follow the image size previously specified by the algorithm. Thus, the next step in the preparation is to change the size of each image according to the algorithm as described by Table 14.

Table 14 - Algorithm Image Sizes

Algorithm	Image Length	Image Width	# of Channels
VGG16	224	224	3
VGG19	224	224	3
DenseNet201	224	224	3
NASNetLarge	331	331	3
InceptionV3	299	299	3
Xception	299	299	3
InceptionResNetV2	299	299	3
MobileNetV2	224	224	3
ResNet50V2	224	224	3
ResNet152V2	224	224	3
ResNeXt50	224	224	3

The size adjustment also created an effective data augmentation since the features of each class was changed due to the compression or expansion of the shape differently and dictated by the pose of the object in the image.

The last step in the image preparation was to change the dynamic range of the images and perform histogram equalization. Lastly, all the images were normalized.

The data was assigned to a training (80%) and a testing (20%) set. From the 20% of the testing set, 15% was used for validation and the 5% was used for final testing.

Data Augmentation

Due to the fact that the training data is small for some of the non-robust algorithms, data augmentation was necessary to accomplish an effective training. Thus, the following data augmentation were performed to increase the data size:

- Random Crop (between 10% to 15%) from either direction randomly
- Horizontal Flip
- Vertical Flip
- Rotation (between -15 degrees to 15 degrees)
- Shear Distortion (between -10% to 10%)
- Change the Hue and Saturation of the images (between -20% to 20%)
- Apply Gaussian Blur (between 0 to 1.5)
- Apply Sharpening (between 0.7 to 1.3)
- Apply Embossing (between 0 to 1.5)

Learning Algorithm Architecture

Initially, the architecture of the Convolutional Neural Network was constructed without any prior knowledge of the images or classification using a blank VGG16 sequential model. The result was understandably unsuccessful, the optimum accuracy achieved was less than 1%. Thus, I resorted to the transfer learning methodology.

Transfer learning is a machine learning technique where the model that was developed for a specific task is partially reused for another task. In other words, the transfer learning methodology allows to take features learned on one problem and leveraging for a similar problem. This is normally done by freezing the top layers of the model and training only the last layers of the model. In the next sections all the architectures discussed below used transferred

learning with the initial weights of the training performed on the imageNet dataset. Lastly, the last dense layers were unfrozen to allow the model to be trained on the distinct features of the 196 classes of cars. This methodology achieved much higher accuracy, as shown by each architecture below.

Another methodology employed in our application which contributed to the faster learning was the cyclical learning rates for training neural networks[6]. In traditional learning rate, it is not a trivial task to choose the optimal initial learning rate; also, it is difficult to monotonically change the rate because it may plateau (asymptote at a non-global optimum) or jump over the minima. Thus, the cyclical learning rate was developed by Leslie Smith in 2017 to address these shortcomings. As discussed in Smith's paper, the most important hyper-parameter to tune for training neural networks is the learning rate. In the referenced paper, a new method named *cyclical learning* was developed to find the best values and schedule for the global learning rates by cyclically varying the rate between two reasonable rates; namely, the base learning rate and the maximum learning rate. Initially the learning rate is very small, over time, the learning rate increases until it reaches the maximum learning rate (Step Size). It then cycles back down to the minimum value again and continues cycling throughout the training as shown on Figure 18. Another variation of this method is to decrease the Maximum Rate either linearly or logarithmically which allows the algorithm to reach lower loss areas.

[6] L. Smith, "Cyclical Learning Rates for Training Neural Networks" *IEEE Winter Conference on Application of Computer Vision (WACV),* 2017 pp.464-472

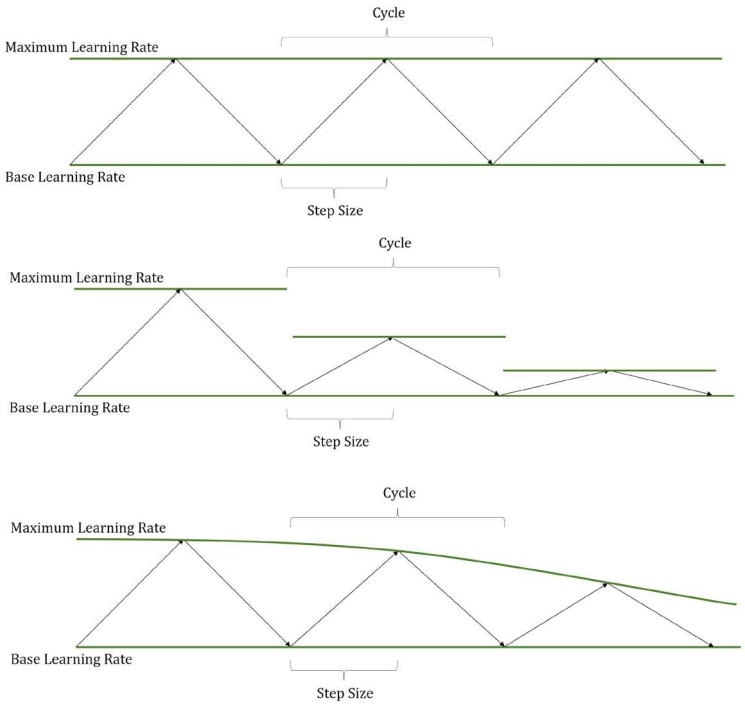


Figure 18 - Cyclical Learning Rate (a) Fixed Rate, (b) Linearly Decaying Rate, (c)Log Decaying Rate

The cyclical learning rate method promised to achieve a higher accuracy in just a few epochs. The result of this learning rate is clear in each of the accuracy graphs discussed in the next sections. Each architecture model achieved at least 80-90% accuracy within the first 4 to 6 epochs.

Our learning algorithm was developed in python with the configuration and package versions listed in appendix 1. The learning algorithm has the following parameters: for the cyclical rate, the base learning rate was set to $1e^{-7}$ and the maximum rate was set to $2e^{-4}$ the step size was set to be 4 times the training size / batch size. The learning rate was set to be logarithmically decaying. All the algorithms below used the transfer learning CNN with the initial weights of the imageNet. Lastly, the top layers were frozen from the training; thus, only the later dense layers were unfrozen for the training. The next section discusses each of the architectures used and the results that were achieved. The last section discusses a collaborative

technique that was developed to reduce the error even further. To check the learning and the progress of the training, the images were divided into 5 folds, where each fold contains different images from the dataset. This enabled a proper average of the accuracy across the various architectures.

VGG16 Convolutional Neural Network

VGG16 Architecture

The VGG16 is a convolutional neural network model that was proposed by the Visual Geometry Group (VGG) at Oxford University by Simonyan and Zisserman in 2015 in the paper "Very Deep Convolutional Network for Large-Scale Image Recognition"[7] when it was one of the first networks that achieved remarkable results for the ImageNet dataset (which is composed of roughly 14 million images from 1000 classes). The main premise of the VGG architectures is that the convolutional layer kernel is fixed at a 3x3 size, and more depth is added to the network to accomplish the learning needed. For this reason, this group created several VGG architectures starting with VGG11 to VGG19 as Table 15 shows below.

[7] K. Simonya, A Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition" ICLR, 2015.

Table 15 - VGG Convolutional Network Configuration

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

The VGG16 architecture is composed of 41 layers "*L"* and 40 "*L-1"* connection this is because the VGG16 network is a sequential network. The input layer is fixed size of 224 x 224 x 3 as Table 14 showed previously. The image is passed through a stack of the convolutional layers separated pooling layers as Figure 19 shows. More precisely, the input layer is followed by two sets of double convolutional layers then a pooling layer, then three sets of 3 convolutional layers then a pooling layer, then followed by 3 single fully connected layers and subsequently a dropout layer. The first fully connected layers were of size 4096 and the later one was 1000.

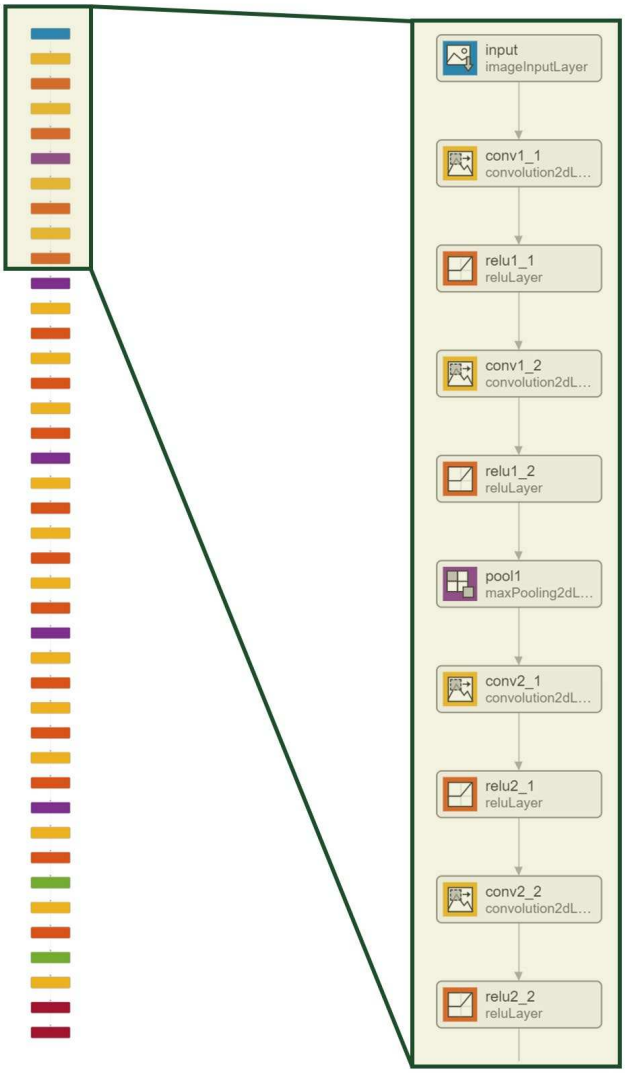


Figure 19 - VGG16 Architecture

In our application, the fully connected layers were the only ones that were not frozen. At first the same architecture discussed above was used, then the fully connected layers size and quantity were increased and decreased to test the performance. The next section discusses the best results achieved with five fully connected layers where the first one size was set to 4096; the second layer size was set to 2048; the third layer was set to 1024; the fourth layer was set to 512; and the last layer was set to 196 to match our class size.

VGG16 Results

Despite the fact that VGG16 achieved remarkable results on the ImageNet Dataset, as well as it had significant performance on Benavides and Tae paper[8], this architecture performed very poorly on the cars dataset (as Figure 20 shows below) and I was not able to replicate the results in the paper referenced above. The highest accuracy achieved on the validation was 1.044% and the log loss never fell below 5.2.

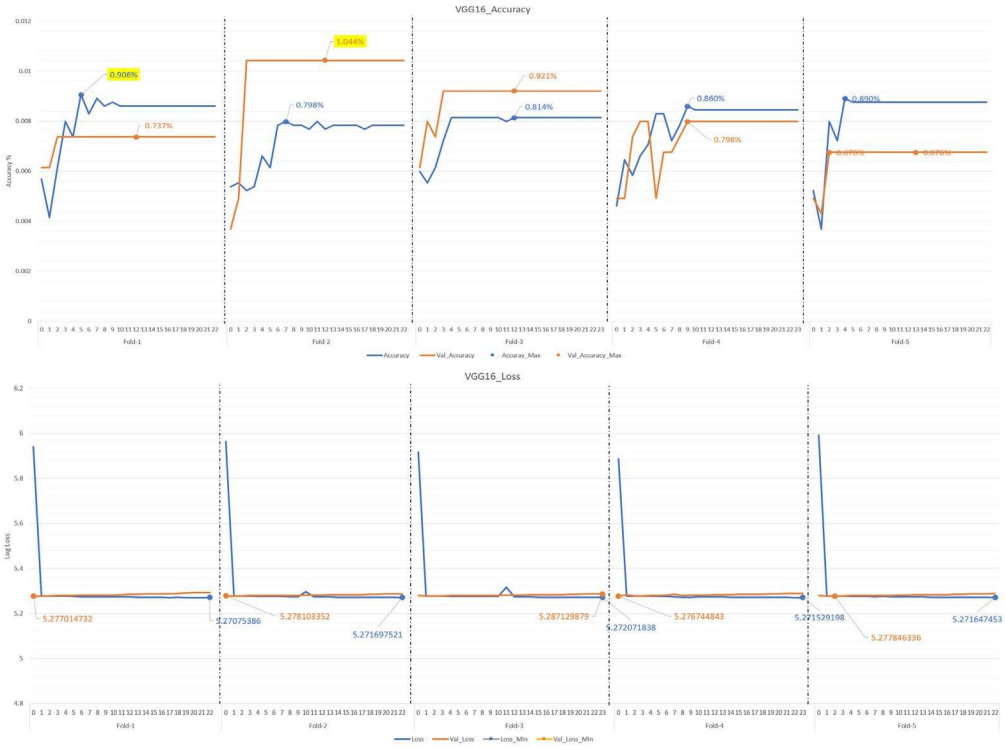


Figure 20 - (a) VGG16 Accuracy, (b) VGG16 Loss

The results were likely poor because all the car features were very similar. The results improved only when the class size was reduced from 196 classes to 3 classes, the largest number of classes for which accuracy exceeded 80%. For this reason, this architecture cannot be used for the application proposed in this paper.

[8] N. Benavides, C. Tae, "Fine-Grained Image Classification for Vehicle Makes and Models using Convolutional Neural Networks" CS230 Stanford

VGG19 Convolutional Neural Network

VGG19 Architecture

The VGG19 is another convolutional neural network model that was also proposed by the Visual Geometry Group (VGG) at Oxford University. The main difference between VGG16 and VGG19 is the number of the convolutional layers as Figure 21 shows.

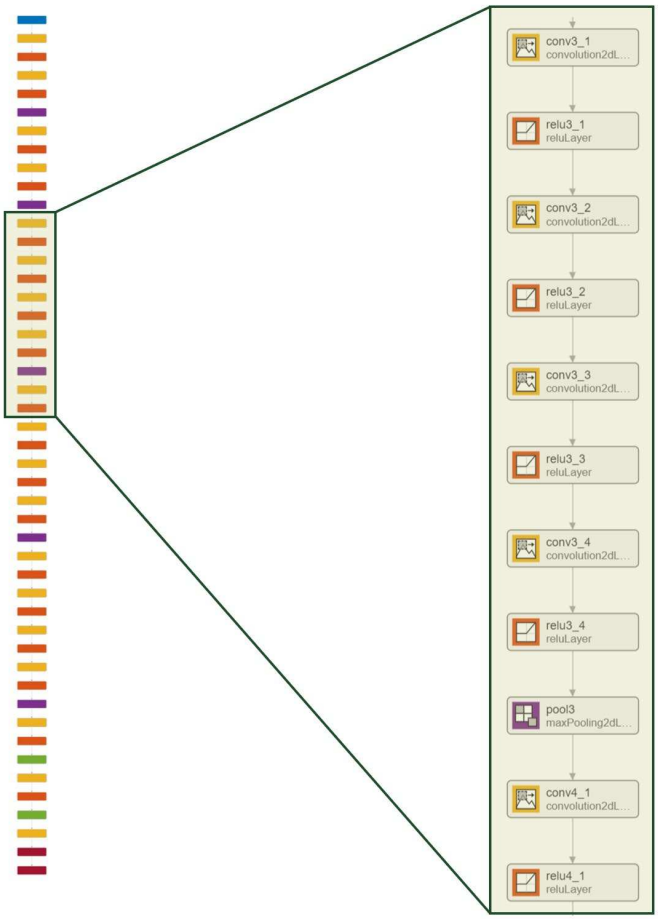


Figure 21 - VGG19 Architecture

The VGG19 architecture comprises 47 layers "*L"* and 48 "*L-1"* connections. This is because the VGG19 network is a sequential network, just like the VGG16 described in the prior section. The input layer is fixed size of 224 x 224 x 3 as Table 14 showed previously. The image is passed through a stack of the convolutional layers separated pooling layers as Figure 21

shows. More precisely, the input layer is followed by two sets of 2 convolutional layers then a pooling layer, then three sets of 4 convolutional layers then a pooling layer, then followed by 3 single fully connected layers followed by a dropout layer. The first fully connected layers were of size 4096 and the later one was 1000.

In our application, the fully connected layers were the only ones that were not frozen. At first the same architecture discussed above was used, then the fully connected layers size and quantity were increased and decreased to test the performance (a form of sensitivity analysis), similar to the VGG16 experiment discussed above. The next section discusses the best results achieved with five fully connected layers where the first one size was set to 4096, then the second layer size was set to 2048, the third layer to 1024, the fourth layer to 512, and the last layer to 196 to match our class size.

VGG19 Results

Again, even though VGG19 achieved remarkable results on the ImageNet Dataset, and most of the high-level and low-level features were transferred to our application using the transfer learning techniques, this architecture performed very poorly on the cars dataset (as Figure 22 shows below). The highest accuracy achieved on the validation was 1.105% and the log loss never fell below 5.2 as well.

The results were poor, in all likelihood, because all the car features were very similar. The results improved only when the class size was reduced from 196 classes to 5 classes, the largest number of classes for which accuracy exceeded 80%. For this reason, this architecture cannot be used for the application proposed in this paper.

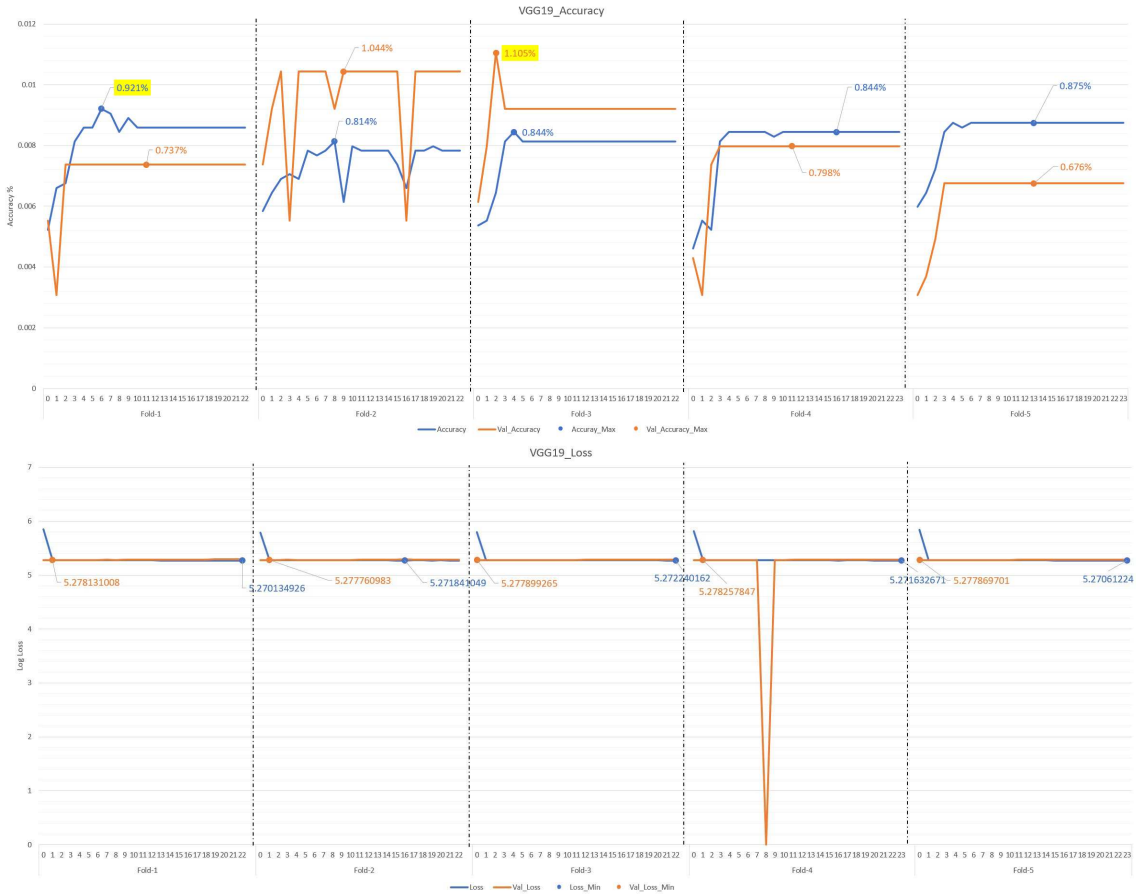


Figure 22 - (a) VGG19 Accuracy, (b)VGG19 Loss

DenseNet201 Convolutional Neural Network

DenseNet201 Architecture

The DenseNet201 is a convolutional neural network model that was proposed by Huang, Liu and other authors in 2017 in the paper "Densely Connected Convolutional Networks"[9] It is a substantially denser network than all predecessor architectures discussed in the previous sections. It achieved remarkable results in the CIFAR-10, CIFAR-100, SVHN and ImageNet datasets. The DenseNet201 Architecture is shown in Figure 23.

[9] G.Huang, Z.Liu, et. al, "Densely Connected Convolutional Networks" Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR) 2017 pp. 4700-4708.

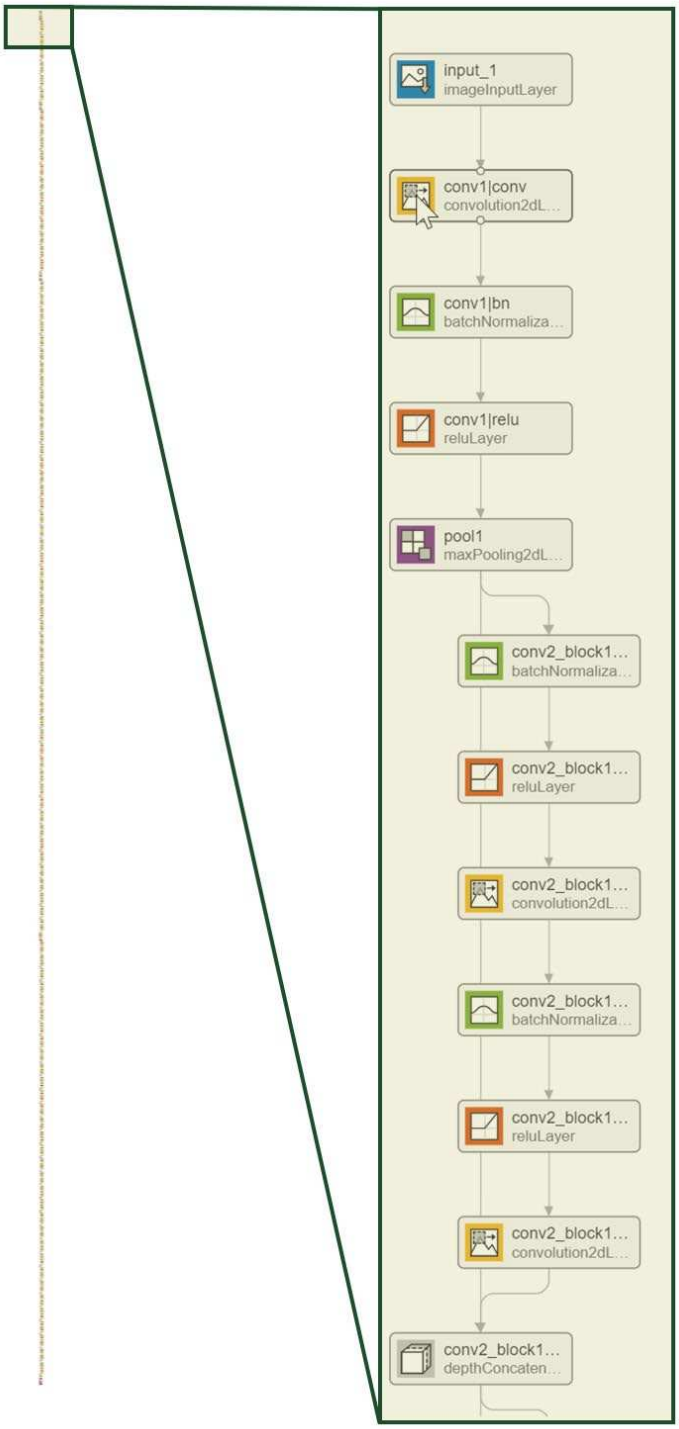


Figure 23 - DenseNet201 Architecture

The DenseNet201 architecture is composed of 708 layers "*L"* and 805 connections. The main difference from the previously discussed architectures is that it is composed of several

dense convolutional blocks with "*l*" layers where these layers are connected to every other layer on that convolutional block with "*l(l+1)/2*" connections. This allows the feature maps of each layer to be passed to the subsequent layers as inputs. It is often a concern that DenseNets are costly and grow out of proportions; however, this is resolved by inserting a transitional block between every dense layer. The transitional block is composed of a 1x1 convolutional layer followed by a pooling layer. The input layer is fixed size of 224 x 224 x 3 as Table 14 showed. The input layer is followed by five dense blocks, with each one increasing in size: it starts with 1, followed by 6, followed by 12, followed by 48, and lastly followed by 6 convolutional blocks in each of the dense blocks, respectfully. The paper discusses several advantages to the DenseNet201 architecture: The network alleviates the vanishing-gradient, strengthens feature propagation, promotes feature reuse, and substantially reduces the number of parameters needed to perform the learning.

DenseNet201 Results

DenseNet201 has shown a significant improvement from the previously mentioned architectures. As Figure 24 shows, it consistently achieved 90% or above accuracy, with the highest being 92.33% on the validation accuracy. The loss dropped to 0.37 without any data augmentation.

