3-16-2007

Development of an Experimental Platform for Testing Autonomous UAV Guidance and Control Algorithms

Justin R. Rufa

DEVELOPMENT OF AN EXPERIMENTAL PLATFORM FOR TESTING AUTONOMOUS UAV GUIDANCE AND CONTROL ALGORITHMS

THESIS

Justin R. Rufa, Captain, USAF

AFIT/GAE/ENY/07-M20

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GAE/ENY/07-M20

DEVELOPMENT OF AN EXPERIMENTAL PLATFORM FOR TESTING AUTONOMOUS UAV GUIDANCE AND CONTROL ALGORITHMS

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Aeronautical Engineering

Justin R. Rufa, BSAE

Captain, USAF

March 2007

AFIT/GAE/ENY/07-M20

DEVELOPMENT OF AN EXPERIMENTAL PLATFORM FOR TESTING AUTONOMOUS UAV GUIDANCE AND CONTROL ALGORITHMS

Justin R. Rufa, BSAE
Captain, USAF

Approved:

/SIGNED/	15 March 2007
Maj Paul A. Blue (Thesis Advisor)	Date
/SIGNED/	16 March 2007
Dr. David R. Jacques (Committee Member)	Date
/SIGNED/	15 March 2007
Dr. Meir Pachter (Committee Member)	Date

AFIT/GAE/ENY/07-M20

Abstract

With the United States' push towards using unmanned aerial vehicles (UAVs) for more military missions, wide area search theory is being researched to determine the viability of multiple vehicle autonomous searches over the battle area. Previous work includes theoretical development of detection and attack probabilities while taking into account known enemy presence within the search environment. Simulations have been able to transform these theories into code to predict the UAV performance against known numbers of true and false targets. The next step to transitioning these autonomous search algorithms to an operational environment is the experimental testing of these theories through the use of surrogate vehicles, to determine if the guidance and control laws developed can guide the vehicles when operating in search areas with true and false targets. In addition to the challenge of experimental implementation, dynamic scaling must also be considered so that these smaller surrogate vehicles will scale to full size UAVs performing searches in real world scenarios.

This research demonstrates the ability of a given sensor to use a basic ATR algorithm to identify targets in a search area based on its size and color. With this ability, the system's target thresholds can also be altered to mimic real world UAV sensor performance. It also builds on previous dynamic scaling studies to show that the performance of a full size UAV can be imitated using a surrogate vehicle. Further investigation will show sensor orientation, field of view, vehicle geometry, and the known size of the target can be used to determine target pixel thresholds as well as the vehicle steering correction angle to navigate directly over the centroid of an identified target.

Acknowledgements

I would like to take this opportunity to express my sincere thanks to so many people who worked so hard to help complete this thesis as well as guide me through my 18 month journey through AFIT. First and foremost, I would like to thank Major Paul Blue, my thesis advisor, for the countless hours he spent giving me words of wisdom as well as assuring me that my work will make a difference. His dedication and hard work ensured that that the Air Force will benefit from the knowledge imparted in several AFIT 07M students during their time under his instruction. I would also like to thank Dr. David Jacques for his timely thesis heading checks and Dr. Meir Pachter for being one of my thesis readers.

Huge thanks also go out to Tim Vincent and Don Smith in the AFIT Advanced Navigation Technology Center's Laboratory. Their tireless work to help me build all of the hardware necessary to complete my thesis and solve those countless seemingly impossible integrations issues made this research possible.

I also want to thank my wife for her patience during those long hours while I was spending too much time at AFIT or in our basement completing those never ending homework assignments in the classes we all enjoyed so much. Her support made it possible for me to succeed at AFIT while knowing that my home life was as good as ever.