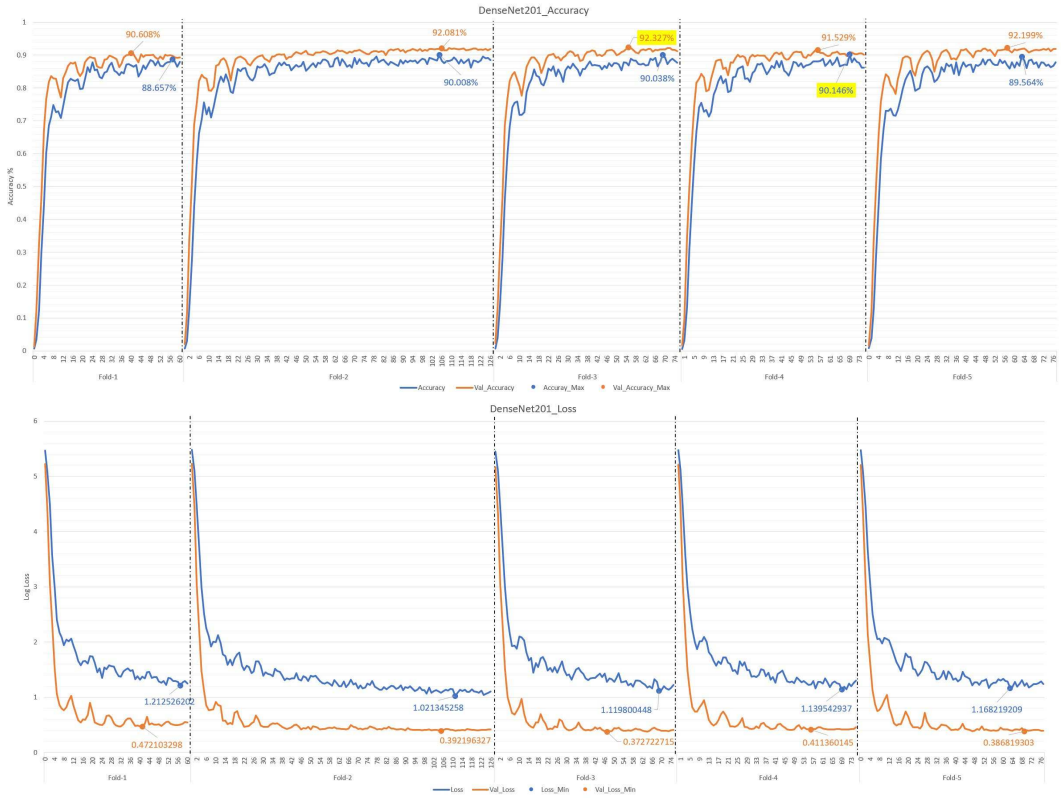


Figure 24 - (a) DenseNet201 Accuracy, (b) DenseNet201 Loss

NASNetLarge Convolutional Neural Network

NASNetLarge Architecture

The NASNetLarge is a convolutional neural network model that was proposed by Zoph, Vasudevan and other authors in 2018 in the paper "Learning Transferable Architectures for Scalable Image Recognition"[10] It is a substantially denser network than all predecessor architectures discussed above. The NASNetLarge is based on a Neural Architecture Search (NAS) framework which means it uses reinforcement learning to optimize the network configuration autonomously. The NASNetLarge is composed of 1243 layers with 1462 transitions. These layers are composed of Normal Cells and Reduction Cells as the architecture is

[10] B. Zoph, V. Vasudevan et. al, "Learning Transferable Architectures for Scalable Image Recognition" Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR) 2018 pp. 8697-8710.

modeled on Figure 25. The Normal Cells return a feature map with the same dimensions as the input and the Reduction Cells returns a feature map reduced by half.

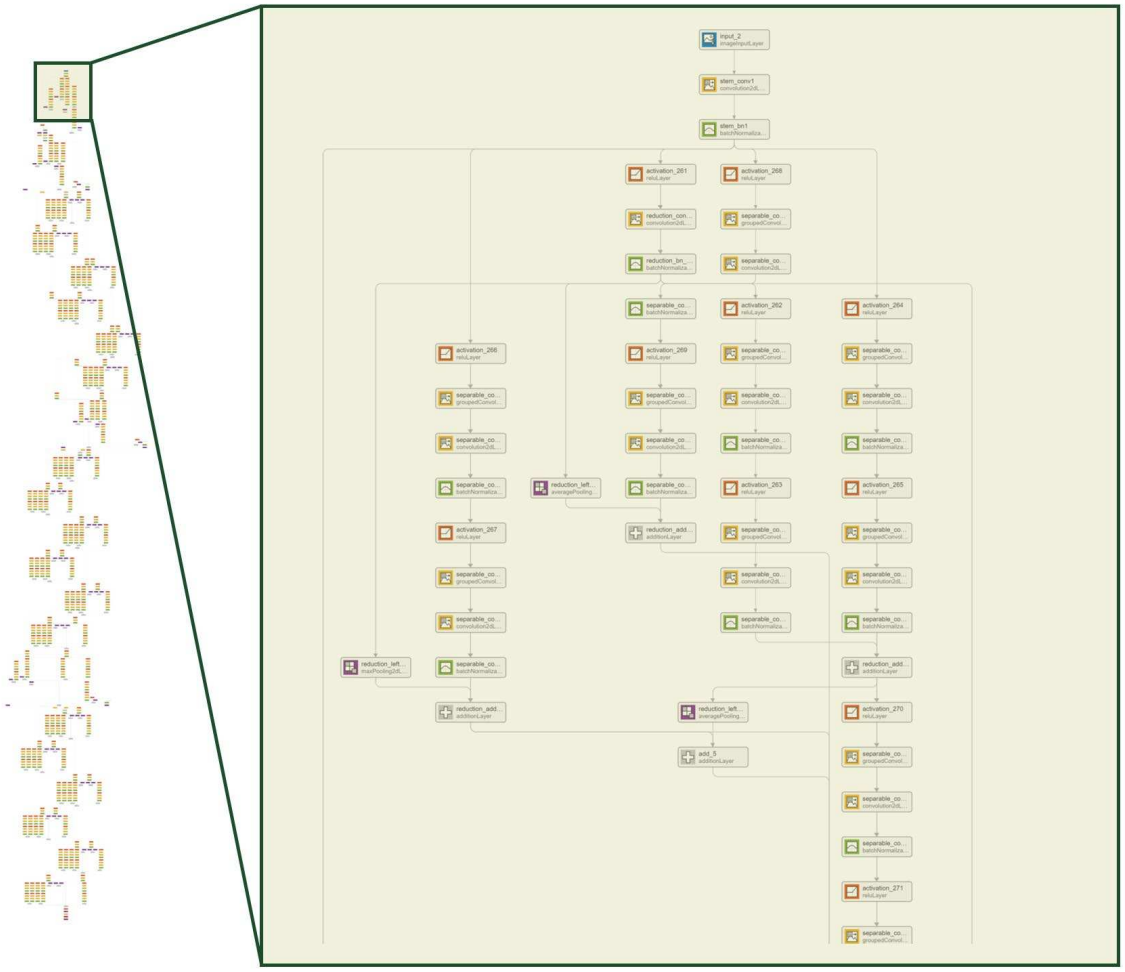


Figure 25 - NASNetLarge Architecture

Applying the NAS framework to a large dataset can be computationally expensive; thus, Zolph's paper proposes a method to search for a good architecture on a smaller dataset and then transfer the search space to a larger space with more cells with identical structure. The main premise of the NAS architecture is best described by Figure 26. A Controller Recursive Neural Network (RNN) predicts architecture "*A*" from a search space with probability "*p.*" A child

network with architecture "*A*" is trained to converge, achieving accuracy "*R*". Then the gradients of "*p*" are scaled by "*R*" to update the RNN controller.

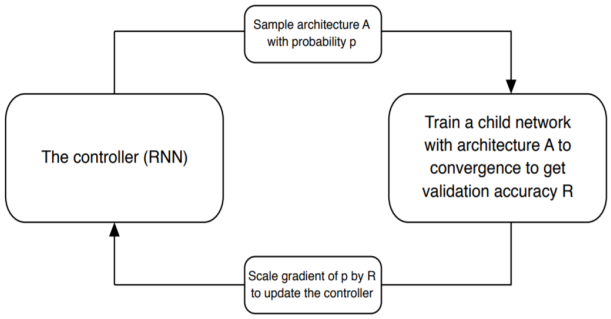


Figure 26 - Overview of Neural Architecture Search

The scalable architecture concept is shown in Figure 27, where the diagram highlights on the left the model architecture using a smaller dataset such as CIFAR-10, then scaled on the right diagram to match the needs of a larger dataset such as ImageNet. The choice for the number of Normal Cells stacked in between the reduction cells can be experimentally/empirically changed.

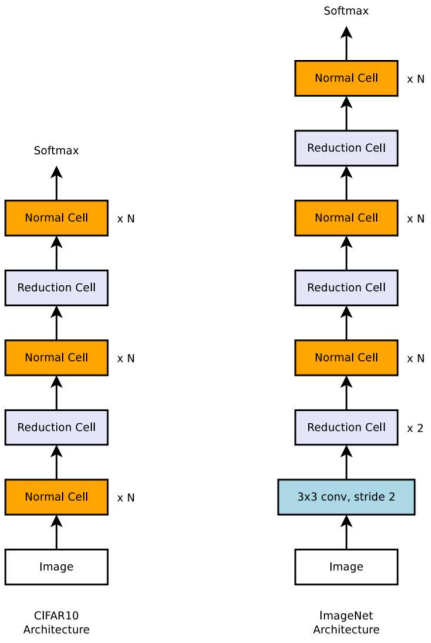


Figure 27 - Scalable Base Blocks

Each of the cells are constructed recursively in block stages. Each block consists of the controller selecting a pair of hidden layers and randomly chooses an operation described by Figure 28(b); it then randomly chooses a combination operation. The combination operations consist of either adding or concatenating the result. The resultant hidden layer is passed to the following block to be added as a choice as shown on Figure 28(a).

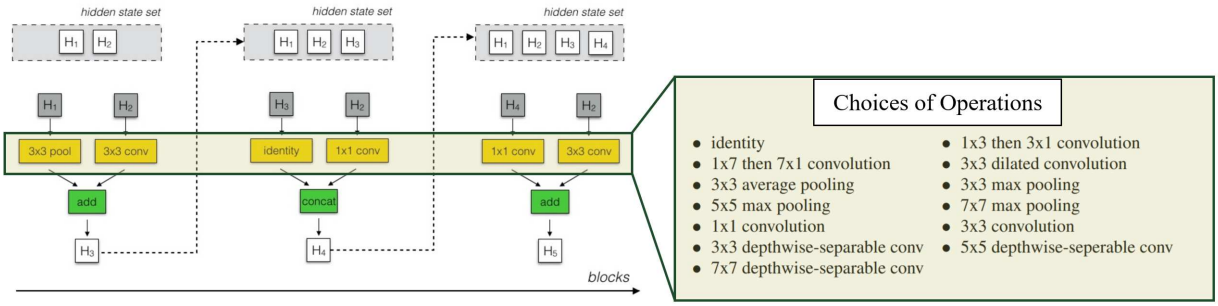


Figure 28 - (a) NASNet Search Space, (b) Convolutional Cell Choices of Operation

Lastly, the previously mentioned procedure is repeated until an optimum cell structure is constructed for each of the two cell types. In our application, we utilized the same structure as ImageNet which is much larger than the Stanford car dataset and adapted through transfer learning for our application. For this reason, it achieved an effective result which will be discussed in the next section.

NASNetLarge Results

NASNetLarge has taken much longer to train; however, it showed a slight improvement from the previously mentioned architectures. Figure 29 shows it consistently achieved accuracies above 92%, with the highest being 92.41% on the validation accuracy. The loss dropped to 0.36 without any data augmentation.

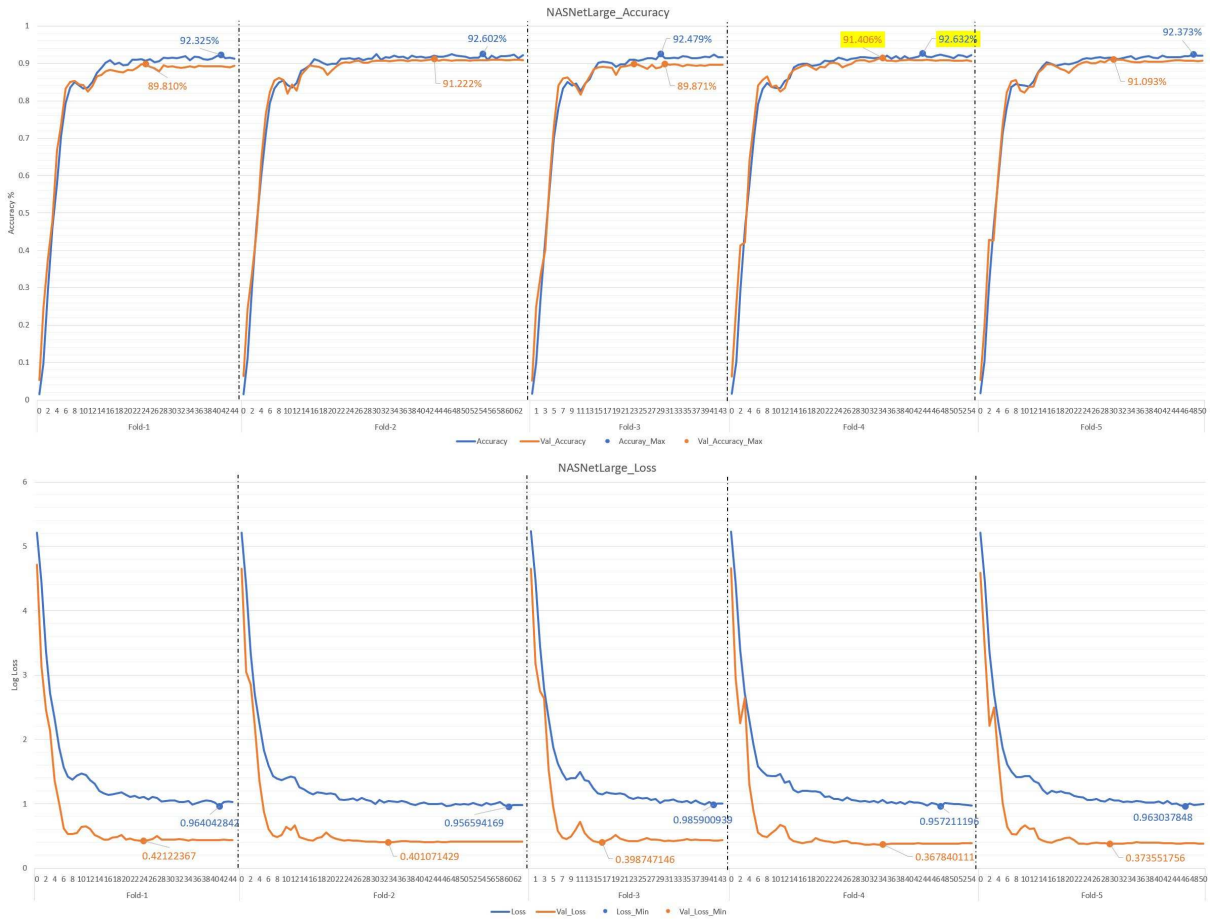


Figure 29 - (a) NASNetLarge Accuracy, (b) NASNetLarge Loss

InceptionV3 Convolutional Neural Network

InceptionV3 Architecture

The InceptionV3 is a convolutional neural network model that was proposed by Szegedy, Vanhouche and other authors in 2016 in the paper "Rethinking the Inception Architecture for Computer Vision".[11] InceptionV3 is based on the Inception framework and GoogLeNet; The network that is trained on ImageNet is composed of 315 layers and 349 connections (as Figure 30 shows below).

[11] C. Szegedy, V. Vanhouche et. al, "Rethinking the Inception Architecture for Computer Vision" Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR) 2016 pp. 2818-2826.

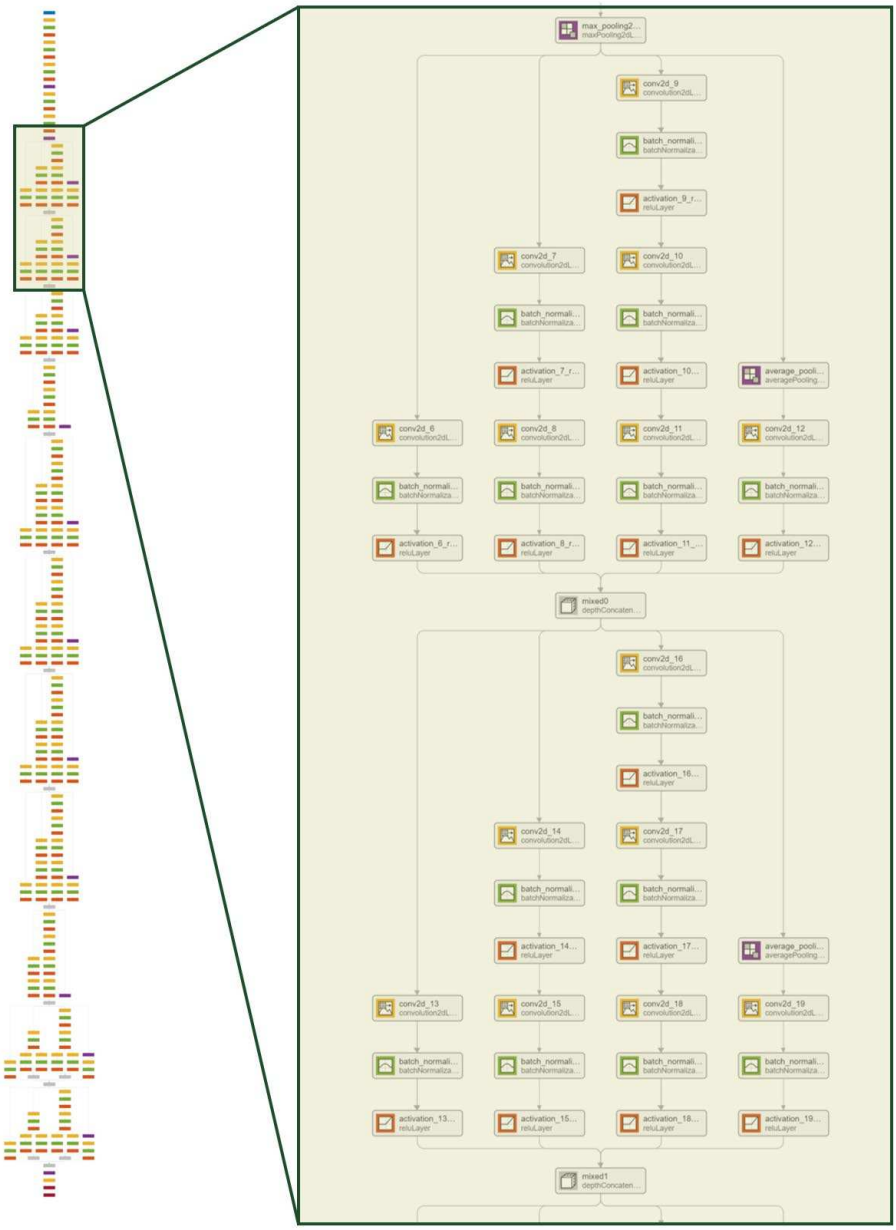


Figure 30 - InceptionV3 Architecture

The main principle of the Inception architecture is based on the Hebbian Principle which states: neurons that fire together, wire together. For this reason, to cover the various clusters in a given image, it was thought to concatenate a 1x1 convolutional layer with 3x3 convolutional layer and 5x5 convolutional layer to form the first Naïve Inception module (as shown Figure 31(a)). This approach posed a computational problem due to the channel dimensionality

increasing drastically. Therefore, the Inception architecture changed the Naïve based inception by inserting a dimensionality reduced set of convolutional layers to the stack as shown in Figure 31(b) below.

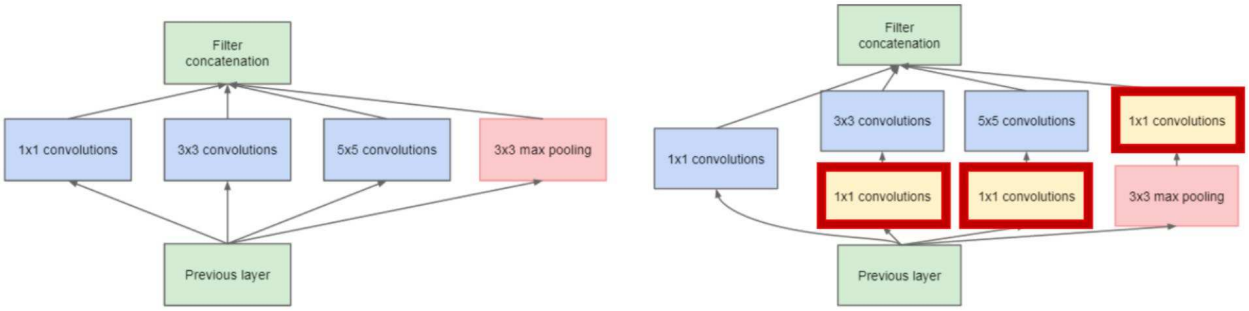


Figure 31 - (a) Naive Inception Module, (b) Inception Module

The InceptionV3 is based on the Inception module described above; however, Szegedy's paper proposed to change the 5x5 convolutional filter with a stack of 3x3 convolutional filters to reduce the computational cost. This contributes to faster learning, since the stack of 3x3 convolutional filters has the same receptive fields as the 5x5 convolutional filter but requires less computation. Similarly, any higher dimensionality convolutional filter can be replaced with a concatenation of 1xn and nx1 convolutional filters. Figure 32 shows the convolutional layer filter transformation.

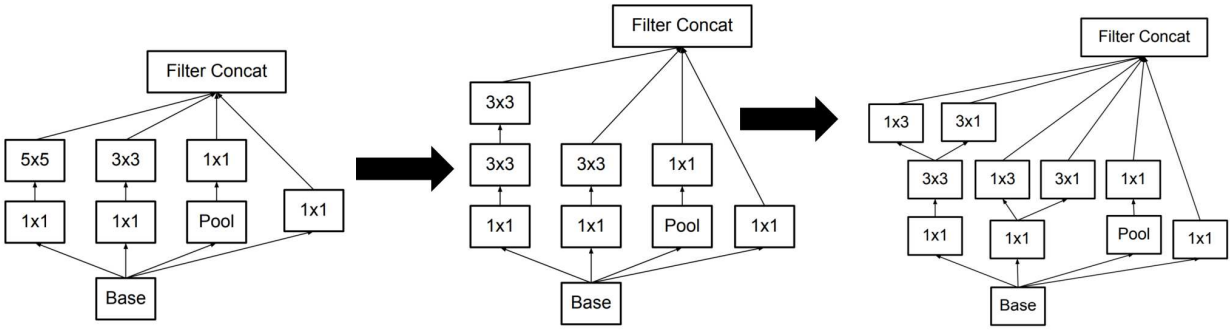


Figure 32 - (a) Inception Module, (b) InceptionV3 Module, (c) InceptionV3 Module with expanded filter bank

InceptionV3 Results

InceptionV3 has converged faster than any of the architectures mentioned above. Figure 33 shows that it consistently achieved accuracies above 90%, with the highest accuracy of 91.01% being observed for the validation accuracy. The optimum loss achieved was 0.377 without any data augmentation.

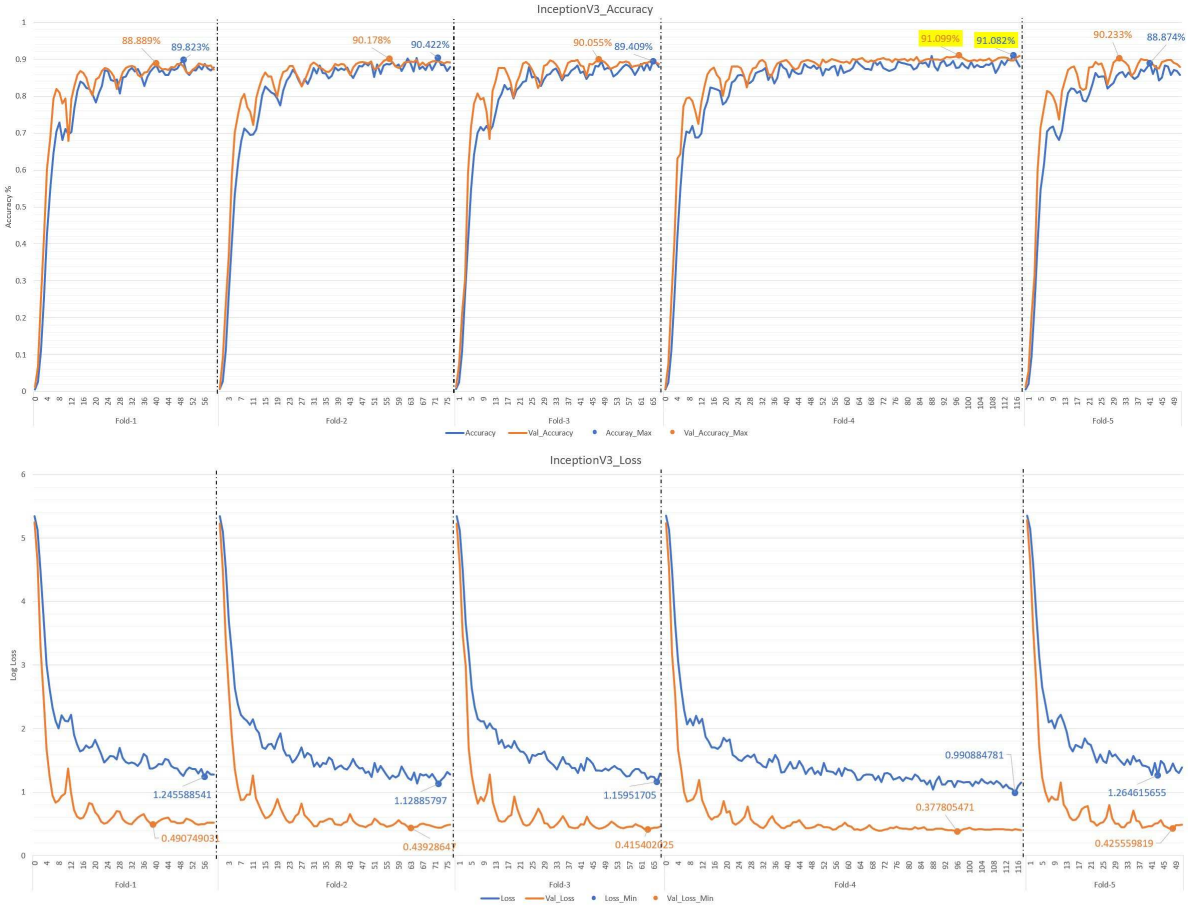


Figure 33 - (a) InceptionV3 Accuracy, (b) InceptionV3 Loss

Xception Convolutional Neural Network

Xception Architecture

The Xception is a convolutional neural network model that was proposed by Chollet, in 2016 in the paper "Deep Learning with Depthwise Separable Convolutions".[12] The Xception name comes from Extreme Inception since the architectural core block is based on the Inception block discussed in the previous section. The network that is trained on ImageNet is composed of 170 layers and 181 connections (as Figure 34 shows below).

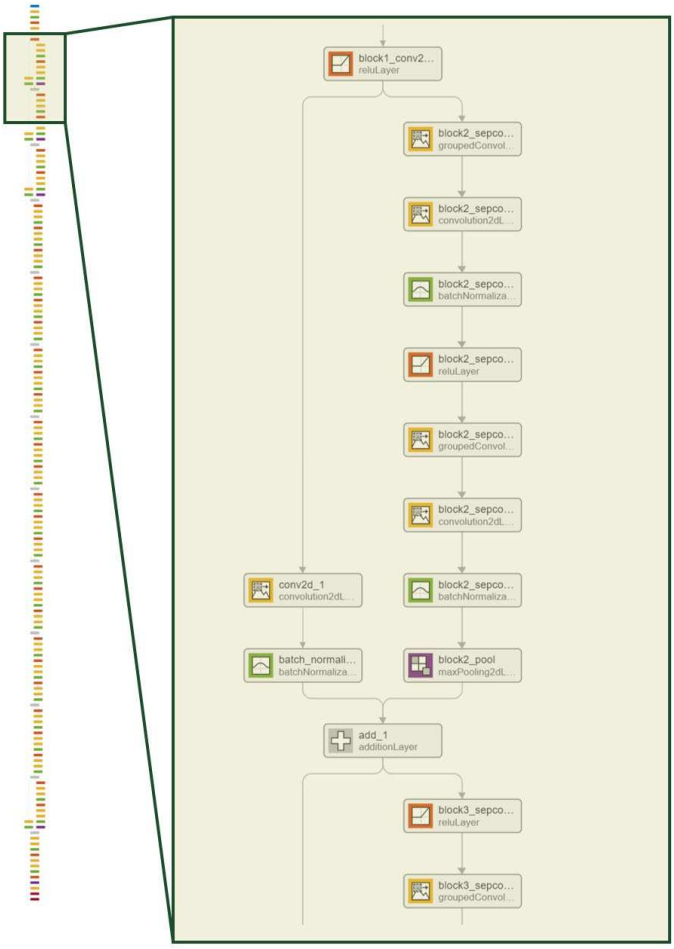


Figure 34 - Xception Architecture

[12]F. Chollet, "Deep Learning with Depthwise Separable Convolutions" Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR) 2017 pp. 1251-1258.

The main premise of Xception is to simplify the architecture of the InceptionV3 architecture; in essence, instead of performing a combination of 3x3 filters and 1x1 filters and pooling with various stacks, the paper proposes an architecture to use 3x3 filters and 1x1 (as Figure 35 shows below).

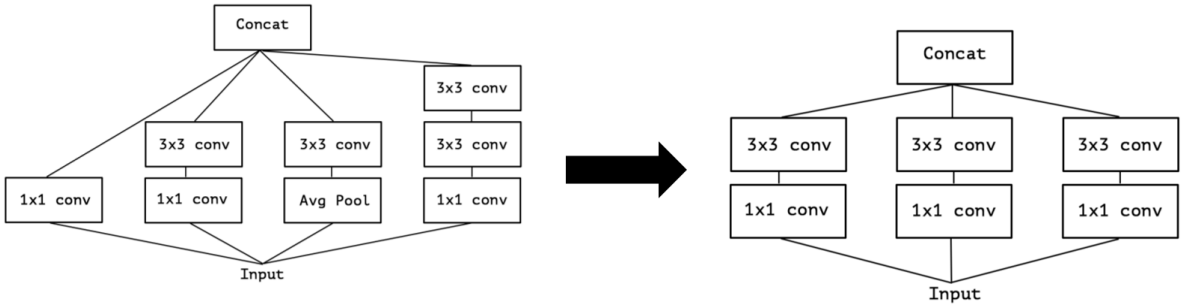


Figure 35 - (a) Inception Module, (b) Simplified Inception Module

The next concept that the paper discusses is to perform group convolution on the simplified inception; in other words, after performing a 1x1 convolution a group of channels is selected, and a 3x3 convolution is performed on this group of channels. This idea can be more generalized; rather than using a group of channels, the convolution can be performed on a single channel (as Figure 36 shows below). This concept of separating the channels and performing the 3x3 convolution on each single channel is the main feature of the Xception architecture; this is often referred to as *depthwise separable convolution*.

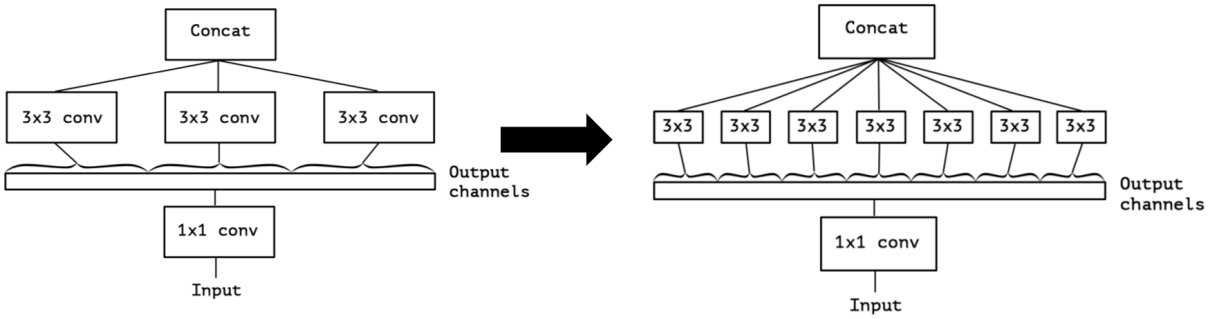


Figure 36 - (a) Reformulated Simplified Inception Module, (b) Xception Module

Figure 34 shows that the architecture is not composed of the same building block discussed above. This is because the architecture is mainly composed of three main flows as Figure 37 shows: Entry Flow, Middle Flow and Exit Flow. The Entry and Exit flows are responsible for changing the dimensionality of the data by performing the stride operation, whereas the middle flow is only composed of the separable convolution repeated 8 times.

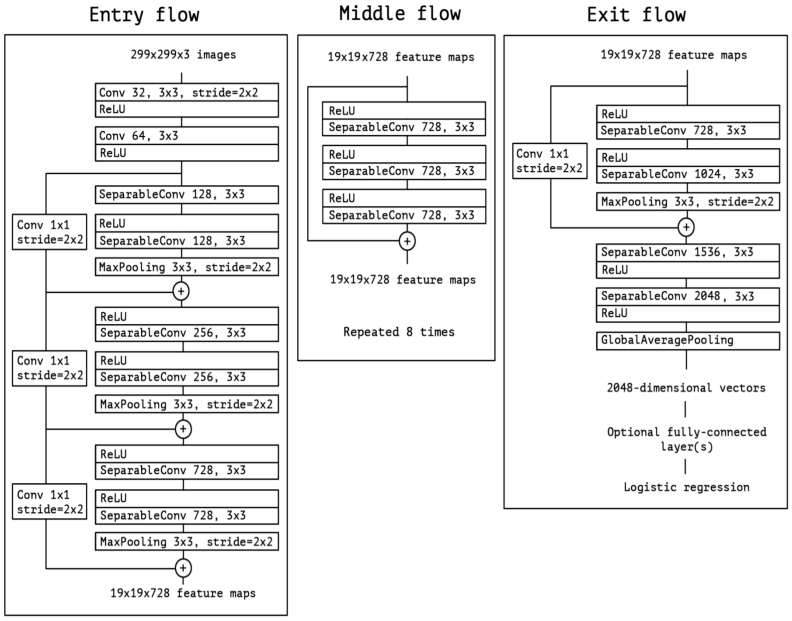


Figure 37 - Xception Architecture Flow

Xception Results

Xception has performed one of the top three architectures that this project implemented. Figure 38 shows that it consistently achieved accuracy above 90%, with the highest being 91.53% on the validation accuracy. The optimum loss achieved was 0.40 without any data augmentation.

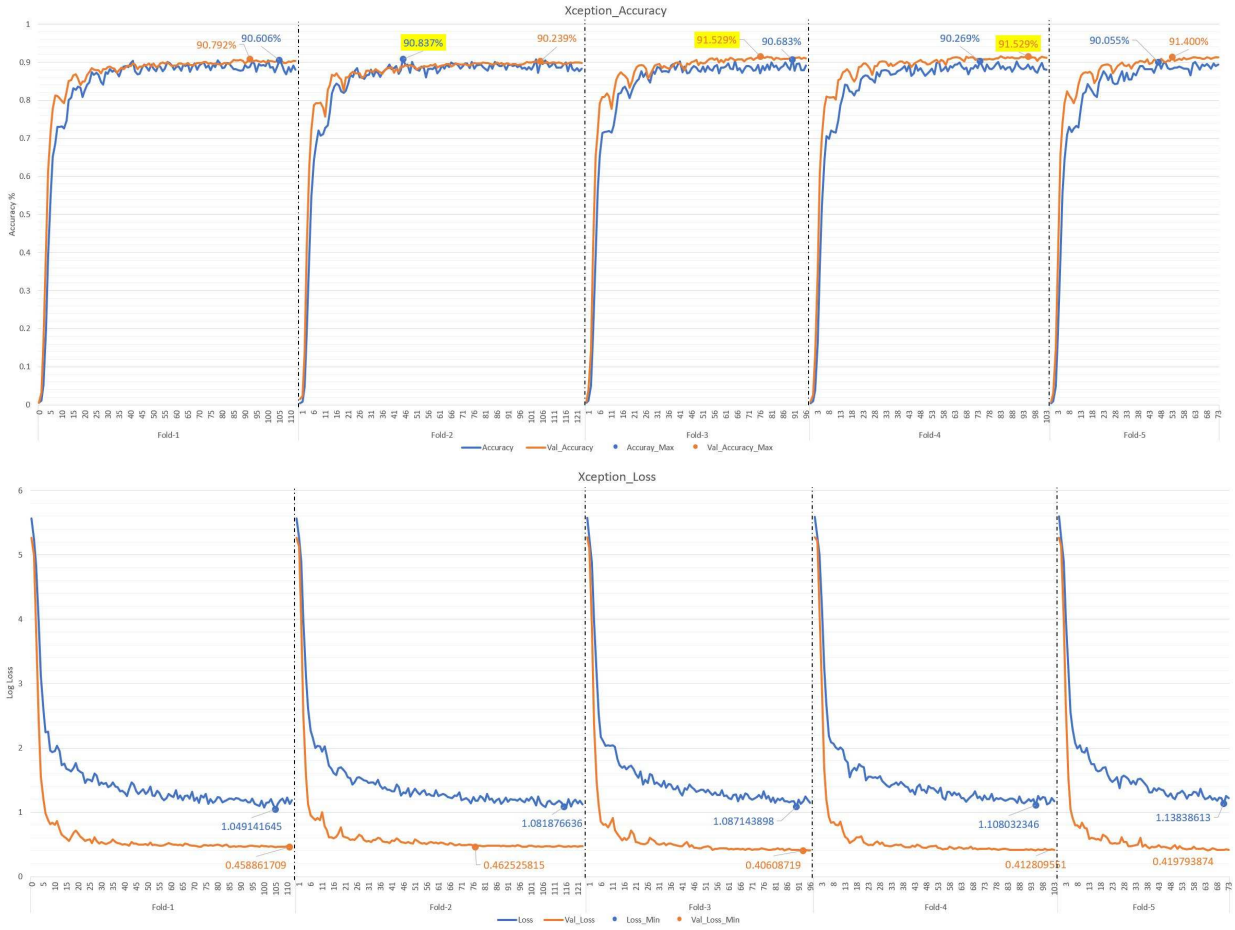


Figure 38 - (a) Xception Accuracy, (b) Xception Loss

InceptionResNetV2 Convolutional Neural Network

InceptionResNetV2 Architecture

The InceptionRestNet architecture is a convolutional neural network model that was proposed by Szegedy, Iofee, and other authors in 2017 in the paper "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning".[13] The InceptionResNet is based on the Inception model in conjunction with the Residual Network (ResNet) architecture. The

[13]C. Szegedy, S. Ioffe, et al., "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning" Thirty-First AAAI Conference on Artificial Intelligence.

network that is trained on ImageNet is composed of 824 layers and 921 connections (as Figure 39 shows below).

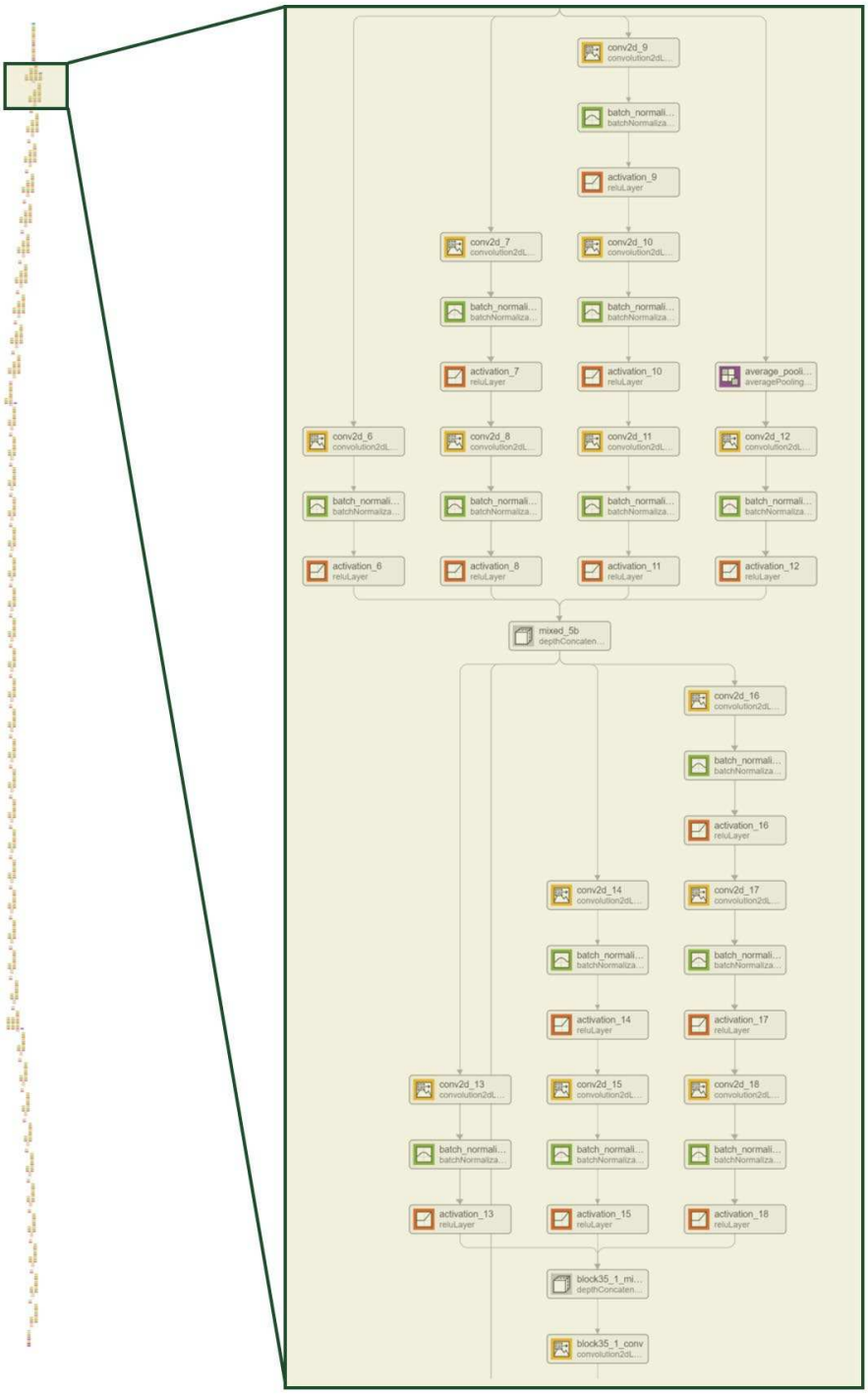


Figure 39 - InceptionResNetV2 Architecture

The macro structure of this algorithm is composed of the input layer, which takes an input of 299x299x3 (as described in Table 14) followed by a stem layer; then a series of inception cells and reduction cells; then, lastly, a layer of average pooling and a softmax. Figure 40 shows the macro structure of the InceptionResNetV2 architecture and the details of every block discussed above.

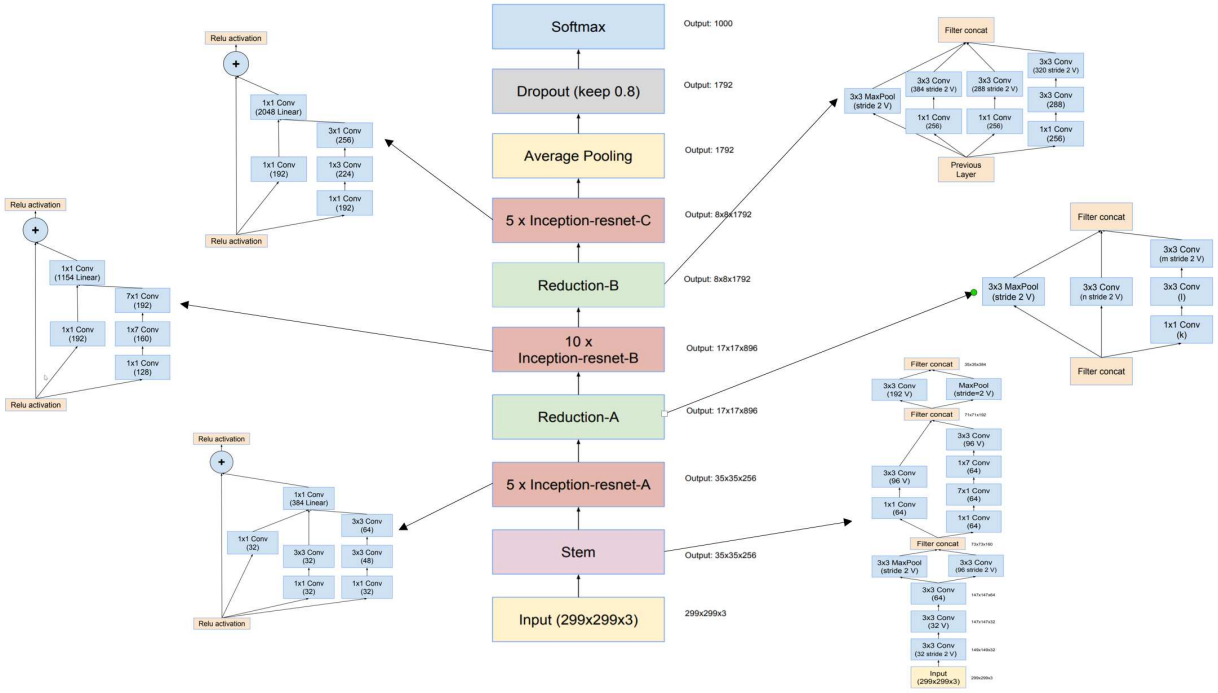


Figure 40 - InceptionResNetV2 Structure

InceptionResNetV2 Results

InceptionResNetV2 has consistently achieved accuracies above 85%, with the highest being 89.44% on the validation accuracy. The optimum loss achieved was 0.41 (as Figure 41 shows). Despite the fact that it didn't perform as well as the previously mentioned architectures, it performed overall rather well without any data augmentation.

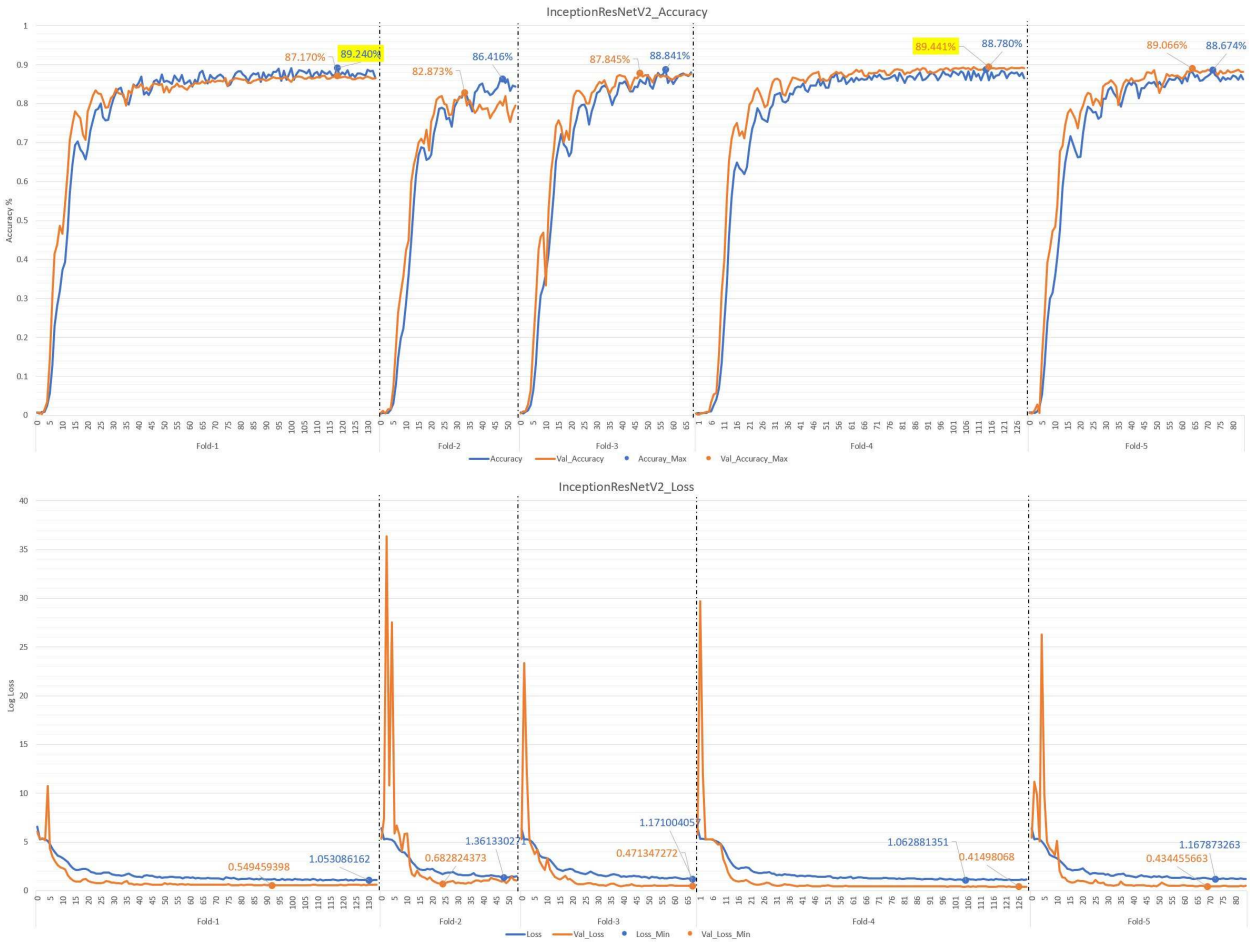


Figure 41 - (a) InceptionResNetV2 Accuracy, (b) InceptionResNetV2 Loss

MobileNetV2 Convolutional Neural Network

MobileNetV2 Architecture

The MobileNetV2 architecture is a convolutional neural network model that was proposed by Sandler, Howard, and other authors in 2018 in the paper "MobileNetV2: Inverted Residuals and Linear Bottlenecks".[14] MobileNet architectures target resource constrained environments; for this reason, the size of these networks is relatively small in comparison with

[14]M.Sandler, A.Howard, et al., "MobileNetV2: Inverted Residuals and Linear Bottlenecks" Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR) 2018 pp. 4510-4520.

the previously discussed architectures. The network that is trained on ImageNet is composed of 154 layers and 163 connections as Figure 42 shows below.