Justin Robert Rufa

Table of Contents

Page

Page

List of Figures

Page

List of Tables

Page

List of Symbols

A_{objpix}	$\equiv$	Area of Object in Pixels
F_S	$\equiv$	Frame Separation
L_{obj}	$\equiv$	Object Characteristic Length
L_{targ}	$\equiv$	Target Characteristic Length
L_{Rx}	$\equiv$	Object's Lower Right Pixel x coordinate
L_{Ry}	$\equiv$	Object's Lower Right Pixel y coordinate
O_L	$\equiv$	Frame Overlap
P_{FTR}	$\equiv$	Probability of False Target Report
P_{TR}	$\equiv$	Probability of Target Report
P_{BHoriz}	$\equiv$	Object's Rear Number of Horizontal Pixels
$P_{ObjHoriz}$	$\equiv$	Object's Front Number of Horizontal Pixels
$P_{ObjVert}$	$\equiv$	Object's Number of Vertical Pixels
U_{Lx}	$\equiv$	Object's Upper Left Pixel x coordinate
U_{Ly}	$\equiv$	Object's Upper Left Pixel y coordinate
V	$\equiv$	Vehicle/Sensor Velocity
$X_{centroid}$	$\equiv$	Object's Centroid x coordinate
$Y_{centroid}$	$\equiv$	Object's Centroid y coordinate
Y_{objin}	$\equiv$	Object's Centroid Vertical Distance from Rear of Frame in Inches
Y_{objpix}	$\equiv$	Object's Centroid Vertical Distance from Rear of Frame in Pixels
a	$\equiv$	Object Horizontal Length
b	$\equiv$	Object Vertical Length
c	$\equiv$	ROC Parameter
d	$\equiv$	Sensor Dead Band

h	≡	Sensor Height (Altitude) above Target
r	≡	Vehicle Minimum Turn Radius
w_b	≡	Frame Rear Width
w_f	≡	Frame Front Width
s_b	≡	Frame Front Vertical Slant Range
s_{obj}	≡	Object Front Vertical Slant Range
s_f	≡	Frame Rear Vertical Slant Range
z	≡	Frame Length
α	≡	Slant Angle
$\beta_{centroid}$	≡	Target Centroid Angle
β_{obj}	≡	Object Vertical Angle
γ	≡	Depression Angle
θ	≡	Sensor Swath Angle
ρ_t	≡	Frame Pixel Density
ρ_{Yobj}	≡	Horizontal Pixel Density at Object's Centroid
φ	≡	Sensor Bore Angle
χ_b	≡	Object Rear Horizontal Angle
χ_f	≡	Object Front Horizontal Angle
ψ	≡	Object's Horizontal Centroid Angle

List of Abbreviations

ATR Autonomous Target Recognition

FTAR False Target Attack Rate

GUI Graphical User Interface

MAV Micro Unmanned Aerial Vehicle

OEM Original Equipment Manufacturer

ROC Receiver Operator Characteristic

SDK Software Development Kit

UAV Unmanned Aerial Vehicle

VFOV Vertical Field of View

DEVELOPMENT OF AN EXPERIMENTAL PLATFORM FOR TESTING AUTONOMOUS UAV GUIDANCE AND CONTROL ALGORITHMS

1. Introduction

1.1 Motivation for Autonomous Cooperative Control of UAVs

1.1.1 Current Search and Destroy Mission

Since the end of the Cold War, the United States has found itself locked in urban warfare and completing military missions other than war at a faster pace than ever before. As a result, tactics once used in the open battlefield are no longer considered viable when fighting against enemies without uniforms in large, mostly civilian, urban settings. One current technology push to give the U.S. Armed Forces an advantage over their enemies in this type of environment is the development of autonomous unmanned aerial vehicles (UAV) and autonomous unmanned micro aerial vehicles (MAV). To best allocate these invaluable resources in a battlefield setting, cooperative control of multiple UAVs & MAVs is being explored at the Air Force Institute of Technology. Some benefits of using cooperative UAV fleets include search redundancy, capability to search larger areas quicker, multiple targets can be simultaneously tracked, and operators can be kept out of the extreme danger of some of today's urban war zones. Also, as suggested by three researchers at Colorado State University (Richards, Whitley, and Beveridge, 2005), if the

UAV used for a particular mission is prone to failure, it might be cheaper to use multiple inexpensive UAVs instead of one costly search system.

As mentioned above, the current enemies of the United States and its allies do not follow established rules of war, and thus it is possible for almost any vehicle, building, or person on the ground in a region of conflict to be a target. When terrorists use hospitals or mosques as their hideouts or hide behind women and children, the line between civilian infrastructure and legitimate targets, according to the rules of war, becomes murky. To ensure collateral damage is minimized in this type of situation, UAVs must be able to discern the actual targets from those entities that at first glance appear to be a target, but are actually part of the civilian infrastructure being used illegally. It is this point that makes the cooperative control aspect of UAV target searching critical to ensure that a UAV has found a legitimate military target before it attempts to destroy it. As the U.S. continues to fight in urban environments around the world, the need for this technology will keep growing and the tolerance for error on the battlefield and in the political arena will keep shrinking.