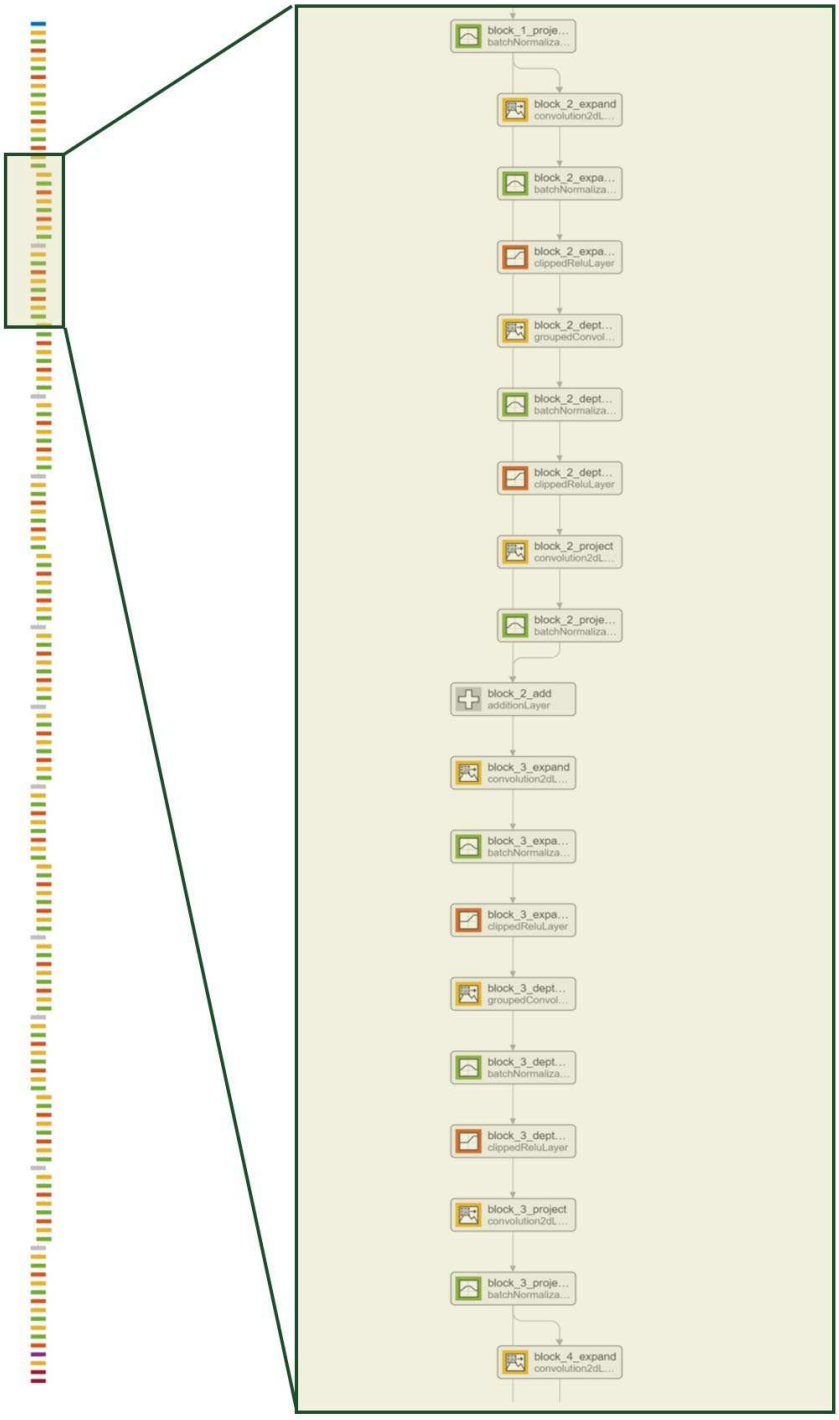


Figure 42 - MobilenetV2 Architecture

Before discussing the MobileNetV2 architecture, it is worth mentioning the evolution of the separable convolutional blocks. Figure 43(a) shows the regular convolution, whereas Figure 43(b) shows the separable convolutional block. The light color block indicates that it is on the next layer. Figure 43(c) shows the concept of the separable block with a linear bottleneck, whereas Figure 43(d) shows the separable block with an expanded layer bottleneck. The smaller hashed blocks indicate layers that do not contain non-linearities. The ReLu6 is a ReLu activation function that is fully saturated at value 6 rather than continuously increasing as the regular ReLu activation function does.

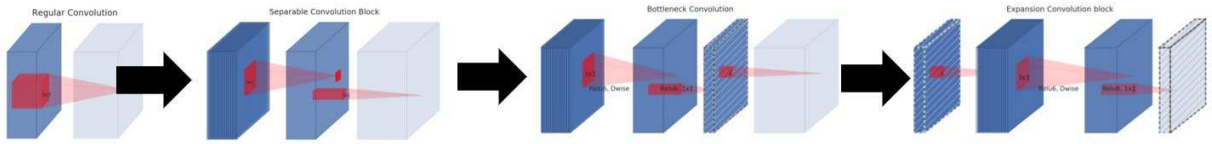


Figure 43 - (a)-(d) Evolution of Separable Convolution Blocks

MobileNetV2 architecture is based on the MobileNet framework with an added layer that implements an inverted residual with linear bottleneck. This module takes a low-dimensionality compressed representation input, then expands it to a higher dimensionality before filtering with a lightweight depthwise convolution. Then transforms it to a low dimensionality output with a linear convolution as Figure 44 shows.

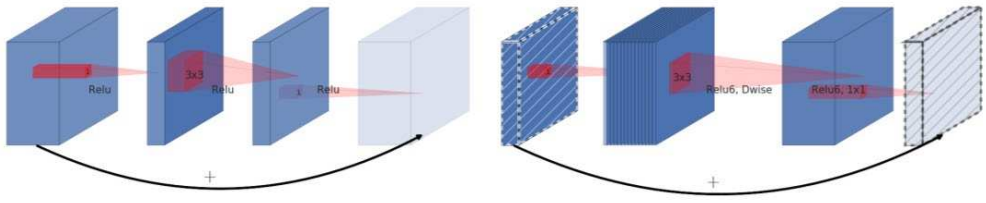


Figure 44 - (a) Residual Connection Block, (b) Inverted Residual Connection Block

MobileNetV2 Results

MobileNetV2 has consistently achieved accuracies above 85%, with the highest being 89.25% on the validation accuracy. The optimum loss achieved was 0.6 (as Figure 45 shows). Despite the fact that it didn't perform as well as the previously mentioned architectures, it still provided high accuracy without any data augmentation while being the smallest and fastest network than any of the architectures implemented on this project.

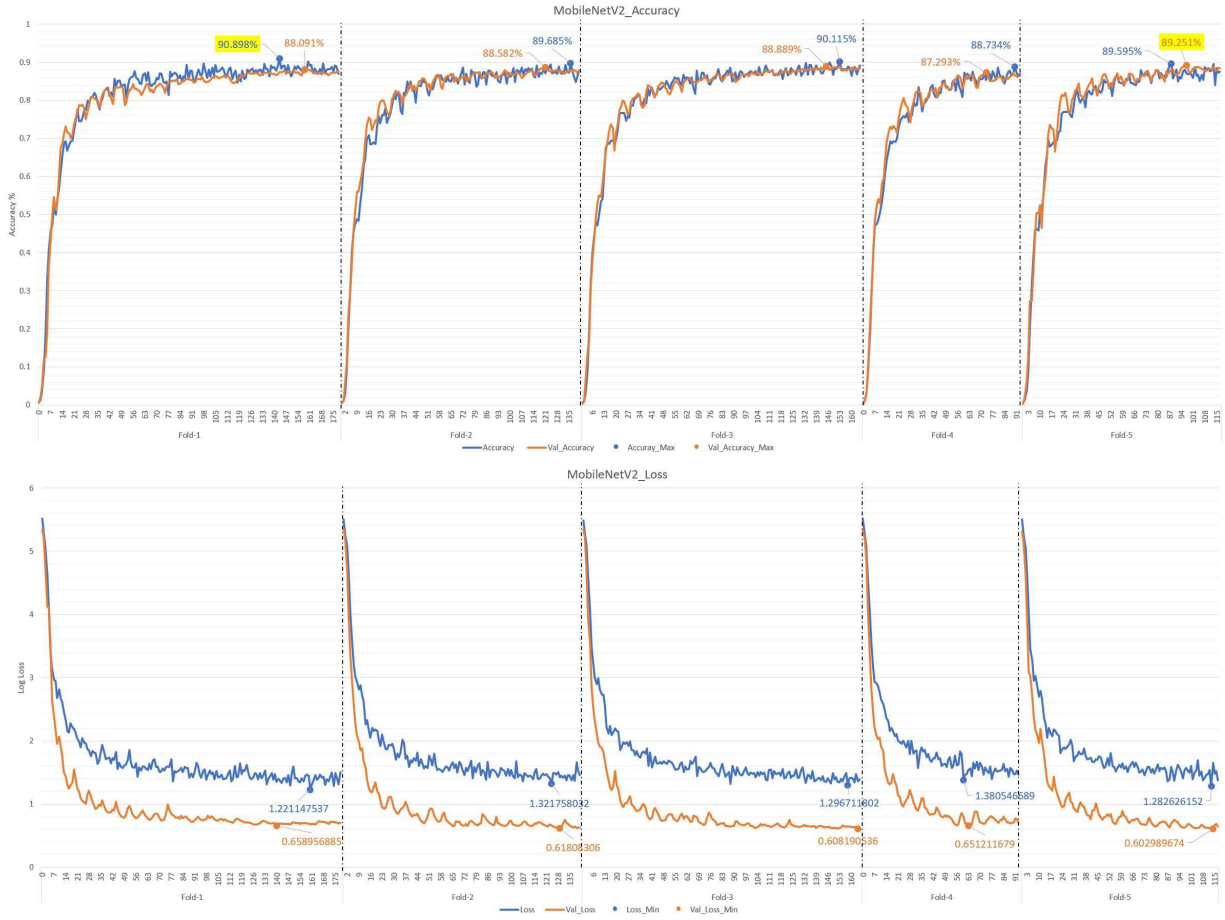


Figure 45 - (a) MobileNetV2 Accuracy, (b) MobileNetV2 Loss

ResNet50V2 Convolutional Neural Network

ResNet50V2 Architecture

The ResNet architecture is a convolutional neural network model that was proposed by He, Zhang, and other authors in 2016 in the paper "Deep Residual Learning for Image Recognition".[15] The ResNets paper is one of the most cited papers in the image recognition field; it has been cited over 105,000 times to date, largely because it introduced the concept of residual connections which resolved an anomaly with deep networks. ResNet50V2 is based on the ResNet architecture with 50 parametrized layers of depth. The ResNet50V2 network that is trained on ImageNet is composed of 177 layers and 192 connections (as Figure 46 shows below).

To appreciate the premise of ResNet and the solution that the paper by He et al. proposed, it is best to visualize the problem that was posed by deep networks as it is a crucial building block to solve more complex learning problems. It is widely believed that the deeper the network gets, the better accuracy it achieves. This is intuitive: the more capacity a network has, the more receptive fields it consumes for better learning. Counterintuitively, the exact opposite was discovered here. When a given 56-layer deep plain network was trained on the CIFAR-10 dataset, it consistently performed more poorly than a 20-layer deep plain network (as Figure 47 shows below). Similar results were observed when a deep network was trained on ImageNet dataset.

[15]K. He, X. Zhang, et al., "Deep Residual Learning for Image Recognition" Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR) 2016 pp. 770-778.

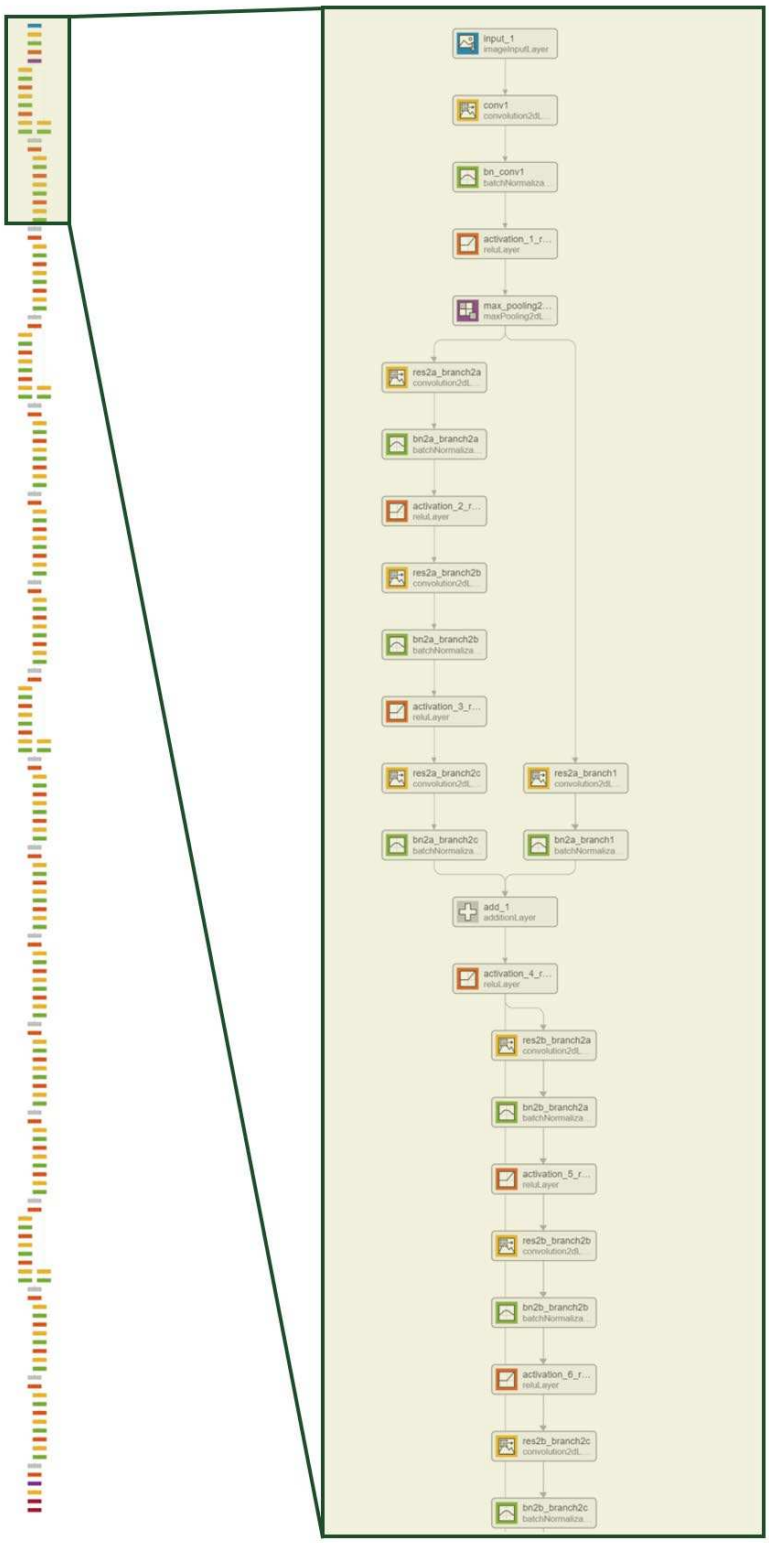


Figure 46 - ResNet50 Architecture

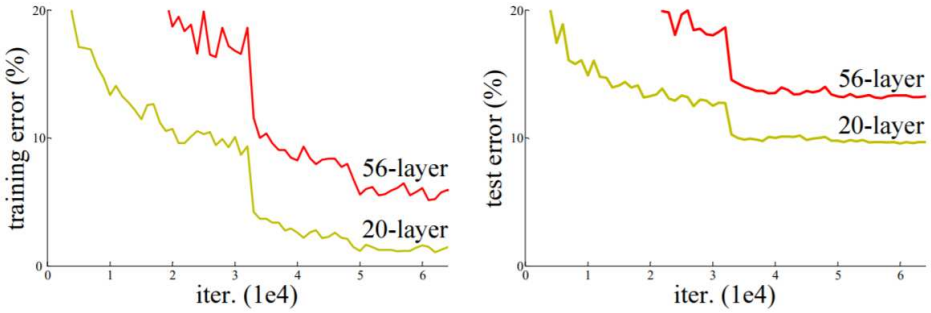


Figure 47 - (a) Deep Plain Network Training Error, (b) Deep Plain Network Test Error

The degradation is not caused by overfitting; instead, it is mainly due to the difficulty of readily optimizing all of the systems in the network. For this reason, the paper introduced the concept of residual connections as shown in Figure 48 below. In essence, it is a self-pruning or self-organizing network. If a convolutional layer is not needed, the network will choose the identity path to overpass the "unnecessary" layers.

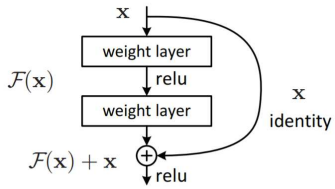


Figure 48 - ResNet Building Block

The output function $y = F(x) + x$ can be written with the following equations. The first equation can be used when the convolutional layer mapping has the same number of parameters, depth, width and computations; however, the second equation can be used if the mapping doesn't have the same dimension (as Figure 49 shows below. All the dashed residual connections are not the same dimensions; thus, the second equation will be used.

$$y = F(x, \{W_i\}) + x$$

$$y = F(x, \{W_i\}) + W_s x$$

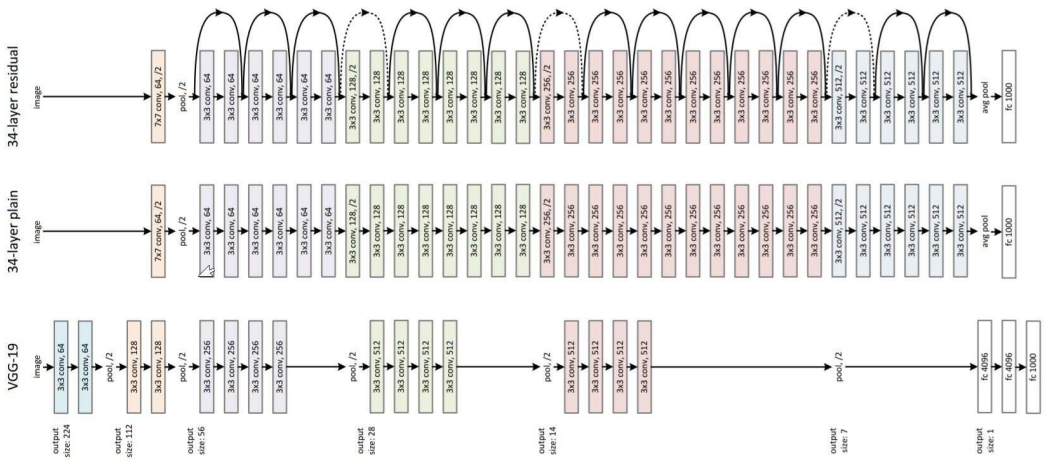


Figure 49 - Network Architectures Comparison Between Plain Networks and ResNet

Another important concept was introduced in He's paper for ResNet50, ResNet101 and ResNet152: the concept of bottleneck. Figure 50 shows the difference between the ResNet building block mentioned above and the bottleneck. As the size of the convolutional layer becomes larger, it becomes costlier. Therefore, changing the dimensions to a smaller size by performing a 1x1 filter, then performing the 3x3 followed by a reprojection to the larger space, performs much faster.

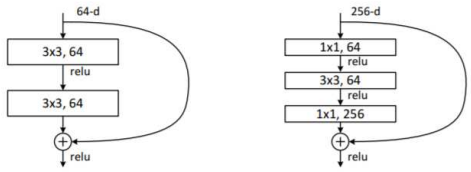


Figure 50 - (a) ResNet Building Block, (b) Bottleneck Building Block

ResNet50V2 Results

ResNet50V2 has consistently achieved accuracies above 88%, with the highest accuracy being 91.19% on the validation accuracy. The optimum loss achieved was 0.50 (as Figure 51 shows). It performed well without any data augmentation and outperformed several of the networks discussed above by implementing the residual connection concept.

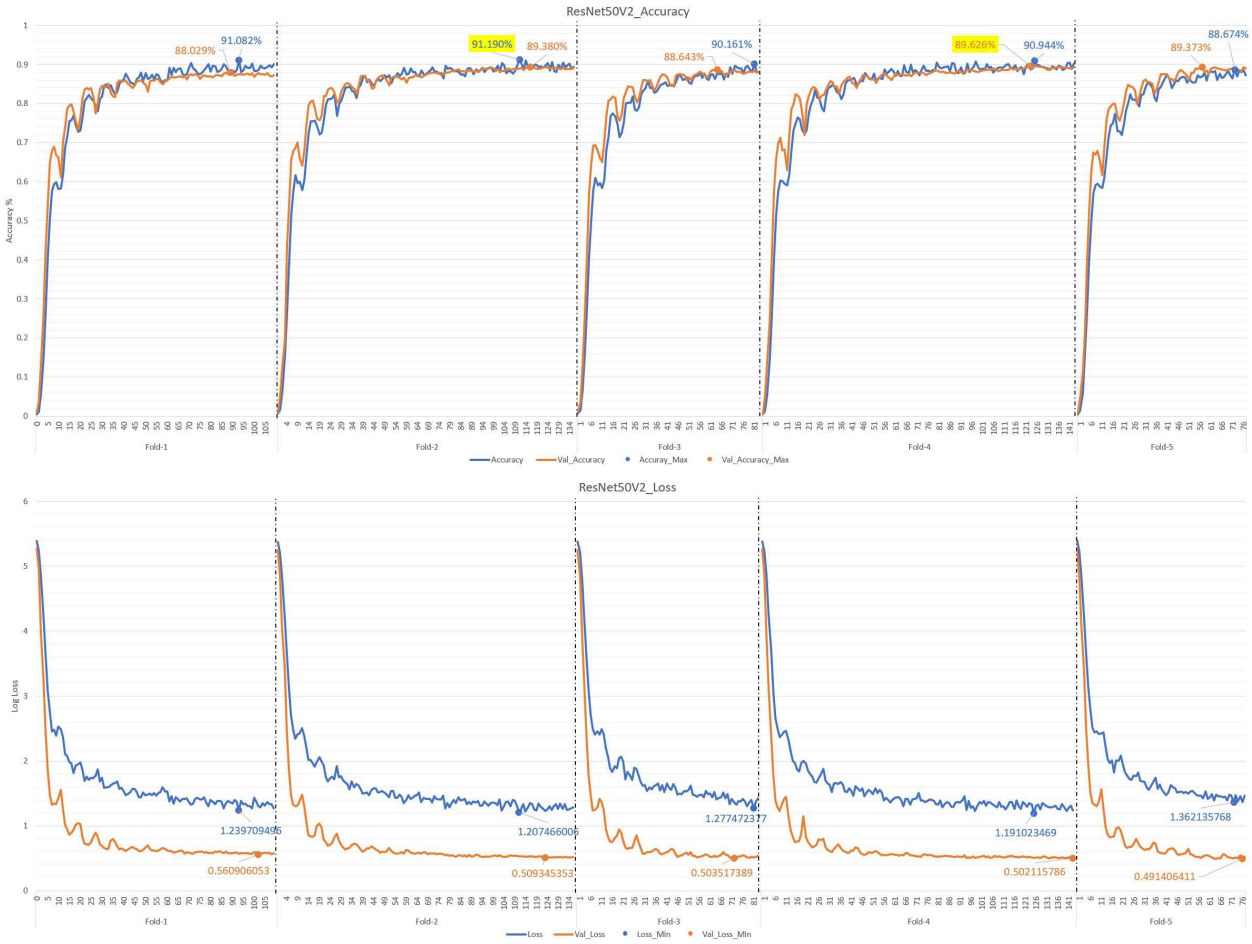


Figure 51 - (a) ResNet50V2 Accuracy, (b) ResNet50V2 Loss

ResNet152V2 Convolutional Neural Network

ResNet152V2 Architecture

ResNet152V2 is a convolutional neural network that was introduced by the same paper discussed in the ResNet50V2 section above. It is using the same concept discussed above; however, it is composed of 152 parameterized layers (where the 152 in the name comes from). Table 16 shows the architectural differences between the various ResNet Architectures.

Table 16 - ResNet Various Layer Architectures

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

The paper showed the performance of the 18-layer and 34-layer deep network of plain networks and ResNet networks on the imageNet dataset. The ResNet showed improvement over the deep network anomaly discussed in the previous section, and also showed that a deeper ResNet network can improve the error rate (as Figure 52 shows). For this reason, the ResNet152 was tested in the application presented in this paper.

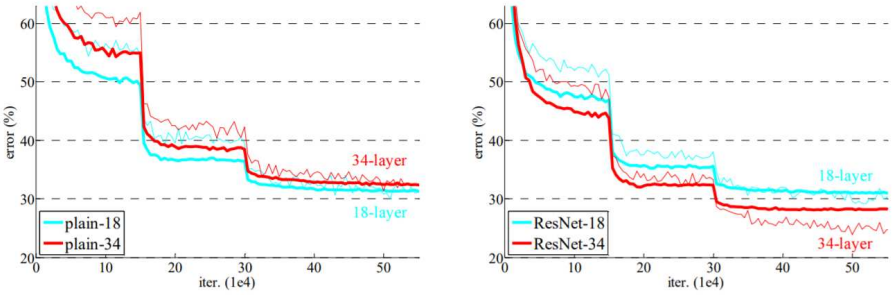


Figure 52 - (a) Performance of Plain Deep Network on ImageNet, (b) Performance of ResNet Network on ImageNet

ResNet152V2 Results

ResNet152V2 has consistently achieved accuracies above 89%, with the highest being 90.79% on the validation accuracy. The optimum loss achieved was 0.40 (as Figure 53 shows). It slightly outperformed ResNet152V2 as expected.

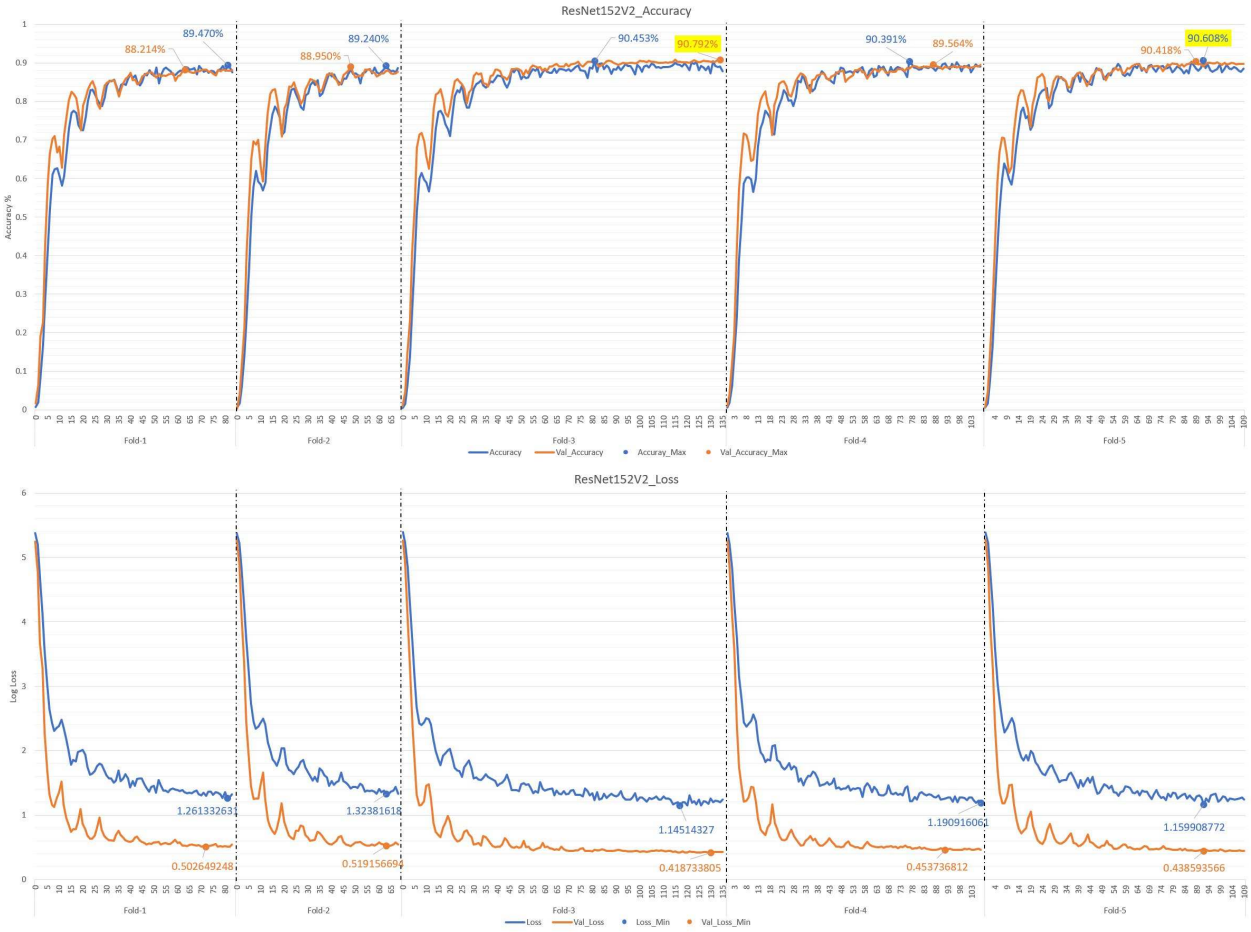


Figure 53 - (a) ResNet152V2 Accuracy, (b) ResNet152V2 Loss

ResNeXt50 Convolutional Neural Network

ResNeXt50 Architecture

The ResNet architecture is a convolutional neural network model that was proposed by Xie, Girshick, and other authors in 2017 in the paper "Aggregated Residual Transformation for Deep Neural Networks".[16]

The main premise of the paper is to introduce the concept of group operation to the ResNet architecture discussed in the prior sections. Starting with the ResNet bottleneck block, it

[16]S. Xie, R. Girshick, et al., "Aggregated Residual Transformation for Deep Neural Networks" Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR) 2017 pp. 1492-1500.

is equivalent to architecture (a) in the equivalent dashed block on Figure 54 by dividing the (256, 1x1, 64) into smaller blocks namely 32x (256, 1x1, 4). The reason for choosing this block size and this depth is because the two architectures thereby have the same number of parameters. This makes it easier to compare and benchmark the architecture performance properly. Concatenating the 2nd convolutional blocks, then projecting them on a higher space, is equivalent and computationally cheaper to perform. For this reason, the paper introduced block (b) in the equivalent dashed block on Figure 54. This operation by definition is called *group operation* which was first introduced above by the inception architecture. Table 17 shows the architecture differences between ResNet50 and ResNeXt50. The C stands for Cardinality which is the group operation discussed previously.

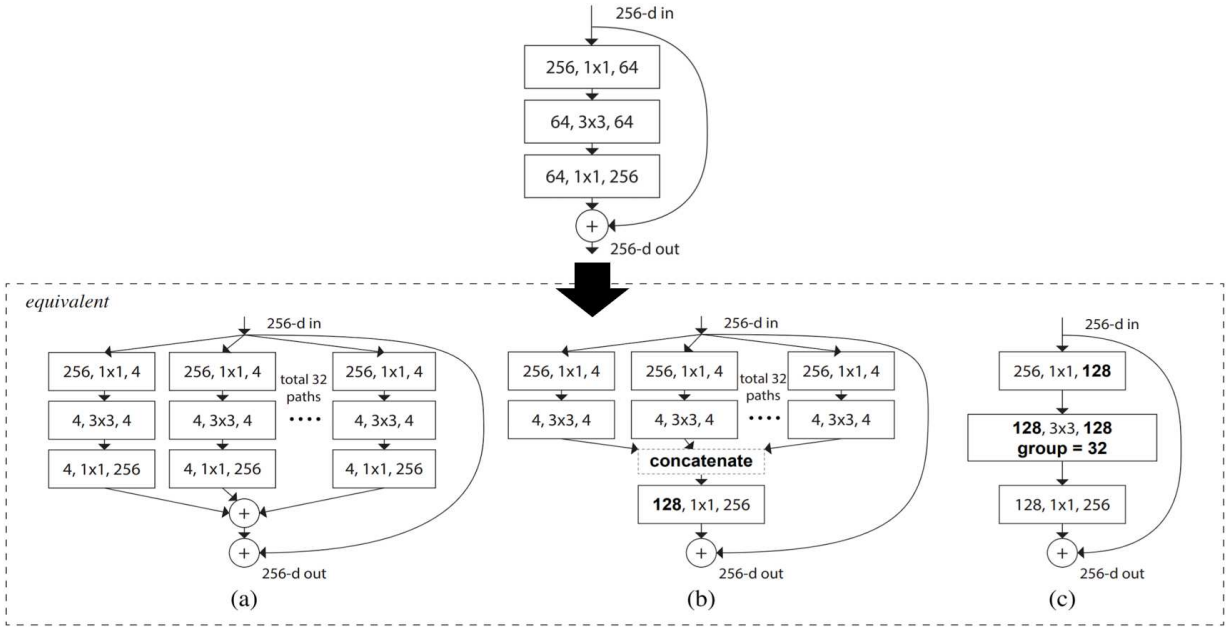


Figure 54 - (a) ResNet Building Block, (b) ResNeXt Building Block

Table 17 - ResNet50 vs ResNeXt50 Architecture

stage	output	ResNet-50	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		**25.5**×10^6	**25.0**×10^6
FLOPs		**4.1**×10^9	**4.2**×10^9

The paper showed the difference in performance between the ResNet50 and ResNeXt50 and, separately, the ResNet101 and ResNeXt101, on the ImageNet dataset (as Figure 55 shows below).

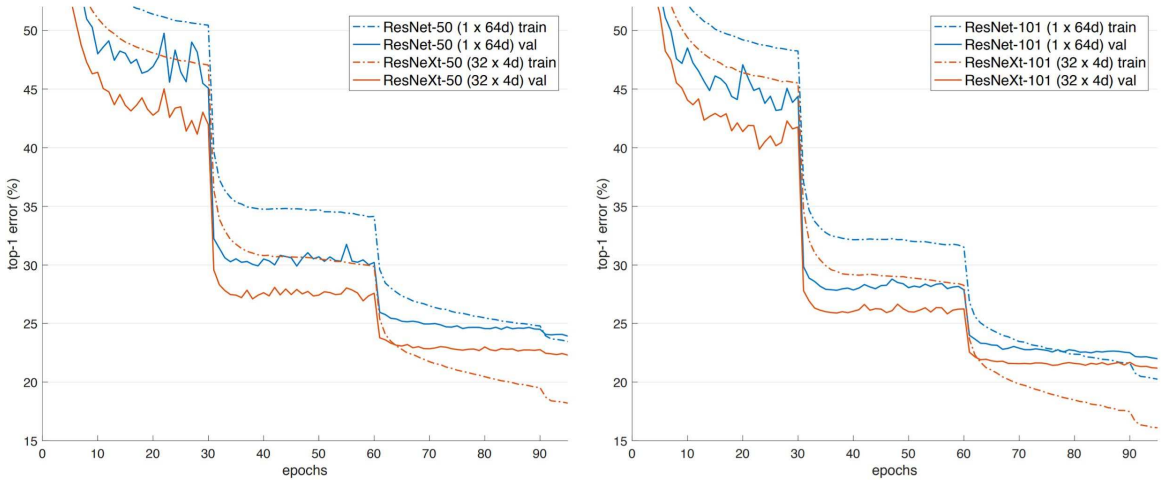


Figure 55 - (a) ResNet50 vs ResNeXt50 Performance, (b) ResNet101 vs ResNeXt101 Performance

ResNeXt50 Results

ResNeXt50 has consistently achieved accuracies above 90.5%, with the highest being 92.14% on the validation accuracy. The optimum loss achieved was 0.34 (as Figure 56 shows). ResNext50 outperformed all the architectures implemented in this project, achieving the highest accuracy and the lowest loss.

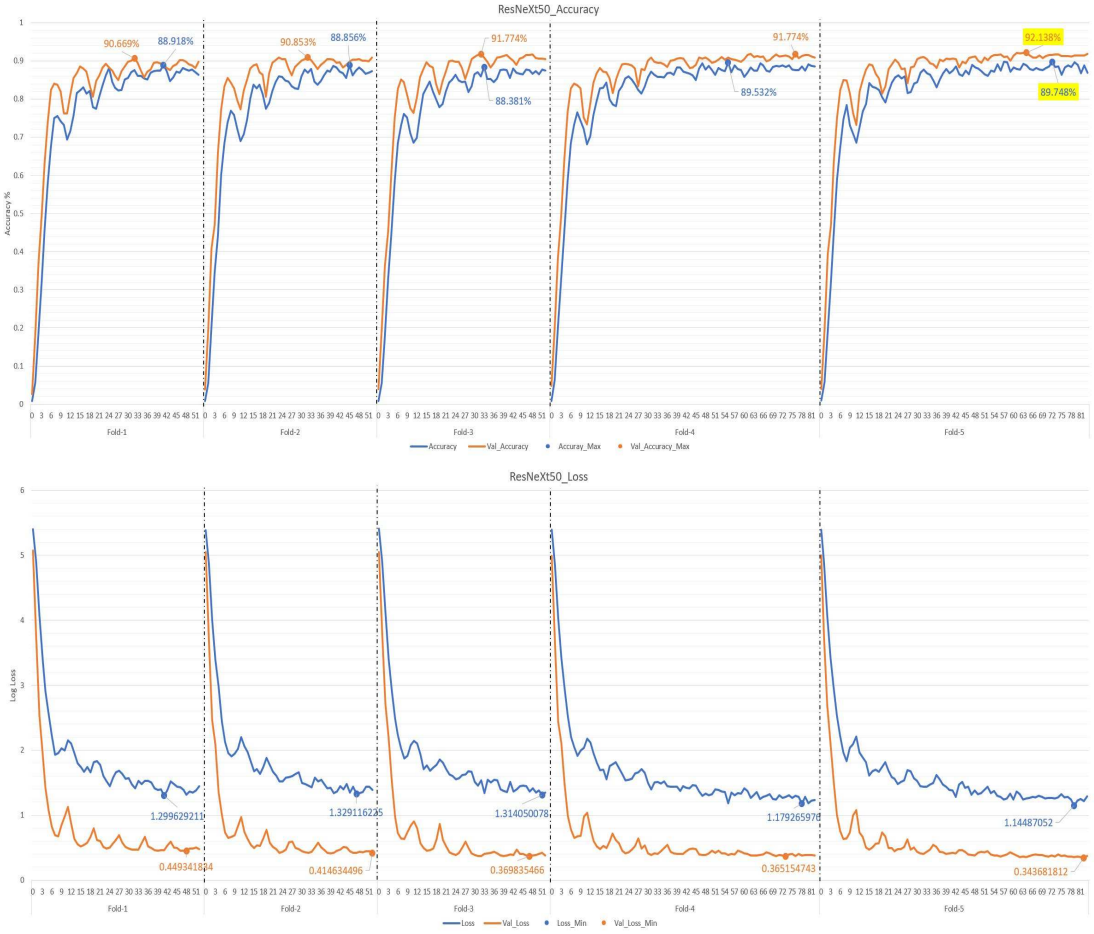


Figure 56 - (a) ResNeXt50 Accuracy, (b) ResNeXt50 Loss

Networks Performance Metrics and Network Comparisons

All the architectures are compared across various parameters as Table 18 and Table 19. Figure 57 shows these comparisons visually. These parameters are:

- Number of parameters
- Model size expressed in megabytes
- Validation accuracy
- Training time per epoch expressed in seconds
- Number of epochs to reach convergence
- Validation accuracy per fold
- Time to validate all images

The architecture with the highest number of parameters was the NASNetLarge with almost 90M parameters, while the smallest architecture parameter was the MobileNet, with only 3M parameters for the entire network. Another important observation is that the number of parameters of the ResNeXt50 and ResNet50V2 is the same size as discussed in the previous section, yet the accuracy was improved by almost 3% (as expected). The size of the network is dependent on the number of parameters; thus, the largest architecture was the NASNetLarge with a 349MB model. The smallest was the MobileNetV2, only 14MB in size. For this reason, the MobileNetV2 architecture is the most efficient on devices with limited resources such as drones and cell phones. The slowest architecture was the NASNetLarge, taking 810 seconds per epoch, while the fastest was MobileNetV2, taking only 33 seconds per epoch. The architecture that reached a convergent state with the least number of epochs was the NASNetLarge, followed by ResNeXt50. The architecture that took the longest until it reached convergence was MobileNetV2. Lastly, the best performing architecture achieving a mean accuracy of 92.5% was the ResNeXt architecture, while the lowest performing architecture excluding the VGG16 and VGG19 architectures was the InceptionResNetV2 architecture, achieving a mean accuracy of 89.7% accuracy. Table 20 and Table 21 show the paired T-Test between the various algorithms

as shown on Figure 57 last graph on the left. While some results show that is no significant statistical difference between the algorithms, this conclusion can't be inferred with high confidence due to the fact that the sample size is low. The Next section will discuss an approach to utilize a collaborative voting to enhance the accuracy of the overall system.

Figure 57 - CNN Architectures Performance Metrics

Table 18 - CNN Architectures Performance Metrics

	ResNeXt50	DenseNet201	Xception	NASNetLarge	ResNet152V2	InceptionV3	ResNet50V2	MobileNetV2	InceptionResNetV2	VGG16	VGG19
Number Of Parameters	25,347,204	20,489,988	25,257,708	89,247,510	60,630,724	24,101,860	25,863,876	3,770,628	56,111,524	15,440,900	20,750,596
Model Size (MB)	99	80	99	349	237	94	101	14	220	60	81
Validation Set Accuracy	92.138%	92.327%	91.529%	91.406%	90.792%	91.099%	89.626%	89.251%	89.441%	1.044%	1.105%
Training Time Per Epoch (s)	234	92	160	810	130	68	51	33	170	60	70
Number of Epochs until Convergence	65.2	83.6	102.4	51.8	101.4	75	110.6	139.8	94.2		

Table 19 - CNN Architectures Accuracy Metrics with Folds

Validation	ResNeXt50	DenseNet201	Xception	NASNetLarge	ResNet152V2	InceptionV3	ResNet50V2	MobileNetV2	InceptionResNetV2	VGG16	VGG19
Fold-1	91.069%	91.037%	91.651%	91.590%	90.546%	90.731%	90.731%	89.626%	89.994%	0.737%	0.737%
Fold-2	92.572%	92.818%	92.388%	92.818%	91.590%	91.897%	91.958%	91.344%	86.863%	1.044%	0.859%
Fold-3	92.634%	92.449%	92.204%	91.283%	92.449%	91.406%	91.467%	90.976%	89.503%	0.921%	0.921%
Fold-4	93.002%	92.756%	92.511%	91.958%	91.651%	92.449%	91.958%	89.564%	91.344%	0.798%	0.798%
Fold-5	93.612%	92.752%	92.076%	92.445%	92.076%	91.769%	91.523%	90.541%	91.216%	0.921%	0.676%
Mean Accuracy	**92.578%**	**92.363%**	**92.166%**	**92.019%**	**91.663%**	**91.650%**	**91.528%**	**90.410%**	**89.784%**	**0.884%**	**0.798%**
Time For Val. (s)	9	4	6	24	6	3	3	1	9	3	4

Table 20 - Paired T-Test

Paired T-Test (2 Tails)	ResNeXt50	DenseNet201	Xception	NASNetLarge	ResNet152V2	InceptionV3	ResNet50V2	MobileNetV2	InceptionResNetV2
ResNeXt50 T-Test		0.303167	0.292513	0.225324	0.022073	0.027041	0.024646	0.008723	0.025753
DenseNet201 T-Test			0.417191	0.316382	0.033957	0.012899	0.005253	0.004714	0.046266
Xception T-Test				0.603506	0.128822	0.030500	0.001751	0.006747	0.047974
NASNetLarge T-Test					0.445761	0.265274	0.110290	0.010737	0.078554
ResNet152V2 T-Test						0.970450	0.641427	0.015977	0.092163
InceptionV3 T-Test							0.319386	0.047587	0.086502
ResNet50V2 T-Test								0.029957	0.119922
MobileNetV2 T-Test									0.598646

Table 21 - Paired T-Test Interpretation

H0	There is no statistical significant difference between the various algorithms
H1	There is a statistical significant difference between the various algorithms

H0 between ResNeXt50 and DenseNet201	H0 between ResNeXt50 and Xception	H0 between ResNeXt50 and NASNetLarge	H1 between ResNeXt50 and ResNet152V2	H1 between ResNeXt50 and InceptionV3	H1 between ResNeXt50 and ResNet50V2	H1 between ResNeXt50 and MobileNetV2	H1 between ResNeXt50 and InceptionResNetV2
	H0 between DenseNet201 and Xception	H0 between DenseNet201 and NASNetLarge	H1 between DenseNet201 and ResNet152V2	H1 between DenseNet201 and InceptionV3	H1 between DenseNet201 and ResNet50V2	H1 between DenseNet201 and MobileNetV2	H1 between DenseNet201 and InceptionResNetV2
		H0 between Xception and NASNetLarge	H0 between Xception and ResNet152V2	H1 between Xception and InceptionV3	H1 between Xception and ResNet50V2	H1 between Xception and MobileNetV2	H1 between Xception and InceptionResNetV2
			H0 between NASNetLarge and ResNet152V2	H0 between NASNetLarge and InceptionV3	H0 between NASNetLarge and ResNet50V2	H1 between NASNetLarge and MobileNetV2	H0 between NASNetLarge and InceptionResNetV2
				H0 between ResNet152V2 and InceptionV3	H0 between ResNet152V2 and ResNet50V2	H1 between ResNet152V2 and MobileNetV2	H0 between ResNet152V2 and InceptionResNetV2
					H0 between InceptionV3 and ResNet50V2	H1 between InceptionV3 and MobileNetV2	H0 between InceptionV3 and InceptionResNetV2
						H1 between ResNet50V2 and MobileNetV2	H0 between ResNet50V2 and InceptionResNetV2
							H0 between MobileNetV2 and InceptionResNetV2

Networks Collaborative Approach

Last section discussed the performance differences between the various architectures implemented in this project. ResNeXt achieved consistently the best accuracy of 92.5%, followed by DenseNet201, Xception and NASNetLarge (all of which achieved mean accuracies of approximately 92%). In this section, a collaborative approach was implemented to test the overall improvement possible for the system if most or all of the architectures have a voting option. This approach was implemented from the idea about collaborative learning in classroom settings; in other words, not all confusion matrices of the architectures discussed have the same outcome. Thus, two voting schemas have been implemented by this project to enhance the outcome of a selected architecture. These voting schemas are equal voting schema, and a selective voting schema.

Equal Voting

Equal voting schema is a collaborative approach where all participating architectures have an equally weighted election also referred to as familiar voting pattern[17] [18]. Analogously, if a student is gathering information, he/she will gather the information from various teachers without any prior knowledge about their prior backgrounds. In this case, upon classifying a given image all participating architectures will vote on that class, the system will choose among the highest count of a given class. If there is a tie, the system will choose the same classification as ResNeXt since it achieved the highest outcome. Table 22 has the four architectures that have the highest accuracy participating in the collaborative voting; whereas, Table 23 has all nine architectures participating in the collaborative voting. As the results, on the first table where only

[17] S. Simske, "Meta-Algorithmics Patterns for Robust, Low Cost, High Quality Systems", 2013 John Wiley & Sons
[18] Dietterich, Thomas G. "Ensemble methods in machine learning." International workshop on multiple classifier systems. Springer, Berlin, Heidelberg, 2000

the highest accuracy architectures collaborated, the collaborative results raised the accuracy by 1.38% in the mean, achieving 93.67% as an overall accuracy of the system. On the second table where all the architectures collaborated, it achieved 93.64% accuracy and raised the accuracy by 1.91% in the mean.

Selective Voting

Selective voting schema is a collaborative approach where all participating architectures have a non-equally weighted election also referred to as Weighted Voting[17] [18]; the weights are dependent on prior knowledge of the participating networks performance. Analogously, if a student is gathering information about a science experiment, he/she will gather the information from various teachers while believing the teachers that have a scientific background more. In this case, upon classifying a given image all participating architectures will vote on that class, the system will assign a weight to all the participating networks answers then choose the highest count of a given class. If there is a tie, the system will choose the same classification as ResNeXt since it achieved the highest outcome. Two weights were implemented: The accuracy as weight, and 1/error as a weight. Table 24 and Table 25 have the four architectures participating that have the highest accuracy in the collaborative voting with weights assigned to these networks as accuracy and 1/error respectively. Table 26 and Table 27 have all nine architectures participating in the collaborative voting with weights assigned as accuracy and 1/error respectively. As the results, on the first tables where only the highest accuracy architectures are used in collaboration, the collaborative results raised the accuracy by 1.36% in the mean achieving 93.65% as an overall accuracy of the system. When all of the architectures are used in the collaboration, it achieved 93.5% accuracy and raised the accuracy by 1.95% in the mean. The collaboration techniques with selective voting achieved similar, but slightly lower, accuracy improvement than

the equal voting approaches, because they slightly accentuate the error of the high performing architectures.

In conclusion, the collaborative techniques enhanced the overall accuracy of the system by lowering the error by more than 10% compared to the best individual architecture.

Table 22 - Best 4 CNN Architectures with Equal Voting

4- Collective Collaboration Validation Equal Voting				
	ResNeXt50	**DenseNet201**	**Xception**	**NASNetLarge**
Fold-1	0.910988	0.910374	0.916513	0.915899
Fold-2	0.925721	0.928177	0.92388	0.928177
Fold-3	0.926335	0.924494	0.922038	0.91283
Fold-4	0.930018	0.927563	0.925107	0.919583
Fold-5	0.936157	0.927563	0.92081	0.924494
Mean Accuracy	**92.584%**	**92.363%**	**92.167%**	**92.020%**
Overall Accuracy Score	**93.665%**			
Improvement %	1.080%	1.301%	1.498%	1.645%

Table 23 - Best 9 Architectures with Equal Voting

9- Collective Collaboration Validation Equal Voting									
	ResNeXt50	DenseNet201	Xception	NASNetLarge	ResNet152V2	InceptionV3	ResNet50V2	MobileNetV2	InceptionResNetV2
Fold-1	91.099%	91.037%	91.651%	91.590%	90.546%	90.731%	90.731%	89.626%	89.994%
Fold-2	92.572%	92.818%	92.388%	92.818%	91.590%	91.897%	91.958%	91.344%	86.863%
Fold-3	92.634%	92.449%	92.204%	91.283%	92.449%	91.406%	91.467%	90.976%	89.503%
Fold-4	93.002%	92.756%	92.511%	91.958%	91.651%	92.449%	91.958%	89.564%	91.344%
Fold-5	93.616%	92.756%	92.081%	92.449%	92.081%	91.774%	91.529%	90.546%	91.222%
Mean Accuracy	**92.584%**	**92.363%**	**92.167%**	**92.020%**	**91.664%**	**91.651%**	**91.529%**	**90.411%**	**89.785%**
Overall Accuracy Score	**93.481%**								
Improv. %	0.896%	1.117%	1.314%	1.461%	1.817%	1.829%	1.952%	3.069%	3.696%

Table 24 - Best 4 CNN Architectures with Selective Voting with Accuracy Weight Measurement

4- Collective Collaboration Validation Selective Voting (accuracy Weight)				
	ResNeXt50	**DenseNet201**	**Xception**	**NASNetLarge**
Fold-1	0.910988	0.910374	0.916513	0.915899
Fold-2	0.925721	0.928177	0.92388	0.928177
Fold-3	0.926335	0.924494	0.922038	0.91283
Fold-4	0.930018	0.927563	0.925107	0.919583
Fold-5	0.936157	0.927563	0.92081	0.924494
Mean Accuracy	**92.584%**	**92.363%**	**92.167%**	**92.020%**
Overall Accuracy Score	**93.653%**			
Improvement %	1.068%	1.289%	1.486%	1.633%

Table 25 - Best 4 CNN Architectures with Selective Voting with 1/error Weight Measurement

4- Collective Collaboration Validation Selective Voting (1/error Weight)				
	ResNeXt50	**DenseNet201**	**Xception**	**NASNetLarge**
Fold-1	0.910988	0.910374	0.916513	0.915899
Fold-2	0.925721	0.928177	0.92388	0.928177
Fold-3	0.926335	0.924494	0.922038	0.91283
Fold-4	0.930018	0.927563	0.925107	0.919583
Fold-5	0.936157	0.927563	0.92081	0.924494
Mean Accuracy	**92.584%**	**92.363%**	**92.167%**	**92.020%**
Overall Accuracy Score	**93.640%**			
Improvement %	1.056%	1.277%	1.473%	1.621%

Table 26 - Best 9 CNN Architectures with Selective Voting with Accuracy Weight Measurement

9- Collective Collaboration Validation Selective Voting (Accuracy)									
	ResNeXt50	DenseNet201	Xception	NASNetLarge	ResNet152V2	InceptionV3	ResNet50V2	MobileNetV2	InceptionResNetV2
Fold-1	91.099%	91.037%	91.651%	91.590%	90.546%	90.731%	90.731%	89.626%	89.994%
Fold-2	92.572%	92.818%	92.388%	92.818%	91.590%	91.897%	91.958%	91.344%	86.863%
Fold-3	92.634%	92.449%	92.204%	91.283%	92.449%	91.406%	91.467%	90.976%	89.503%
Fold-4	93.002%	92.756%	92.511%	91.958%	91.651%	92.449%	91.958%	89.564%	91.344%
Fold-5	93.616%	92.756%	92.081%	92.449%	92.081%	91.774%	91.529%	90.546%	91.222%
Mean Accuracy	**92.584%**	**92.363%**	**92.167%**	**92.020%**	**91.664%**	**91.651%**	**91.529%**	**90.411%**	**89.785%**
Overall Accuracy Score	**93.481%**								
Improv. %	0.896%	1.117%	1.314%	1.461%	1.817%	1.829%	1.952%	3.069%	3.696%

Table 27 - Best 4 CNN Architectures with Selective Voting with 1/error Weight Measurement

9- Collective Collaboration Validation Selective Voting (1/error)									
	ResNeXt50	DenseNet201	Xception	NASNetLarge	ResNet152V2	InceptionV3	ResNet50V2	MobileNetV2	InceptionResNetV2
Fold-1	91.099%	91.037%	91.651%	91.590%	90.546%	90.731%	90.731%	89.626%	89.994%
Fold-2	92.572%	92.818%	92.388%	92.818%	91.590%	91.897%	91.958%	91.344%	86.863%
Fold-3	92.634%	92.449%	92.204%	91.283%	92.449%	91.406%	91.467%	90.976%	89.503%
Fold-4	93.002%	92.756%	92.511%	91.958%	91.651%	92.449%	91.958%	89.564%	91.344%
Fold-5	93.616%	92.756%	92.081%	92.449%	92.081%	91.774%	91.529%	90.546%	91.222%
Mean Accuracy	**92.584%**	**92.363%**	**92.167%**	**92.020%**	**91.664%**	**91.651%**	**91.529%**	**90.411%**	**89.785%**
Overall Accuracy Score	**93.579%**								
Improv. %	0.995%	1.215%	1.412%	1.559%	1.915%	1.928%	2.050%	3.168%	3.794%

CHAPTER 4 – COMMON COORDINATE SYSTEM

As previously discussed, object tracking is an active research area in computer vision thanks to the increasing demands in the Intelligence, Surveillance and Reconnaissance (ISR) applications and the Autonomous Vehicles Systems (AVS). Many algorithms have been developed to track the Object of Interest (OOI) across the view of the camera, and even predict its position when it is obfuscated; however, the tracking system doesn't coordinate its finding about the OOI position with nearby cameras. This section discusses ways to resolve this issue, and will introduce a method to unify the mesh of cameras to a common coordinate system and relay information about the OOI on a common grid with and without prior knowledge of the location and orientation of the cameras as shown on Figure 58 below.