1.1.2 Full Scale Autonomous UAV Experimental Work

Even though this autonomous and cooperative technology is being heavily researched and funded by the US Department of Defense, the UK Ministry of Defence is also working to develop the same type of technology. As recently as 30 October 2006, Qinetiq, a UK defence contractor, completed an in flight demonstration of the UAV Command and Control Interface (UAVCCI) by using a BAC 1-11 1960's era jetliner to simulate a fighter pilot managing four UAVs as well as their own jet. To add realism to

the test and prove the functionality of the UAVCCI, the pilot in control of the BAC 1-11 sat in the back of jet where he controlled it as well as the UAVs.
The UAVCCI system is designed to allow for semiautonomous flight of the UAVs so pilots can easily control their jet, without worrying about always giving commands to the UAVs. When the UAVs do not get commands, they are programmed to fly straight and level, but the pilot has the ability to direct them through a moving map and push buttons. With these controls, the pilot can direct the UAVs to loiter, start a search, or attack. This test showed that cooperative and autonomous control of UAVs can occur not only from a ground station, but also from the cockpit of a military jet closer to the fight. The pilot would then be able to use the displays as well as the real time battlefield environment to give the UAVs specific commands (Marks, 2006). As previously noted, the remote or autonomous control of military assets will help greatly in the Global War on Terrorism to keep US and allied service members farther from their nameless and uniformless enemies and their treacherous improvised explosive devices (IEDs). According to Icasualties.org, a non military website that provides DoD verified information on Operation Iraqi Freedom casualties, 1183 of the 3085 U.S. deaths through the end of January 2007 (roughly 38 percent) have been caused by IEDs (iCasualties.org, 2007). Development of autonomous search vehicles will help mitigate the effects of this deadly tactic in the future. In fact, the research in this thesis will help the Pentagon towards their goal of having one third of their military assets "robotic or remotely controllable by 2015 (Marks 2006)."

While the physical integration of hardware and software of sensors into an unmanned vehicle can be quite complex, the operational concept of the system is quite straightforward. The system can be thought to be analogous to a self checkout area at a grocery or retail store. With the self checkout process one operator monitors multiple checkout stations and only intervenes if the customer at the station is having problems that they cannot solve themselves. In the autonomous UAV search group concept one operator will have the capability to monitor multiple UAVs to ensure that the group is working towards its mission objectives, and only intervenes if there is a problem that one or more of the UAVs cannot fix on their own.

1.1.3 Autonomous UAV Cost /Benefit Analysis

Many benefits come from operating UAVs in the autonomous regime. The simplest advantage comes from the ability to allocate less personnel to operate more UAVs. When UAVs are flown manually by an operator, there is at least one human for each UAV and often several. If one operator can monitor 3-4 UAVs, then more UAVs can be utilized with the same number of operators. This operator can also perform this job from any ground station within communications range (radio, satellite, etc) of the UAV fleet they are controlling, thus keeping them off of the battlefield. Other advantages include being able to perform coordinated searches over larger areas than a single UAV could search, and engaging multiple targets with multiple vehicles in the same search.

Some challenges involved in fielding networked UAV systems include the development of adaptable operational procedures, as well as planning and deconfliction of assets. As these technologies progress, UAVs will be able to make better allocation and targeting decisions on their own. However, autonomous UAVs will always have the chance to make poor decisions because they are taking data acquired through real time sensing and computing solutions based on human produced algorithms to make targeting decisions that could result in a bad target selection as well as damage to or outright loss of the air vehicle (Vachtsevanos, 2004). While some of these algorithms will possibly involve multiple checks from other UAVs in the fleet before engaging targets, they will never be foolproof instructions to ensure a wrong target is never hit. Because these algorithms operate independent of human control, they must continually be updated, refined, double checked, and monitored to keep up with the ever changing conditions on the battlefields of the world.