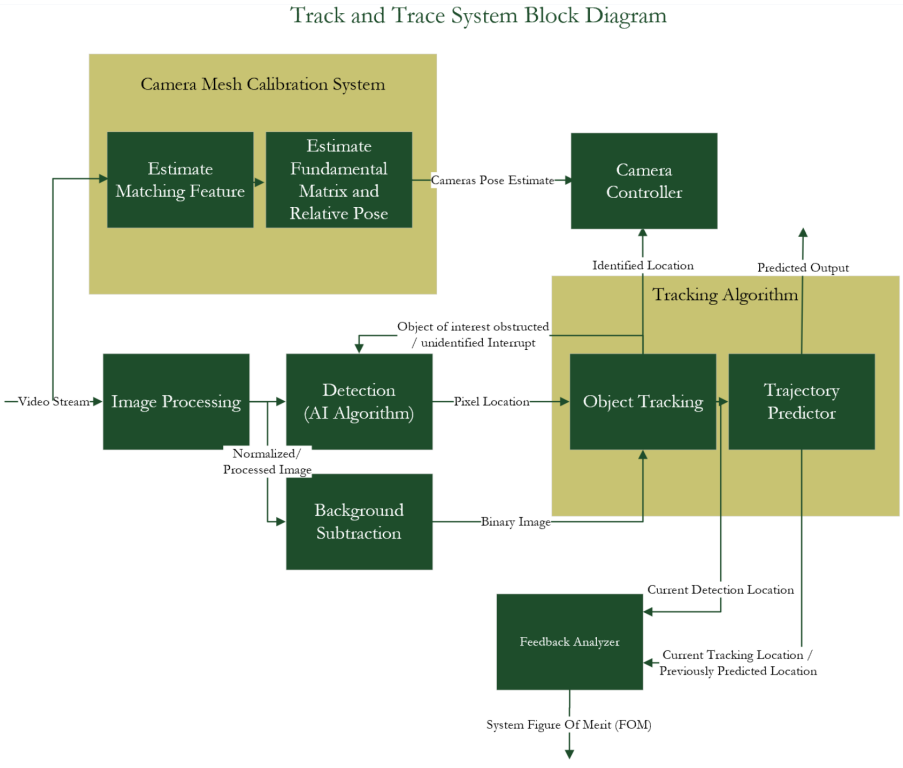


Figure 58 - Enhanced System Block Diagram

Surveyed Positions Solution

This solution requires to have a priori knowledge of the location and orientation of each camera in the mesh[19].

Assuming that latitude and longitude (lat/long) of each camera are expressed as $\varphi \; and \; \lambda$(radians) respectively; then the distance can be calculated using the great-circle between two points, also known as the 'haversine' formula. In the case shown on Figure 59 below, the focal center of the image was chosen to be the reference lat/long point of the camera.

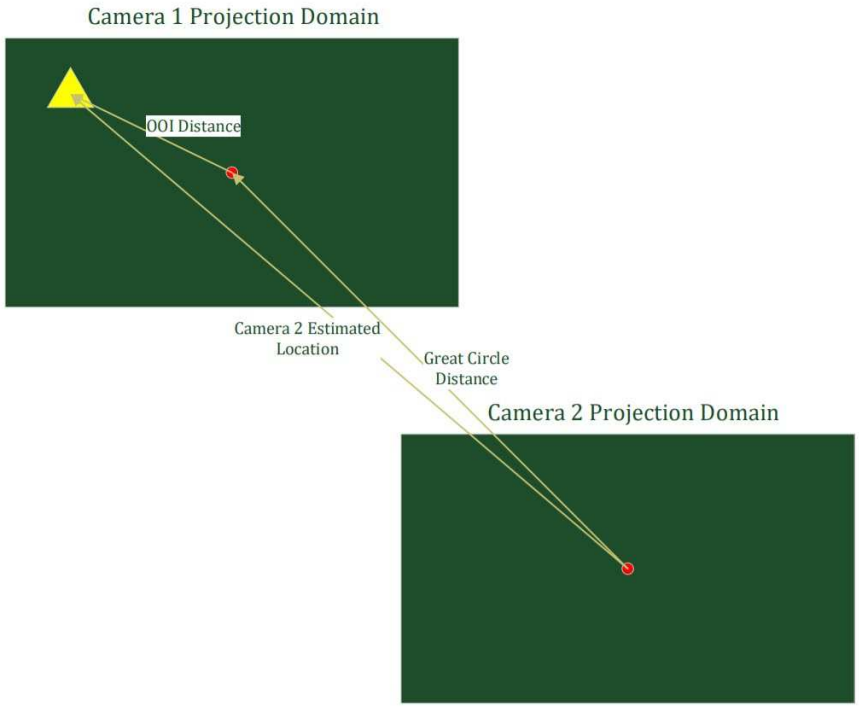


Figure 59 - Surveyed Cameras Positions Solution Concept

The distance is calculated as following:

$$a \; = sin^2(\frac{\Delta\varphi}{2}) \; + cos(\varphi_1)\, cos(\varphi_2)\, sin^2(\frac{\Delta\varphi}{2})$$

[19] V. Nastro and U. Tancredi, "Great Circle Navigation with Vectorial Methods" *The Journal of Navigation*, Vol. 63, Iss. 3, 557-563 Cambridge, July, 2010.

$$c = 2\,atan2(\sqrt{(a)}, \sqrt{(1-a)})$$

$$d = R\,c$$

Where R is the earth mean radius (6,371km), a is the square of half the chord length between the points and c is the angular distance in radians. To properly localize the object of interest (OOI) a stereo vision system must be in place. The vector $\underline{v}$ pointing to the OOI centroid is expressed by an azimuth and elevation. Thus, the distance from any camera on the network can be calculated by adding these aforementioned steps. The advantage of this method is its accuracy of defining the mesh parameters; whereas the main disadvantage of this method is the amount of information needed makes it harder to be autonomous and self-calibrating.

Distant Point Calibration Solution

Given two stereo camera systems, the general idea of this method to determine where the OOI is with respect to a second stereo camera system is to determine, via the calibration process, the relative alignment of the cameras using the "distant point method" explained below and the relative position vector connecting the cameras. The relative position vector is determined by using the cameras relative alignment and their view the same object that is close enough for a reasonably accurate position determination from the stereo cameras; in other words, no external position survey is required. Using the results of the aforementioned camera calibration which will be discussed later in this section, and the first camera system's position vector measurement of the OOI position, the expected position of the OOI with respect to the second camera can be calculated. The OOI's expected position with respect to the second camera can be far out of its field of view because that position can be converted to a fully spherical azimuth and elevation to which the second camera can be commanded to point. The advantages of this method are:

- The expected azimuth and elevation with respect to the second camera can be far out of its field of view (e.g. behind or far above where the camera is pointing).
- The mathematics is much simpler, and therefore easier to debug than the mathematics of determining the camera's relative alignment and relative positions using multiple parallax observations of the same objects by two non-stereo camera systems
- Trajectory estimation/prediction are not required.

Once the camera calibration is completed, the equation below provides the OOI expected position with respect to the second camera given the information from the first camera from which an azimuth and elevation can calculated with the following equation and as Figure 60 shows below.

$$[R_2^{OOI}]_2 = [R_2^1]_2 + C_1^2[R_1^{OOI}]_1$$

Where,

$[R_2^{OOI}]_2$ is the position vector of OOI relative to camera 2 in camera 2 frame of reference coordinate system.

$[R_2^1]_2$ is the position vector of camera 1 relative to camera 2 in camera 2 frame of reference coordinate system.

C_1^2 is the direction cosine matrix which transforms camera 1 vector to coordinates to vector coordinates of camera 2 frame of reference coordinate system.

Lastly, $[R_1^{OOI}]_1$ is the position vector of OOI relative to camera 1 in camera 1 frame of reference coordinate system.

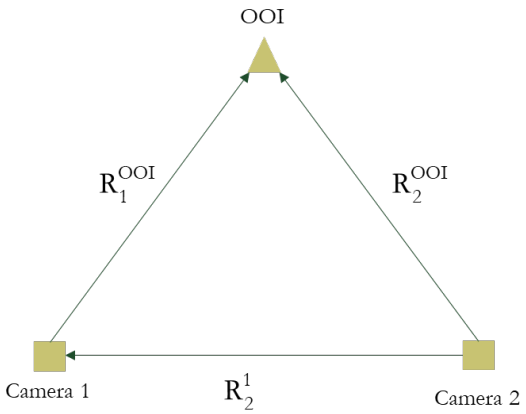


Figure 60 - OOI Position Calculation

When equation described above is solved for the "position vector of camera 1 relative to camera 2 in camera 2 coordinates" the position measurements made by the two stereo cameras provide the calibration process's determination of the relative position between the cameras. If a position survey were to be used, the camera's orientation relative to the Earth would be needed because survey coordinates such as latitude, longitude, and altitude are relative to Earth Centered Earth Fixed axis. Obtaining the camera's orientation relative to Earth would be very inconvenient. The distant point method of determining the camera's relative alignment will finally be discussed. The fundamental principal employed is that the directions of position vectors connecting the cameras to distance points such as stellar constellations do not depend on the camera's position. Thus, if the two cameras measure the directions specified by a unit vector or equivalently azimuth and elevation of three distant points, two different views of the same coordinate system are obtained. The coordinates of each camera's view of the common coordinate system is used to determine the direction cosine matrix relating the cameras.

Each distant point of the three will be expressed in a different coordinate system for each of the camera. We can express these unit vectors to the three points with $\widehat{d_1}, \widehat{d_2}, \widehat{d_3}$. We can

orthogonize the system using the Gram-Schmidt as the following equations then normalize the system.

$$\{\widehat{D_1} = \widehat{d_1}\ \widehat{D_2} = \widehat{d_2} - \frac{\widehat{d_2^T}\ \widehat{D_1}}{\widehat{D_1^T}\ \widehat{D_1}}\ \widehat{D_1}\ \widehat{D_3} = \widehat{d_3} - \frac{\widehat{d_3^T}\ \widehat{D_1}}{\widehat{D_1^T}\ \widehat{D_1}}\ \widehat{D_1} - \frac{\widehat{d_3^T}\ \widehat{D_2}}{\widehat{D_2^T}\ \widehat{D_2}}\ \widehat{D_2}$$

Where $\widehat{D}$ the axis system is common to all the cameras but different coordinates for each camera; in other words, they are different coordinate because each camera is pointed differently; however, unit vectors $\widehat{D_1}, \widehat{D_2}, \widehat{D_3}$ point in the same direction because the points are too far away.

The Direction Cosine Matrix $C_{axis\ 1}^{axis\ 2}$ transforms the axis from system 1 to system 2. It is expressed in the following matrix:

$$C_{axis\ 1}^{axis\ 2} = [\langle\widehat{x_2}, \widehat{x_1}\rangle\ \langle\widehat{x_2}, \widehat{y_1}\rangle\ \langle\widehat{x_2}, \widehat{z_1}\rangle\ \langle\widehat{y_2}, \widehat{x_1}\rangle\ \langle\widehat{y_2}, \widehat{y_1}\rangle\ \langle\widehat{y_2}, \widehat{z_1}\rangle\ \langle\widehat{z_2}, \widehat{x_1}\rangle\ \langle\widehat{z_2}, \widehat{y_1}\rangle\ \langle\widehat{z_2}, \widehat{z_1}\rangle\ ]$$

Where $\langle\widehat{x_2}, \widehat{x_1}\rangle$ are the inner product of vectors $\widehat{x_2}\ and\ \widehat{x_1}$; where $\widehat{x_1}, \widehat{y_1}\ and\ \widehat{z_1}$ are the basis vector coordinates of the two axis systems

Thus, changing from the old coordinate to the new coordinate can be expressed as following:

$$[x_2\ y_2\ z_2\ ] = C_{axis\ 1}^{axis\ 2}\ [x_1\ y_1\ z_1\ ]$$

The Direction Cosine Matrix C_1^D, C_D^2 relating the two cameras, where C_1^D transforms a given vector from camera 1 axis to D axis described above, and C_D^2 transforms a given vector from D axis to camera 2 axis can be calculated as following:

$$C_1^D = \left[D_{1x}\ D_{1y}\ D_{1z}\ D_{2x}\ D_{2y}\ D_{2z}\ D_{3x}\ D_{3y}\ D_{3z}\ \right]\ ,$$
$$C_D^2 = \left[D_{1x}\ D_{2x}\ D_{3x}\ D_{1y}\ D_{2y}\ D_{3y}\ D_{1z}\ D_{2z}\ D_{3z}\ \right]$$

Where $\widehat{D}$ is the new axis coordinate system in the C_1^D transformation where its components are calculated by the Gram-Schmidt above and expressed as the following:

$$\widehat{D_1} = \left[D_{1x}\ D_{1y}\ D_{1z}\ \right]\ ,\ \widehat{D_2} = \left[D_{2x}\ D_{2y}\ D_{2z}\ \right]\ ,\ \widehat{D_3} = \left[D_{3x}\ D_{3y}\ D_{3z}\ \right]$$

Thus, the transformation from camera 1 to camera 2 can be calculated as following $C_1^2 = C_D^2 \, C_1^D$.

The main disadvantage of this method is that a stereo camera needs to be used for every camera position.

Point Correspondence Solution

This method of coordinating and calibrating the camera network relies on overlap between the cameras. The system is composed of two subsystems. The first subsystem extracts the matching features between two frames of a video feeds from two difference sources. This is done by detecting the edges and corner, then it extracts the neighborhood features to these corners and edges. Next it finds the matching features in the correspondent image. The subsystem is shown on Figure 61 below.

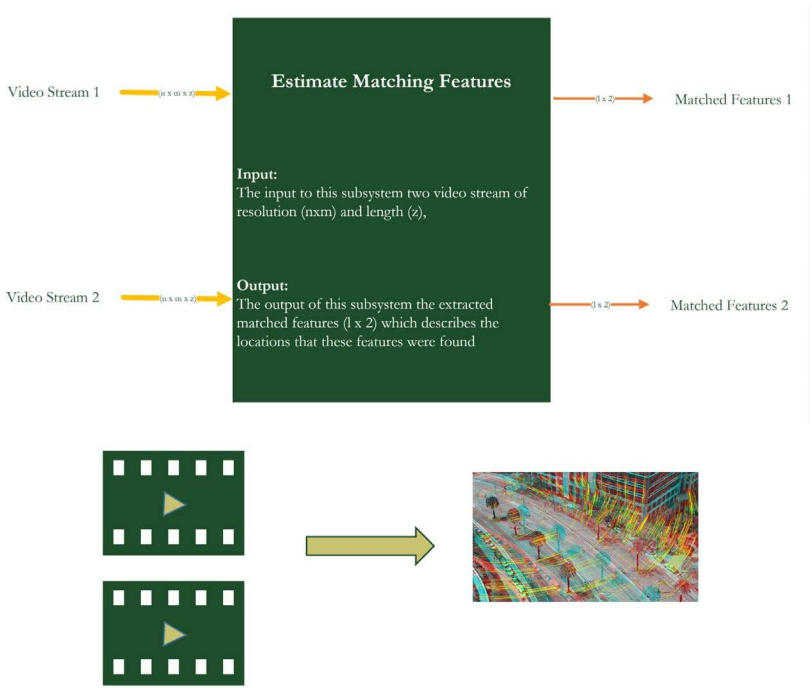


Figure 61 - Estimate Matching Features Subsystem

The second subsystem estimates the Fundamental Matrix F and estimates the relative pose between the two cameras; in other words, it estimates the relative rotation and translation between the cameras[20]. There are several methods to estimate the Fundamental Matrix for example:

- The Random Sample Consensus algorithm (RANSAC)
- The M-Estimator Sample Consensus (MSAC) which converges faster than RANSAC.
- The Least Median Squares algorithm (LMedS).
- Least Trimmed Squares (LTS) which converges faster than LMedS.
- Or by the 8 point correspondent algorithm developed by Longuet-Higgins.

The estimate pose is calculated and it is dependent on the camera intrinsic calibration. Figure 62 shows the block diagram of the second subsystem.

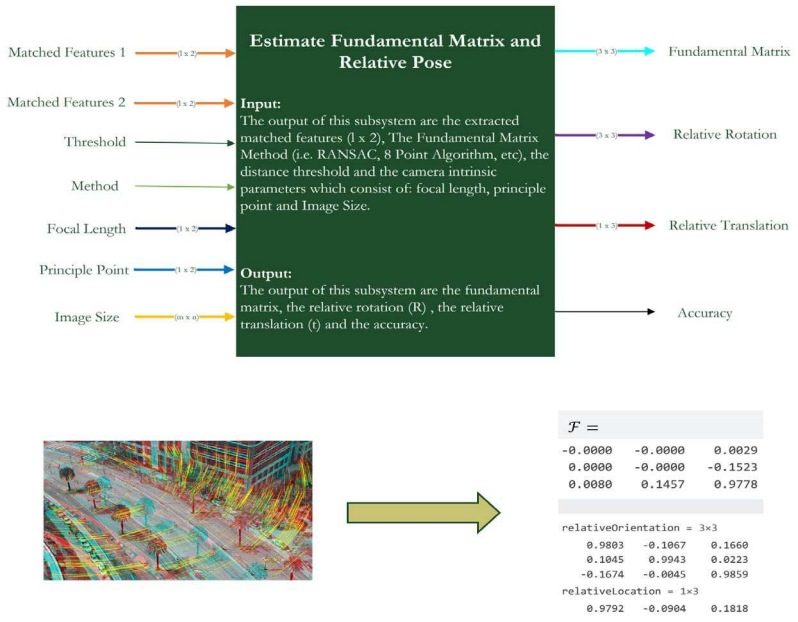


Figure 62 - Estimate Fundamental Matrix and Relative Pose Subsystem

[20] QT. Luong and O.D. Faugeras, "The fundamental matrix: Theory, algorithms, and stability analysis." *Int J Computer Vision* 17, 43–75, 1996.

In other words, this method leverages the stereo vision concept and applies it to a much higher scale. If camera X locates the Object of Interest, All the cameras in the network can coordinate their relative position to that camera X, by performing the cumulative transformation. The next section will describe the algorithms used for each subsystem and will show the results.

Point Correspondence Solution Data Gathering

The data was generated by a movie created by Google Earth Studio, the cameras were then placed at random positions where there was some overlap between them. Figure 63 shows below the trail of the camera where each white dot shows the major keyframe that was used as camera placement.

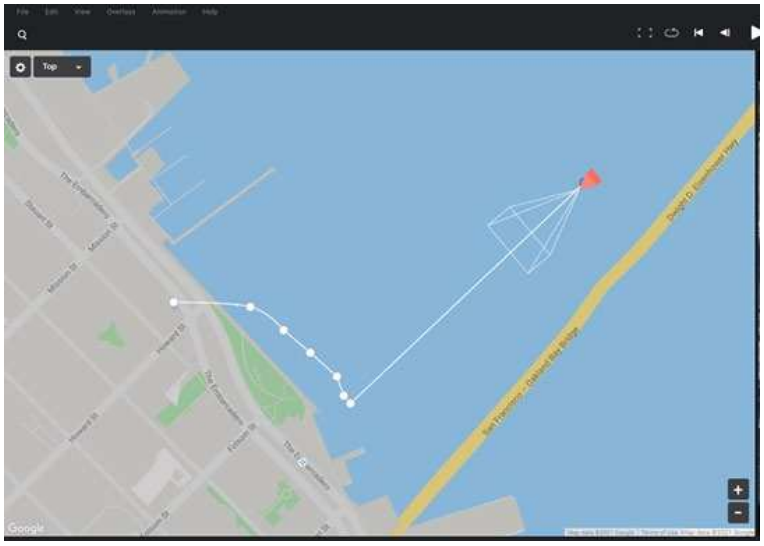


Figure 63 - Synthesized Data Overview

Point Correspondence Solution Detailed Algorithm

In the first subsystem mentioned above, the features of the image are detected using the Harris Features detection algorithm. Then the features were extracted between the images by a combination of algorithms namely Speeded-Up Robust Features (SURF) and Fast Retina

Keypoint algorithms. Then these features get corresponded between the images. Figure 64 visualizes the point correspondence between the two algorithms.

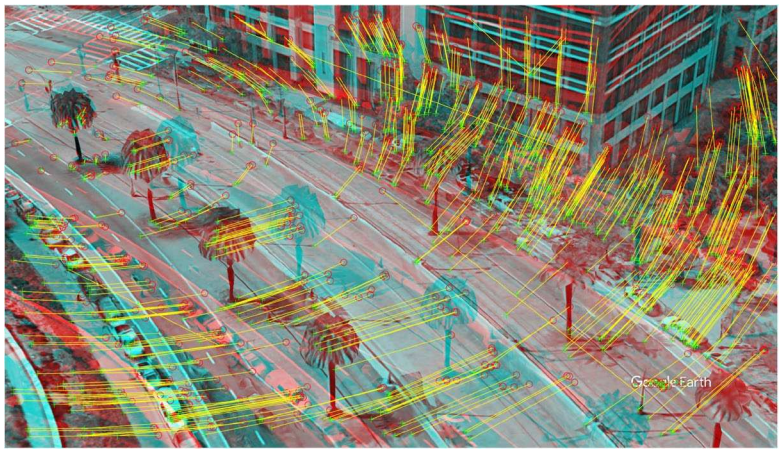


Figure 64 - Correspondence Features

Figure 65 below shows the strongest corresponding points between the two images after removing the outliers.

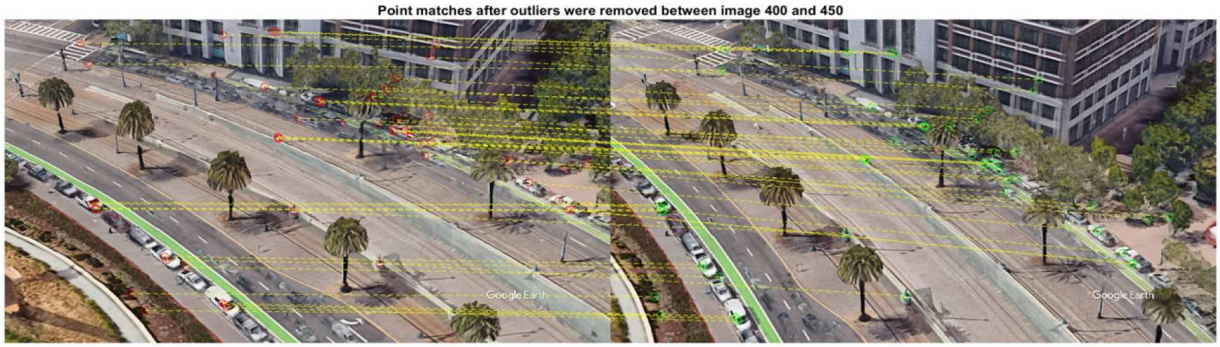


Figure 65 - Synthesized Data Overview

In the second subsystem, the fundamental matrix is generated by the Random Sample Consensus (RANSAC) algorithm such that the following equation is satisfied.

$$x_2^T F\, x_1 = 0$$

To estimate the relative location of the cameras, the intrinsic properties of the cameras were assumed to be ideal for distortion and skew factors since the data was synthesized. The focal

length is assumed to be 3000 millimeters in the x and y directions and the optical center of the camera is exactly in the middle. The focal length for simulated data is infinite, an analysis was performed to achieve a realistic estimate. Focal length was tested from 1600 to 5000, beyond that 3000 there was insignificant improvement; thus, focal length of 3000 was chosen. Figure 66 below shows an example of the result of the relative orientation and relative position of the two cameras. Where camera 1 (on left) is placed at the origin (0,0,0) and camera two (on the right) is relatively placed based on the position described by the Rotation and Translation matrices.

```
Relative Position and Orientation between Camera 400 and camera 450
relativeOrientation = 3×3
    0.9806   -0.1068    0.1642
    0.1055    0.9943    0.0165
   -0.1651    0.0011    0.9863
relativeLocation = 1×3
    0.9717   -0.0981    0.2150
validPointsFraction = 1
```

Figure 66 - Relative Position and Orientation

Scene Reconstruction and Testing

To assess the proposed algorithm, the scene has been reconstructed by triangulating the matched points calculated by the correspondence algorithm that was discussed in the previous section. Figure 67 below shows the scene reconstruction.

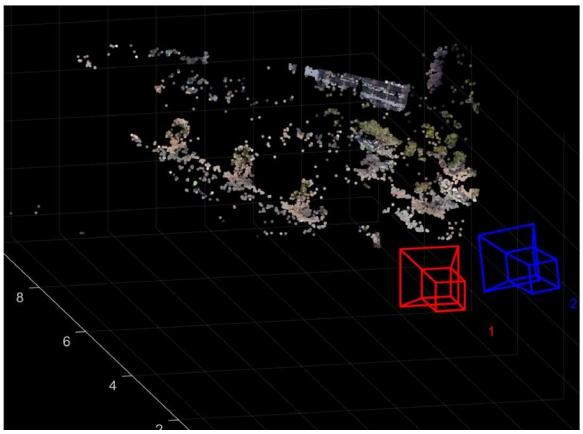


Figure 67 - Scene Reconstruction

By inserting more overlapping cameras in the scene, the accuracy and the quality of the photogrammetry will increase substantially. To calculate the error from the reconstructed scene, several points were chosen randomly to remap them into the projected space onto the two images, as shown by Figure 68 and Figure 69 below. These set of images show there is an error when comparing the two images; for instance, the point chosen shows that it is at the corner of the building by the middle of the window, whereas the second camera remaps it into the corner of the building by the top of the window.

Figure 68 - First Example (a) Remap onto Camera1 Scene, (b) Remap onto Camera2 Scene

Similarly, for the second example, there was an error when comparing the two images. Thus, the next section will discuss the method used to quantify the associated error.

Figure 69 - Second Example (a) Remap onto Camera1 Scene, (b) Remap onto Camera2 Scene

Error Estimation

The error was estimated by performing a normalized 2-D cross-correlation between a template taken from image one and a section of image 2. For example, Figure 70(a) shows the template to be chosen as the 50x50 pixels from the center of the remapped point from camera 1;

Figure 70(b and c) show a window around the of 200x200 pixels from the centers of the remapped points from camera 1 and camera 2 respectively.

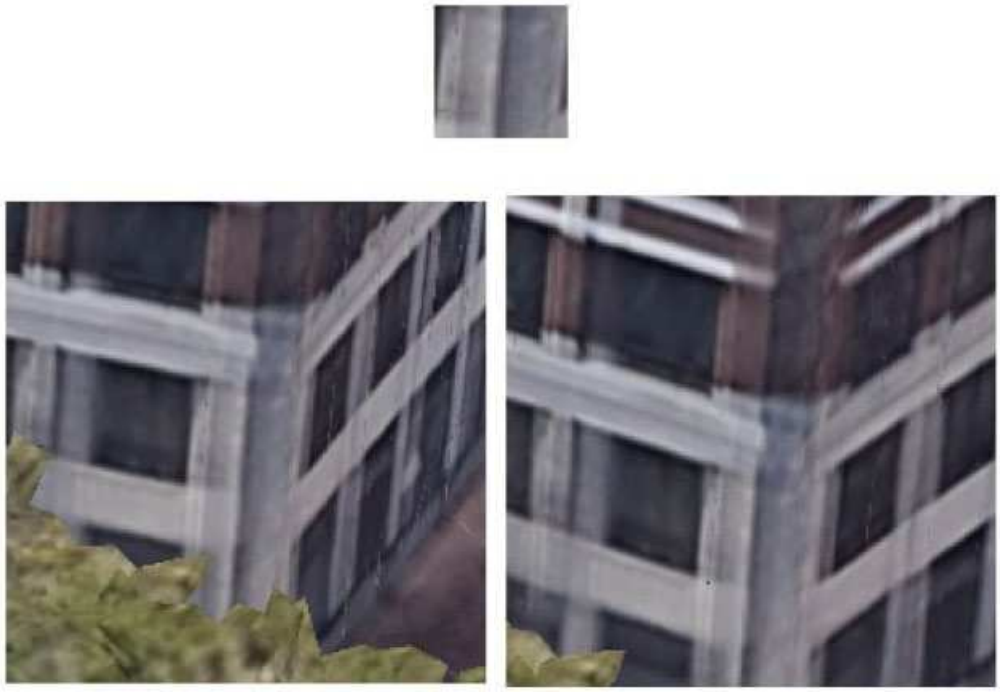


Figure 70 - (a) Template, (b) Remapped ROI onto Camera1 Scene, (c) Remapped ROI onto Camera2 Scene

Figure 71 shows the results of the 2-D cross-correlation as a surface map, where x,y are the pixels of the image and the z axis is the correlation coefficient magnitude.

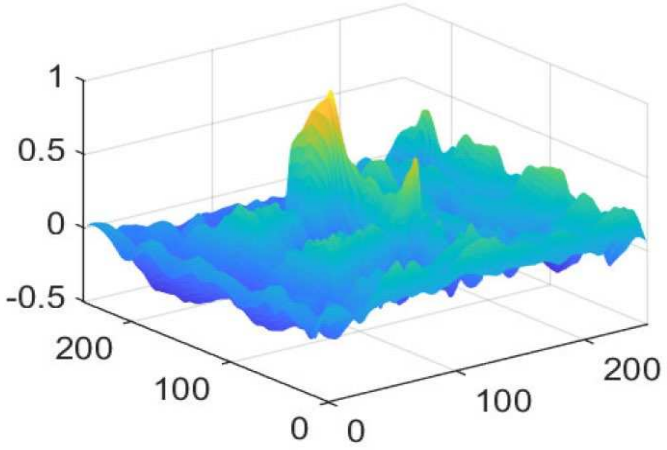


Figure 71 - Cross-Correlation Result

Figure 72 below shows the estimated error. Figure 72(a) shows the original location from camera 1 indicated by the blue-cross/dark-cross; whereas, Figure 72(b) shows the original mapped location indicated by the red-cross/dark-cross and the found location with the cross-correlation indicated by the green-cross/light-cross. The error is estimated to be -38 in the Y direction and about -2 in the X direction.

Figure 72 - (a) Original Mapped Location onto Camera1 Scene, (b) Error Estimated onto Camera2 Scene

Similarly, the same process was done to the second random point discussed above. In this case, the error was estimated to be -40 in the Y direction and 1 in the x direction as shown on Figure 73.

Figure 73 - (a) Original Mapped Location onto Camera1 Scene, (b) Error Estimated onto Camera2 Scene

Future Enhancements

The first enhancement to the proposed algorithm will deal with optimizing the cameras distances and pose estimation[21], where the minimum overlap between the two images will be estimated. Another enhancement is to optimize the execution time; currently the execution time for 7 cameras is approximately 15.2 seconds. Another enhancement to the system is to implement a method that mimics the idea of MPEG-2 "I" frame to change the frame of reference after N number of cameras to minimize the error from accumulating the rotation and translation from one camera to another. Lastly, the aforementioned system will be integrated with the overall system described by the previously published paper to locate the Object Of Interest (OOI)

[21] A. Trabelsi, M. Chaabane, N. Blanchard and R. Beveridge, "A Pose Proposal and Refinement Network for Better 6D Object Pose Estimation." *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2382-2391, 2021.

CHAPTER 5 – CAMERA POSITION OPTIMIZATION

Introduction

The previous chapter, a method to unify the camera system to a common coordinate scheme was introduced. This method uses the feature overlap between the two cameras without addressing the optimized camera positions. In other words, it didn't answer the following questions: (1) "How many cameras are needed to cover a given scene?" (2) What is the optimum position for each camera to minimize the numbers of cameras needed?" In real-life scenarios, it is necessary (or at least cost-sensitive and thus practical) to utilize the minimum number of cameras to cover a given scene. Answering the previous two questions is essential for autonomous systems applications.

In drone applications, estimating the overlap between the captured drone images has to be done in real time; for this reason, one of the main requirements is that it must be done by an efficient function that can adapt to the scene changes seamlessly. Performing such functions with neural networks and object recognition is not feasible due to the fact that deep learning and scene recognition are not trivial functions and may not be robust to changes in lightning, perspective, and other variable imaging concerns. Thus, more rudimentary signal and image processing applications are essential in such applications.

Overlap in a given scene can be thought of as analogous to an echo in voice application signals. One of the most effective algorithms developed in the early 60s[22] to detect and remove the echo in voice applications is the Cepstral analysis.

[22] B. Bogert, et al: "The Quefrency Alanysis of Time Series for Echoes: Cepstrum, Pseudo Autocovariance, Cross-Cepstrum and Saphe Cracking" M. Rosenblatt, Ed Chap.15, 1963.

Theory

The word "cepstral" is the juxtaposition of the first four letters of the word "spectral." Cepstrum analysis is a nonlinear mapping of the magnitude spectrum in order to redistribute the energy of the image in the transform domain while the phase remains unchanged[23]. The idea of cepstrums came about, as discussed previously, from Bogert's famous paper that introduced the concept along a whole set of glossary terms such as quefrency, saphe, and alanysis (for frequency, phase, and analysis, respectively). To understand where these terminologies came about, we need to take a look at the theoretical representation below[24]. The simple echo in a 1D signal, x(t), is represented as follows:

$$x(t) = s(t) + \alpha\, s(t - \tau)$$

Where α is the attenuation and τ is the delay. The Fourier spectral density (spectrum) of such a signal is represented as follows:

$$|X(f)|^2 = |S(f)|^2\,[1 + \alpha^2 + 2\alpha\cos(2\pi f\tau)]$$

Thus, the spectral density of the signal with an echo has the form of an envelope of the spectrum of the original signal and the spectrum of the contribution of the echo. By taking the logarithm of the spectrum, the product is converted to the sum of the two components as shown below:

$$C(f) = \log|X(f)|^2 = \log|S(f)|^2 + \log[1 + \alpha^2 + 2\alpha\cos(2\pi f\tau)]$$

Here, C(f) is a waveform that has an additive periodic component whose fundamental frequency is the echo delay τ. This new "spectral" representation is not in the frequency domain, nor is it in

[23] M. Azimi-Sadjadi "Digital Image Processing" Lectures 21 & 22, Colorado State University, 2017

[24] A. Oppenheim, et al. "From Frequency to Quefrency: a History of the Ceptrum" IEEE Signal Processing Magazine vol.21, no.5 pp. 95-106, 2004

the time domain; for this reason, Bogart chose to refer to this new domain as “Quefrency Domain.” He also termed the spectrum of the log “Cepstrum.”

A similar thread of research was performed by Oppenheim during his dissertation at MIT in the early 1960s, focused on homomorphic mapping between algebraic groups and vector spaces. He developed a theory for nonlinear signal processing referred to as homomorphic mapping systems. The essential idea of such systems is that many operations satisfy the same algebraic forms as additions; thus, homomorphic mapping between the signal spaces plays an essential role in linear combination in the general sense. The homomorphic mapping system comprises three cascading subsystems: the first is an invertible nonlinear operation that maps a nonadditive combination operation such as convolution into an ordinary addition. The second subsystem is a linear system obeying the additive superposition rules. The last subsystem is the inverse of the first nonlinear system. If we have two signals that are convolved, their Fourier transforms are multiplied and the complex logarithm will produce the sum of the two log Fourier transforms. After this, the inverse Fourier transform of a sum is the sum of the individual inverse transforms. Thus, this cascading operation transforms two convolved signals into the sum of the corresponding signals. Oppenheim’s homomorphic mapping in essence is the same concept described by Bogart’s “Cepstrum”.

Similarly, if the Cepstrum and the homomorphic mapping concepts are extrapolated into 2D applications, an echo in a 2D signal which is in essence the overlap between two images can be detected using the same three cascading operations discussed previously. Here, it is often referred to as *image registration*. The main reason that this method is not employed in 2D applications such as image registration is that not as simple as previously discussed in 1D application; that is because in 2D applications the echo signal can be translated, scaled, rotated or

any combination thereof which means the echo signal may be completely distorted beyond recognition from the original signal. For this reason, early attempts to apply cepstral analysis to images was only limited by translation and small rotations (<2 degrees)[25] [26]. Later papers proposed methods that extend these concepts to rotation and scaling; however, the authors demonstrated that it had overall inferior performance. The theory of each of the cases will be discussed in detail here. The general form of Cepstral in 2D ($\hat{f}(x,y)$) is expressed by the following formula, which is defined as the magnitude of the inverse Fourier of the logarithmic magnitude of the Fourier transform of the image.

$$\hat{f}(x,y) = |\mathcal{F}^{-1}\{\log|\mathcal{F}\{f(x,y)\}|\}|$$

Cepstral Analysis for Translation Estimation

If we let $f_1(x,y)$ and $f_2(x,y)$ represent two images, where f_2 is the translated version of f_1 displaced by an arbitrary vector $T\ (\tau_x, \tau_y)$ and attenuated by α, we get the following equation:

$$f_2(x,y) = \alpha\, f_1(x,y) + T\ (\tau_x, \tau_y)$$

If $h(x,y)$ is the sum of the two images f_1 and f_2, then the previous equation can be written as following:

$$h(x,y) = f_1(x,y) + f_2(x,y) = f_1(x,y) + \alpha\, f_1\big(x+\tau_x, y+\tau_y\big)$$

By taking the 2D Fourier Transform of both sides we get:

$$H(u,v) = \mathcal{F}_1(u,v) + \alpha\, \mathcal{F}_1(u,v)e^{-j(u\tau_x+v\tau_y)}$$

[25] R. Gonzalez "Robust Image Registration via Cepstral Analysis" International Conference on Digital Image Computing: Techniques and Applications, 2011, pp 40-50

[26] DJ Lee, et al. "Power Cepstrum and Spectrum Techniques Applied to Image Registration" Applied Optics, 1988, vol 27, pp.1099-1106

Factoring out $F_1(u,v)$ we get:

$$H(u,v) = \mathcal{F}_1(u,v)\left(1 + \alpha\, e^{-j(u\tau_x + v\tau_y)}\right)$$

By taking the log of both sides and simplifying using the Taylor series expansion of log(1+x), we obtain the following equality and series representation:

$$\log\big(H(u,v)\big) = \log|\mathcal{F}_1(u,v)| + \log\big(1 + \alpha\, e^{-j(u\tau_x + v\tau_y)}\big)$$

$$\log\big(H(u,v)\big) = \log|\mathcal{F}_1(u,v)| + \sum_{n=1}^{\infty} \frac{(-1)^n \alpha^n}{n}\, e^{-j(u\tau_x + v\tau_y)}$$

By taking the inverse Fourier Transform of both sides, we get the Cepstrum as following:

$$\hat{h}(x,y) = \widehat{f_1}(x,y) + \frac{(-1)^n \alpha^n}{n}\, \delta(x - \tau_x, y - \tau_y)$$

Where the location of the Dirac delta function is given by (τ_x, τ_y), which is the relative displacement between the two images.

Cepstral Analysis for Rotation and Scaling Estimation

If we let $f_1(x,y)$ and $f_2(x,y)$ represent two images where f_2 is the rotated version of f_1 by an arbitrary angle θ and scaled by a factor s, we get the following equation:

$$f_2(x,y) = f_1(s\,x\cos\theta + s\,y\sin\theta - x_0\,, s\,y\cos\theta - s\,x\sin\theta - y_0)$$

Taking the Fourier transform of both sides

$$\mathcal{F}_2(u,v) = \frac{1}{|s|}\, e^{-j(ux_0 + vy_0)} \mathcal{F}_1\left(\frac{u}{s}\cos\theta + \frac{v}{s}\sin\theta\,, \frac{u}{s}\cos\theta - \frac{v}{s}\sin\theta\right)$$

Taking the magnitude of both sides we get:

$$|\mathcal{F}_2(u,v)| = \frac{1}{|s|}\left|\mathcal{F}_1\left(\frac{u}{s}\cos\theta + \frac{v}{s}\sin\theta\,, \frac{u}{s}\cos\theta - \frac{v}{s}\sin\theta\right)\right|$$

Transforming the polar coordinate (u,v) on the right hand side to polar coordinate expressed in (r,φ) by substituting u = r cos φ and v = r sin φ, we obtain:

$$|\mathcal{F}_2(u,v)| = \frac{1}{|s|}\left|\mathcal{F}_1\left(\frac{r}{s}\cos\theta\cos\varphi + \frac{r}{s}\sin\theta\sin\varphi\ , \frac{r}{s}\sin\theta\cos\varphi + \frac{r}{s}\cos\theta\sin\varphi\right)\right|$$

By applying the product to sum trigonometric identity, we get:

$$|\mathcal{F}_2(u,v)| = \frac{1}{|s|}\left|\mathcal{F}_1\left(\frac{r}{s}\cos(\theta + \varphi), \frac{r}{s}\sin(\theta + \varphi)\right)\right|$$

By transforming the previous equation into polar coordinate system where $r = \sqrt{u^2 + v^2}$ and $\varphi = arctan\ arctan\ \frac{u}{v}$ results in the following equation:

$$|\mathcal{F}_2(r,\varphi)| = \frac{1}{|s|}\left|\mathcal{F}_1\left(\frac{r}{s}, \theta + \varphi\right)\right|$$

By taking the log of both sides we get:

$$\log|\mathcal{F}_2(r,\varphi)| = \log\frac{1}{|s|} + \log\left|\mathcal{F}_1\left(\frac{r}{s}, \theta + \varphi\right)\right| =$$

$$\log\frac{1}{|s|} + |\mathcal{F}_1(\log r - \log s\,, \theta + \varphi)|$$

The rotation and the scaling difference between $|\mathcal{F}_1|$ and $|\mathcal{F}_2|$ are recovered in the polar coordinate system using the cepstrum method as if it were translated in Cartesian space[22].

Results

The performance of cepstral analysis in image registration and disparity estimation was evaluated using several images incorporating only the translation displacement schema. The data was generated from Google Earth Studio using only trans-lateral trajectory, as Figure 74 shows. The camera was fixed at 65 degrees tilt angle and 225 degrees pan angle and constantly flying at altitude 179 feet between longitude/latitude -122.387/37.792 and -122.388/37.793.

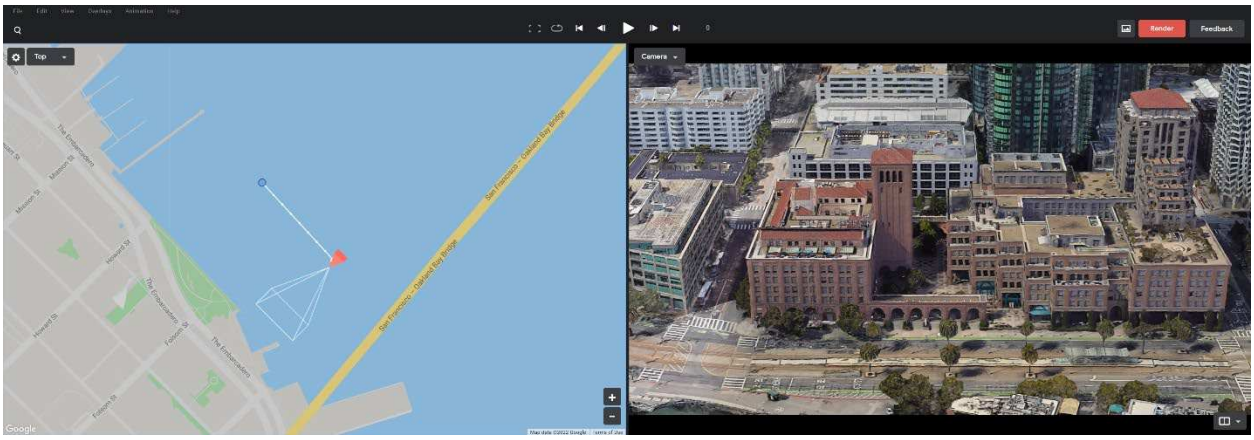


Figure 74 - Test Scenario Trajectory

To first test the disparity estimation using Cepstral analysis, a portion of the first frame image was concatenated with the whole first frame image as shown in Figure 75a below. The Cepstrum was calculated using the equations described in the previous section. The graph on Figure 75b shows two Dirac deltas. The Dirac deltas are at locations 1153 and 385, with a magnitude of 1 and 0.018, respectively. This means that the displacement of the echo signal is at pixel location 768 of the test image. This pixel location is calculated by subtracting the two location values of the Dirac deltas. The marker in Figure 75a shows the location at which the cepstral analysis indicated that the overlap captured.

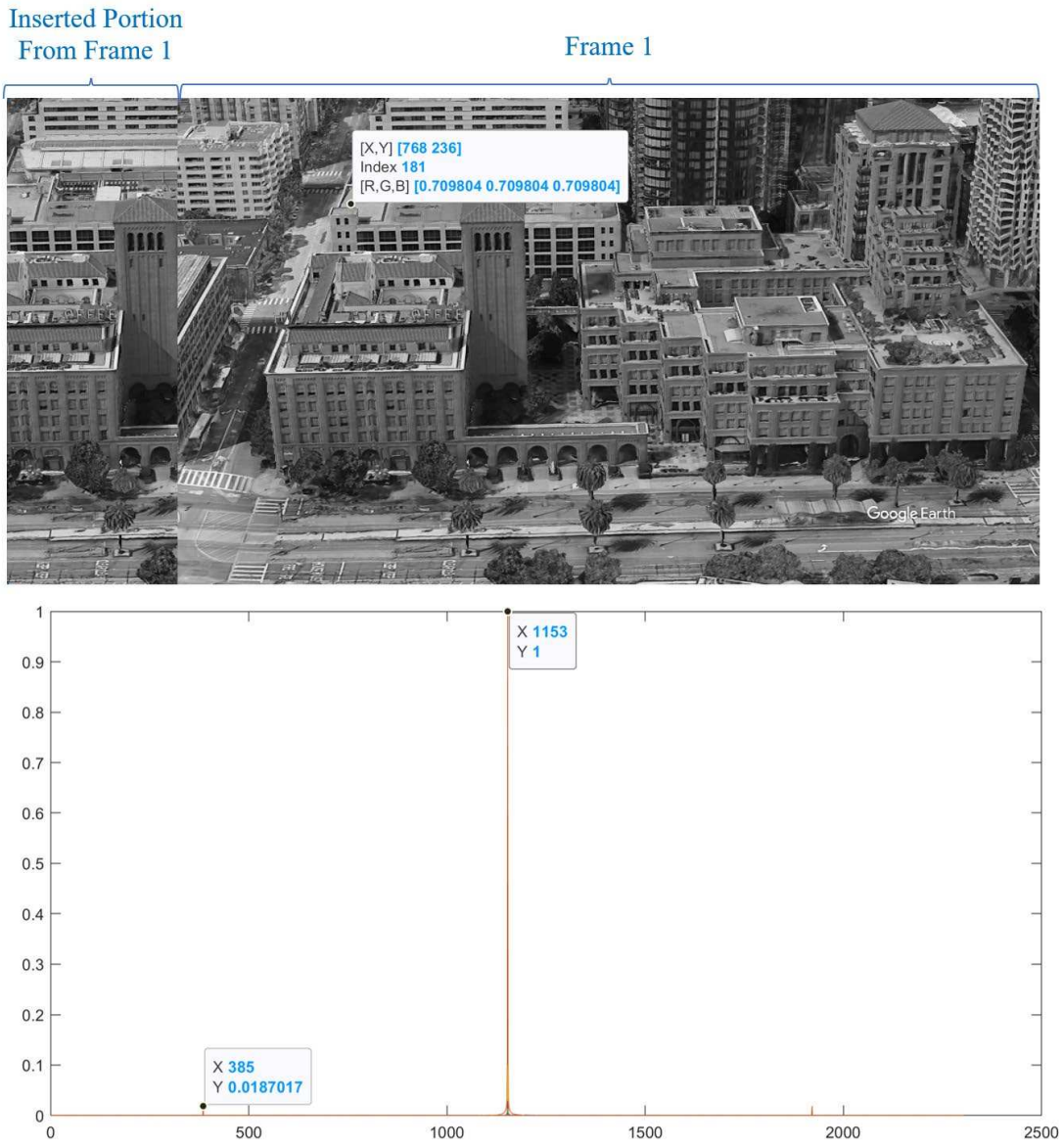


Figure 75 - (a) I1/I1 Concatenation, (b) I1/I1 Concatenation Cepstrum

The second test of disparity estimation concatenates the same slice from the first image used in the previous experiment, with the 5th frame image as shown in Figure 76a below. The cepstrum was calculated using the equations described in the previous section. The graph on Figure 76b shows two Dirac deltas. The Dirac deltas are at locations 1153 and 534, with a magnitude of 1 and 0.0013, respectively. This means the displacement of the echo signal is at pixel location 619 of the test image. This pixel location is calculated by subtracting the two

location values of the Dirac deltas. The marker in Figure 76a shows roughly the location at which the cepstral analysis indicated: this is indeed where the overlap was captured.