1.2 Previous Applicable Research

The current state of the art in Unmanned Aerial Vehicle (UAV) targeting research at the Air Force Institute of Technology (AFIT) has implemented analytical concepts into robust multi-warhead and multi-vehicle Matlab/Simulink simulations. Since many AFIT theses as well as a multiple dissertations have explored the autonomous UAV targeting concepts and simulations, the next logical step in the process is to develop hardware to prove it is possible for autonomous target recognition (ATR) systems to properly detect and identify objects. This experimental validation of theoretical concepts will help the Air Force move towards implementing robust targeting algorithms into operational autonomous UAV fleets in the future.

Some of the topics of the wide area search research involve optimal path planning, applying probability theory to the UAV fleet, conducting simulations using the Multi-UAV simulation test bed (Rasmussen, Mitchell, Chandler, 2005), automatic target recognition (ATR), performance under limited communication, non-linear control of UAVs in close coupled formation, and most recently dynamic scaling of UAVs. Each topic contributes greatly to cooperative control of autonomous UAVs, but only ATR and dynamic scaling will be expounded in the present research. ATR theory will be used in the development of a simple target identification algorithm that a ground based search vehicle platform will use to identify targets and dynamic scaling will be used to ensure that the vehicle has the proper dynamics to reasonably represent a flyable experimental UAV system.

1.2.1 Autonomous Target Recognition

To better understand the logic behind cooperative UAV targeting algorithms, the concept of a confusion matrix must first be introduced. It has been used in the work of Dr. David Jacques and Dr. Meir Pachter (2003) to provide conditional probabilities for each possible outcome when a search vehicle sweeps a given area and encounters an object it determines is not part of the background. For simplicity, the concept will be explained below using a single target scenario.

For a UAV to detect a single type of target during a wide area search, two events must occur. The first event is the proper characterization of the target. This can occur, with operator involvement, during the search or this information can be preloaded into the UAV's ATR algorithm. Targets are characterized by size, shape, color, another unique signature (e.g. IR), location in relation to other objects, or a combination of these attributes depending on the type of onboard sensor(s) and their capabilities. Like with any search, the sensor must know what it is searching for or it will not know when it has found a target. Once the target is properly characterized, the second event is the actual detection of the target by the UAV's ATR system. The ATR system includes both the sensor(s) used to obtain signature information about objects and the ATR algorithms used to detect and classify/identify objects based on the sensor data. Since no ATR system is perfect there are times when it might misidentify objects it encounters. Table 1 shows the four possible outcomes of this type of search when an object is encountered.

Table 1. Simple Binary Confusion Matrix

	Object Encountered	
Object Declared	True	False
True	P_{TR}	$1-P_{FTR}$
False	$1-P_{TR}$	P_{FTR}

When the ATR algorithm processes the sensor data at a given instant it will either classify the object as a target or a false target (perhaps a decoy or just background noise). Note that in the simple binary case, a false target classification occurs when either an object in the sensor footprint is not classified as a target or if there is no object in the sensor footprint. If the object is a target, the percent of the time the sensor properly identifies it as such is the probability of true target report, P_{TR} in the confusion matrix. If that object is a target, the percent of time the sensor incorrectly dismisses it as a false target is $1- P_{TR}$. Alternatively, if the object is a false target object or just clutter, the percent of the time it is properly identified as such is the probability of false target report, P_{FTR}. The final piece of the confusion matrix is $1- P_{FTR}$, the percent of the time the sensor encounters an object that is not a target, but identifies it as a target.

To account for all possible outcomes given a target or false target encounter, the conditional probabilities of each column will add up to one because the ATR algorithm is forced to state that its field of view either contains a target or does not contain a target. Expanding this concept to the multiple target case is as straightforward as expanding the dimensions of the matrix to make it an *m* x *n* rectangle where *m*-1 is equal to the number of possible specific target declarations with the final declaration being an "Other" or "None of the Above" and *n* is equal to the number of possible object types that can be encountered in the search area.