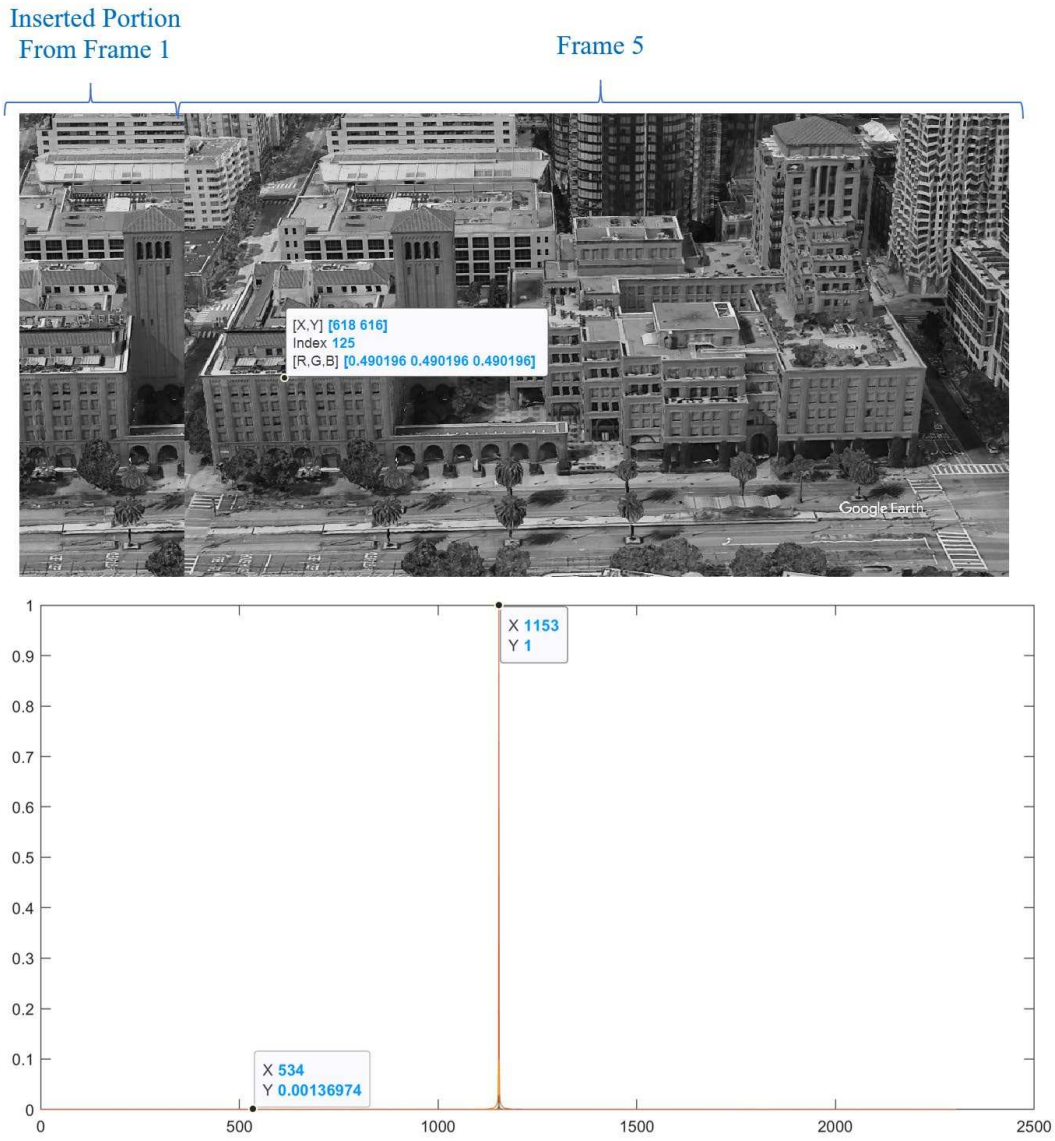


Figure 76 - (a) I1/I5 Concatenation, (b) I1/I5 Concatenation Cepstrum

The third test of disparity estimation concatenated the same slice from the first image that is used in the previous experiment with the 10th frame image as shown in Figure 77a below. The cepstrum was calculated using the equations described in the previous section. The graph on Figure 77b shows two Dirac deltas. The Dirac deltas are at locations 1153 and 683 with a

magnitude of 1 and 0.00103, respectively. This means the displacement of the echo signal is at pixel location 470 of the test image. This pixel location is calculated by subtracting the two location values of the Dirac deltas. The marker in Figure 77a shows the location at which the cepstral analysis indicated. This is indeed where the overlap was captured. The overlap was still visible despite the fact that the cepstral is becoming weaker as the closeup graph in Figure 77c shows.

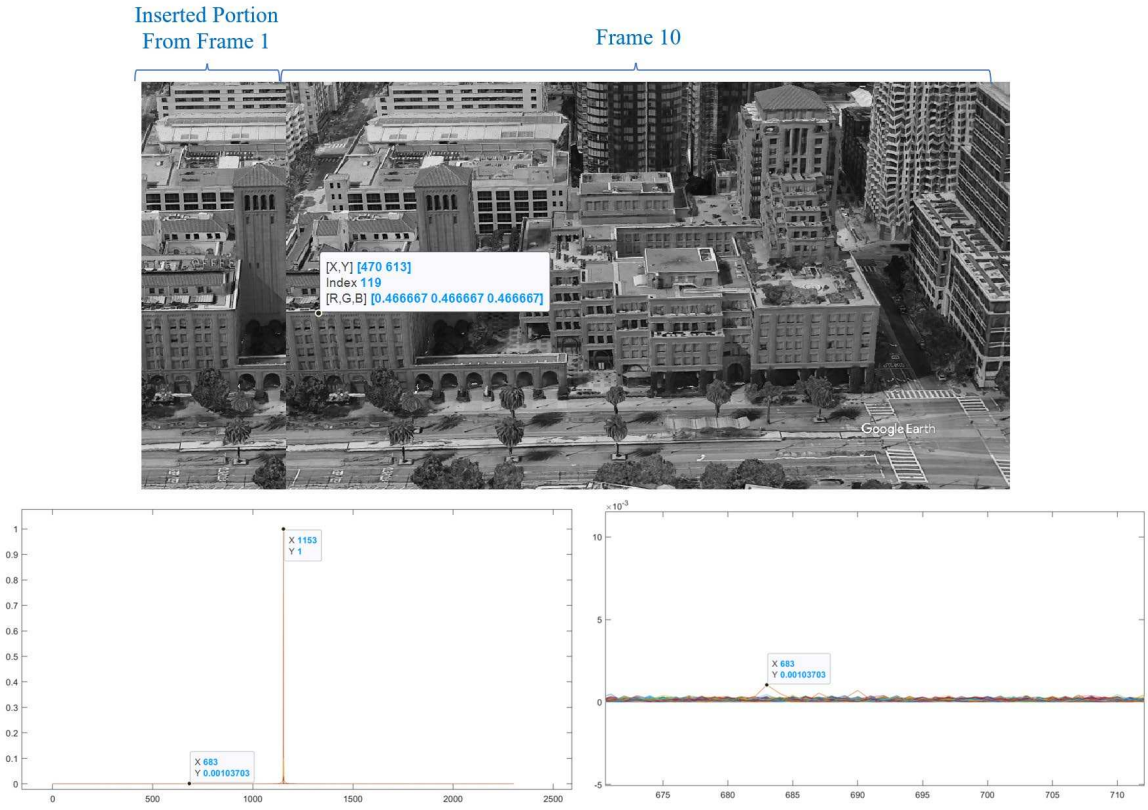


Figure 77 - (a) I1/I10 Concatenation, (b) I1/I10 Concatenation Cepstrum, (c) I1/I10 Ceptrum Closeup

The last test of disparity estimation that was performed concatenated the same slice from the first image that is used in the previous experiment with the 15th frame image, as shown in Figure 78a below. The Cepstrum was calculated using the equations described in the previous section. The graph on Figure 78b shows two Dirac deltas. The Dirac deltas are at locations 1153 and 714, with a magnitude of 1 and 0.00061, respectively. This means the displacement of the

echo signal is at pixel location 442 of the test image. This pixel location is calculated by subtracting the two location values of the Dirac deltas. The marker in Figure 78a shows the location at which the cepstral analysis indicated. This is indeed where the overlap was captured. The overlap was at this point harder to detect due to the fact the Dirac delta was obscured in the noise, as the closeup graph in Figure 78c shows. The reason for this is that, despite the camera movement being lateral, the projected information from the 3D captured scene has added information that was not previously available in the original perspective.

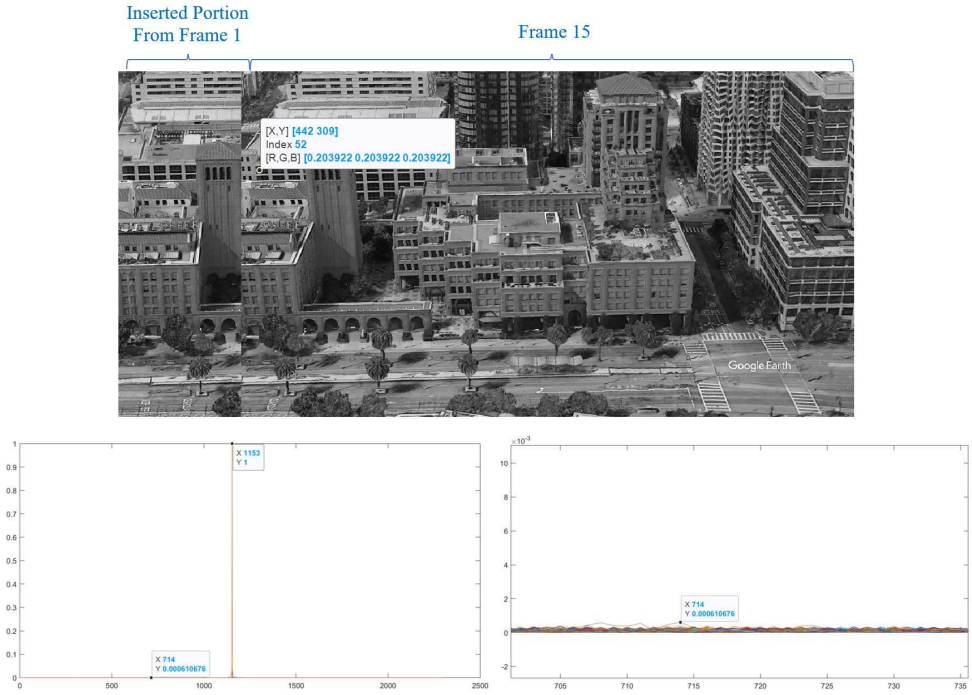


Figure 78 - a) I1/I15 Concatenation, (b) I1/I15 Concatenation Cepstrum, (c) I1/I15 Ceptrum Closeup

In summary, even though the cepstral worked well in the closeup scene and it is a relatively cheap function to perform, it is hard to auto detect the overlap between the images as they get further apart. Thus, a more robust method has to be developed to estimate image disparity to optimize the camera positions.

CHAPTER 6 – SUMMARY

This dissertation addressed a solution for the continuity aspect of tracking. The solution utilized Systems Engineering methodology throughout the design process. The design process started with the needs analysis, wherein the operational concept and the functional analysis were performed to fully estimate the feasibility of the project and how to meet the needs identified by the analysis. The next step in the design process detailed the concept through the performance of a trade study to address all the requirements, while leaving room for follow-up experiments. The next step in the design process identified the risks with the design and the implementation of the project. The risk analysis identified several areas of concerns for reasons such as drone integration, object detection algorithm, camera coordination, and camera position optimization. After the risks were identified, the concept was outlined and the requirements were hardened by performing the advanced development. The advanced development is also responsible for developing the concepts and the prototypes of the algorithms. Some of the more complex subsystems that posed higher risk were designed and prototyped in the subsequent chapters. The last sections of the first chapter conceptually detailed the software system, the engineering design, the integration and evaluation, the production and operation, and the support of the product. All these sections preliminarily detailed the operations, since the product is not going to be produced or fielded by this dissertation.

Chapter 2 discussed the first publication for this design. This chapter detailed the system block diagram, as well as each of the subsystem functional diagrams. The system is composed of the following subsystems: image processor; object detector; background subtractor; the tracking subsystem which is composed of an object tracker; a trajectory predictor; the feedback analyzer;

the camera mesh calibration system which is composed of a matching feature estimator and a relative pose estimator; and lastly the camera controller. The chapter discussed the three phases of implementation. The first proposed phase suggested a single camera that performed the first six functions described above. The second proposed phase suggested a multi-camera stationary system. This system used all the nine subsystems described above. The last proposed phase suggested a multi-drone mesh network, where the coordination of all the cameras have six degrees of freedom. The rest of the chapters addressed the high-risk subsystems for the multi-camera applications.

Chapter 3 addressed the object detector subsystem. It discussed several deep neural networks and benchmarked their performance using the Stanford Cars Dataset. This dataset comprises 16,185 images of 196 classes of cars. This dataset resembles realistic scenarios for our application since the images are taken in various lighting conditions, backgrounds, size, orientation and blur effects. The deep neural networks that were benchmarked are: VGG16, VGG19, DenseNet201, NASNetLarge, InceptionV3, Xception, InceptionResNetV2, MobileNetV2, ResNet50V2, ResNet152V2, ResNeXt50. The chapter discussed the architecture of each network and its performance. The network ranked as following: ResNeXt50, DenseNet201, Xception, NASNetLarge achieved above 92% accuracy; ResNet152V2, InceptionV3, ResNet50V2 achieved above 91% accuracy; and MobileNetV2 and InceptionResNetV2 achieved above 90%. The VGG16 and VGG19 approaches did not perform well with this dataset and I was not able to reproduce the results achieved by Benavides and Tae paper. The chapter introduced two collaborative approaches to enhance the performance. The first collaborative approach is the Equal Voting technique where all the participating networks had equal voting in selecting the proper class. The second collaborative approach is called

Selective or Weighted Voting where each network has a different voting weight based on its achieved performance during the training. Both approaches appreciably reduced the error (by approximately 10%).

Chapter 4 discussed the second publication for this dissertation. It addressed three methods to correlate a mesh (multi-camera) system. The first method had to have a priori knowledge of the camera locations and orientation. The second method correlated the cameras based on several non-orthogonal distant points such as a stellar constellation. The last method proposed a system that leverages the idea of stereo vision at a larger scale. This method relied on overlap between the images since it corresponds to the matching features between the two images before finding the fundamental matrix. This matrix is essential in estimating the pose between the two images to reconstruct the scene in three dimensions. This Chapter implemented the latter system, showing the results and the corrections that were made to optimize the results.

Chapter 4 used the feature overlap between the two cameras without addressing the optimized camera positions. Chapter 5 attempted to address the camera position optimization using Cepstral Analysis since it is a more rudimentary image processing function. The chapter discussed the history and the theory of Cepstral Analysis, and then showed the results achieved by calculating the cepstrum between frames of translation displacement simulated video. Even though the cepstral analysis worked well in the closeup scene and is a relatively cheap function to perform, it is hard to auto detect the overlap between the images as they get further apart. Thus, a more robust method has to be developed to estimate image disparity to optimize the camera positions.

This dissertation designed and implemented some of the complex subsystems. Several enhancements will be implemented during the post-doctoral research to complete additional

aspects of this research. The first is to develop a more robust, yet rudimentary, method to estimate the image disparity and optimize the camera position. Another necessary optimization is to implement the idea of an MPEG-2 “I” frame to change the frame of reference after N number of cameras to minimize the accumulated error of rotation and translation, while correlating the images. Another area that needs to be prototyped is the trajectory predictor, which is essential to predict the next move of the object of interest. The last area that needs to be researched further is the integration of this system on 6-degrees of freedom drones so it can track the desired object of interest throughout space.

BIBLIOGRAPHY

[1] A. Kossiakoff, Systems Engineering Principal and Practice, Wiley, 2011.

[2] L. Maddalena and A. Petrosino, "A Self-Organizing Approach to Background Subtraction for Visual Surveillance Applications," *IEEE Transactions on Image Processing,* pp. Vol. 17, no. 7, pp. 1168-1177 , 2008.

[3] H. Williams and S. Simske, "Object Tracking Continuity through Track and Trace Method," in *Electronic Imaging, Autonomous Vehicles and Machines*, San Francisco, CA, USA, 2020.

[4] S. Saha, 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[5] J. Krause, 2013. [Online]. Available: http://ai.stanford.edu/~jkrause/cars/car_dataset.html.

[6] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks For Large-Scale Image Recognition," in *ICLR*, 2015.

[7] D. S. Bolme, J. R. Beveridge, B. A. Draper and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *IEEE Computer Society Conference On Computer Vision and Pattern Recognition*, San Francisco, CA, USA, 2010.

[8] G. Huang, Z. Liu, L. Van Der Maaten and K. Weinberger, "Densely Connected Convolutional Networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[9] J. Krause, M. Stark, J. Deng and L. Fei-Fei, "3D Object Representations for Fine-Grained Categorization," in *IEEE International Conference on Computer Vision (ICCV)*, Sydney, Australia, 2013.

[10] L. N. Smith, "Cyclical Learning Rates for Training Neural Networks," in *IEEE Winter Conference On Applications of Computer Vision (WACV)*, Santa Rosa, CA, USA, 2017.

[11] B. Zoph, V. Vasudevan, J. Shlens and Q. V. Le, "Learning Transferable Architectures for Scalable Image Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[13] F. Chollet, "Deep Learning with Depthwise Separable Convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[14] C. Szegedy, S. Ioffe, V. Vanhoucke and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," in *AAAI Publications, Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[15] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[16] S. Xie, R. Girshick, P. Dollar, Z. Tu and K. He, "Aggregated Residual Transformation for Deep Neural Networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[17] S. J. Simske, Meta-Algorithmics Patterns for Robust, Low Cost, High Quality Systems, John Wiley & Sons, 2013.

[18] V. Nastro and U. Tancredi, "Great Circle Navigation with Vectorial Methods," *The Journal of Navigation,* pp. Vol. 63, Issue 3, pp. 557-563, 2010.

[19] Q.-T. Luong and O. Faugeras, "The fundamental matrix: Theory, algorithms, and stability analysis.," *International Journal of Computer Vision,* pp. Vol. 17, pp. 43-75, 1996.

[20] A. Trabelsi, M. Chaabane, N. Blanchard and R. Beveridge, "Pose Proposal and Refinement Network for Better 6D Object Pose Estimation," in *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021.

[21] H. Williams, S. Simske and F. G. Bishay, "Unify The View of Camera Mesh Network to a Common Coordinate System," in *Electronic Imaging, Autonomous Vehichles and Machines*, 2021.

[22] B. Bogert, "The Quefrency Alanysis of Time Series for Echoes: Cepstrum, Pseudo Autocovariance, Cross-Cepstrum and Saphe Cracking," *Time Series Analysis,* pp. 209-243, 1963.

[23] A. Oppenheim and R. Schafer, "From Frequency to Quefrency: a History of the Ceptrum," *IEEE Signal Processing Magazine,* pp. Vol. 21, Issue 5, pp. 95-106, 2004.

[24] M. Azimi-Sadjadi, Digital Image Processing Lectures 21-22, Colorado State University, 2017.

[25] R. Gonzalez, "Robust Image Registration via Cepstral Analysis," in *IEEE International Conference on Digital Image Computing: Techniques and Applications*, Noosa, Australia, 2011.

[26] D.-J. Lee, T. Krile and S. Mitra, "Power Cepstrum and Spectrum Techniques Applied to Image Registration," *Optical Society of America,* pp. Vol. 27, Issue 6, pp.1099-1106, 1988.

[27] C. Veness, 2020. [Online]. Available: https://www.movable-type.co.uk/scripts/latlong.html.

[28] S.-C. Cheung and C. Kamath, "Robust techniques for background subtraction in urban traffic video," in *SPIE Visual Communications and Image Processing*, 2004.

[29] L. Maddalena and A. Petrosino, "A Self-organizing Approach to Detection of Moving Patterns for Real-Time Applications," in *International Symposium on Brain, Vision, and Artificial Intelligence (BVAI)*, Berlin, Heidelberg, 2007.

[30] P. Torr and D. Murray, "The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix," in *International Journal of Computer Vision*, 1997.

[31] C. Park, J.-E. Lee and K.-H. Bae, "Corresponding Metadata-Based Stereo Object Tracking System Using Disparity-Motion Estimation," *Journal of Image Science and Technology,* pp. Vol 53, Issue: 1, pp.10502-1-10502-6, 2009.

[32] E. Reyes-Santos, H. Jimenez-Hernandez, L. Barriga-Rodriguez, J. Soto-Cajiga, H. H and F. Carrizo-Corral, "Tracking and Estimating Tridimensional Position Through Camera-PT Array," *Electronic Imaging, Video Surveillance and Transport Imaging Applications,* pp. 1-7, 2016.

[33] J. Li, H. Aghajan, J. R. Casar and W. Philips, "Camera Pose Estimation by Vision-inertial Sensor Fusion: An Application to Augmented Reality Books," *Electronic Imaging, The Engineering Reality of Virtual Reality,* pp. 1-6, 2016.

[34] J. Li-Chee-Ming and C. Armenakis, "UAV navigation system using line-based sensor pose estimation," *Geo-spatial Information Science,* pp. Vol.21, Issue 1, pp. 2-11, 2018.

[35] Z. Zhou, D. Yin, J. Ding, Y. Luo, M. Yuan and C. Zhu, "Collaborative Tracking Method in Multi-Camera System," *Journal of Shanghai Jiaotong University (Science),* pp. Vol.25, pp.810-810, 2020.

[36] D. H. Ye, J. Li, Q. Chen, J. Wachs and C. Bouman, "Deep Learning for Moving Object Detection and Tracking from a Single Camera in Unmanned Aerial Vehicles (UAVs)," *Electronic Imaging, Imaging and Multimedia Analytics in a Web and Mobile World,* pp. 466-1-466-6, 2018.

[37] E. Unlu, E. Zenou, N. Riviere and P.-E. Dupouy, "An autonomous drone surveillance and tracking architecture," in *Electronic Imaging, Autonomous Vehicles and Machines Conference*, 2019.

[38] S. Campbell, N. O'Mahony, L. Krpalcova, D. Riordan, J. Walsh, A. Murphy and C. Ryan, "Sensor Technology in Autonomous Vehicles : A review," in *IEEE 2018 29th Irish Signals and Systems Conference (ISSC)*, Belfast, UK, 2018.

[39] M. Walters, "Sensor Techologies for Autonomous Vehicles," in *Electronic Imaging, Autonomous Vehicles and Machines*, 2020.

[40] Y.-J. Chang and Y.-S. Ho, "Disparity Estimation Using Fast Motion-Search Algorithm and Local Image Characteristics," in *Electronic Imaging, Image Processing: Algorithms and Systems XVI*, 2018.

[41] J. W. Davis and A. Bobick, "The Representation and Recognition of Human Movement Using Temporal Templates," *IEEE,* pp. 928-934, 1997.

[42] E. P. Ijjina, "Human Fall Detection in Depth-Videos Using Temporal Templates and Convolutional Neural Networks.," *Learning and Analytics in Intelligent Systems,* pp. 217-236, 2022.

[43] J.-H. Mun and Y.-S. Ho, "Guided Image Filtering based Disparity Range Control in Stereo Vision," *Electronic Imaging, Stereoscopic Displays and Applications XXVIII,* pp. 130-136, 2017.

[44] N. Benavides and C. Tae, "Fine-Grained Image Classification for Vehicle Makes & Models using Convolutional Neural Networks," CS230 Standford.

[45] T. G. Dietterich, "Ensemble Methods in Machine Learning. In: Multiple Classifier Systems.," *MCS 2000. Lecture Notes in Computer Science,* pp. vol 1857, pp. 1-15, 2000.

[46] S.-H. Seo and M. R. Azimi-Sadjadi, "A 2-D Filtering Scheme for Stereo Image Compression Using Sequential Orthogonal Subspace Updating," *IEEE Transactions on Circuits And Systems For Video Technology,* vol. 11, no. 1, pp. 52-66, 2001.

[47] S.-W. Seo, M. Azimi-Sadjadi and B. Tian, "A least-squares-based 2-D filtering for disparity estimation," in *Proceedings of International Conference on Image Processing*, Santa Barbara, CA, USA, 1997.

APPENDIX 1

Python Version List

Package	Version
python 3.9	Python 3.9.7
Jinja2	3.0.1
Keras-Applications	1.0.8
Keras-Preprocessing	1.1.2
Markdown	3.3.4
MarkupSafe	2.0.1
Pillow	8.3.2
PyWavelets	1.1.1
PyYAML	5.4.1
Pygments	2.10.0
Shapely	1.7.1
Werkzeug	2.0.1
absl-py	0.13.0
adam	0.0.0.dev0
albumentations	1.0.3
astunparse	1.6.3
backcall	0.2.0
bokeh	2.4.0
cachetools	4.2.2
certifi	2021.5.30
charset-normalizer	2.0.5
clang	5
colorama	0.4.4
console-progressbar	1.1.2
cycler	0.10.0
daytime	0.4
decorator	5.1.0
easydict	1.9
efficientnet	1.0.0
flatbuffers	1.12
gast	0.4.0
google-auth	1.35.0
google-auth-oauthlib	0.4.6
google-pasta	0.2.0

graphviz	0.19.1
grpcio	1.40.0
h5py	3.1.0
idna	3.2
image-classifiers	1.0.0
imageio	2.9.0
imgaug	0.4.0
ipython	7.27.0
jedi	0.18.0
joblib	1.1.0
keras	2.6.0
kiwisolver	1.3.2
livelossplot	0.5.4
matplotlib	3.4.3
matplotlib-inline	0.1.3
networkx	2.6.3
numpy	1.19.5
oauthlib	3.1.1
opencv-python	4.5.3.56
opencv-python-headless	4.5.3.56
opt-einsum	3.3.0
packaging	21
pandas	1.3.3
parso	0.8.2
pickleshare	0.7.5
pip	21.3.1
prompt-toolkit	3.0.20
protobuf	3.18.0
pyasn1	0.4.8
pyasn1-modules	0.2.8
pydot	1.4.2
pydot2	1.0.33
pydot3	1.0.9
pyparsing	2.4.7
python-dateutil	2.8.2
pytz	2021.1
requests	2.26.0
requests-oauthlib	1.3.0
rsa	4.7.2
scikit-image	0.18.3
scikit-learn	1
scipy	1.7.1

seaborn	0.11.2
segmentation-models	1.0.1
sequence	0.3.4
setuptools	57.0.0
six	1.15.0
sklearn	0
tensorboard	2.6.0
tensorboard-data-server	0.6.1
tensorboard-plugin-wit	1.8.0
tensorflow	2.6.0
tensorflow-estimator	2.6.0
termcolor	1.1.0
threadpoolctl	3.0.0
tifffile	2021.8.30
torch	1.9.1
torchvision	0.10.1
tornado	6.1
tqdm	4.62.2
traitlets	5.1.0
typing-extensions	3.10.0.2
urllib3	1.26.6
wcwidth	0.2.5
wheel	0.36.2
wrapt	1.12.1