Table 2. Multiple Target Confusion Matrix

	Object Encountered			
Object Declared	Object 1	Object 2	Object 3	Object $_n$
Target Class 1	$P_{TR1\|1}$	$P_{TR1\|2}$	$P_{TR1\|3}$	$P_{TR1\|n}$
Target Class 2	$P_{TR2\|1}$	$P_{TR2\|2}$	$P_{TR2\|3}$	$P_{TR2\|n}$
Target Class $_{m-1}$	$P_{TRm-1\|1}$	$P_{TRm-1\|2}$	$P_{TRm-1\|3}$	$P_{TRm-1\|n}$
Other	$1-\Sigma PTR_{j\|1}$	$1-\Sigma PTR_{j\|2}$	$1-\Sigma PTR_{j\|3}$	$1-\Sigma PTR_{j\|n}$

In the binary confusion matrix, the ideal case would be to have an identity matrix where $P_{TR} = 1$ and $P_{FTR} = 1$. With these values, the system would always attack targets and never attack false targets. Since the real world does not allow for this, the best case is to strike a balance between the competing objectives of P_{TR} and P_{FTR}.

To better understand how the probability of a false target being declared a true target, 1- P_{FTR}, relates to system performance, the false target encounter rate, η_f must also be considered. This parameter is multiplied by $1-P_{FTR}$ to determine the false target attack rate or FTAR. The two metrics, FTAR and P_{TR} were used by Gillen (2001) in a previous AFIT thesis as a measure of success for ATR search algorithms. From a logical standpoint, having a high FTAR not only shows that the sensor is not properly characterized, but in reality it equates to civilian or other nonmilitary objects being accidentally targeted, or wasted munitions if the targeted object is of no military value. Having a low P_{TR} is just as dangerous because it could result in missed targets that will cause later harm because they were not destroyed. Making the tradeoff between the two so that P_{TR} is high enough to be mission effective and FTAR is low enough to be acceptable becomes a non trivial problem that is dependent on both the quality of the sensor and also the ATR algorithm written to make the crucial targeting decisions

A tool used by Kish (2005) to visualize the relationship between P_{TR} and 1-P_{FTR} is called the Receiver Operating Characteristic (ROC) curve. This curve traditionally shows 1-P_{FTR} on the x-axis and P_{TR} on the y axis and is plotted for multiple values of *c* (ROC parameter). The ROC parameter defines a performance envelope for the sensor/ATR, with a higher *c* value providing better performance.

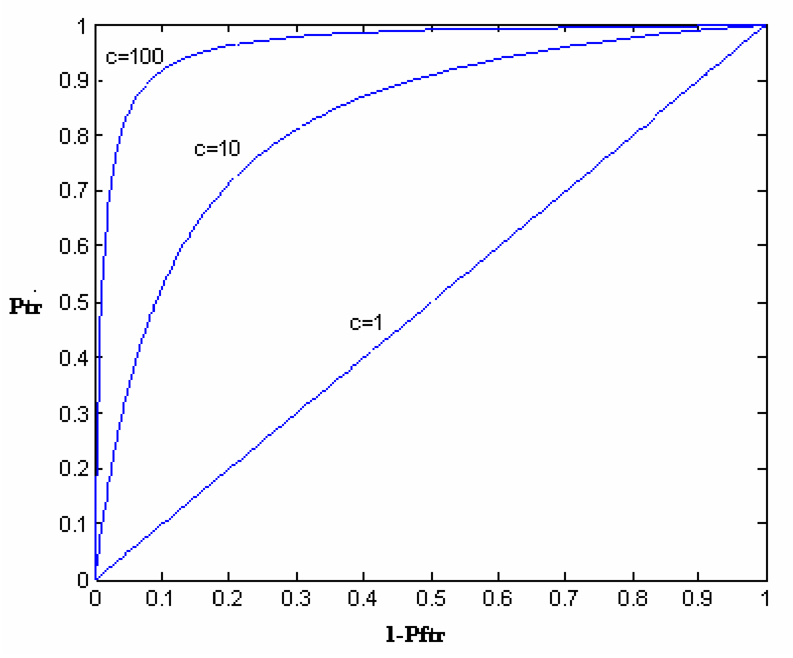


Figure 1. Receiver Operating Characteristic Curve

As seen in Figure 1, when P_{TR} gets close to unity, 1-P_{FTR} also gets close to unity. This represents the situation where the ATR threshold is kept very low so as to not miss targets, but it will also be very likely to falsely classify other objects or the background as targets. The ideal ROC curve would spike from 0 to 1 on the y –axis at x=0. Notice that as *c* increases, the ROC curve comes closer to the ideal ROC curve. Equation 1, adapted from (Moses, Shapiro, Littenberg, 1993), empirically relates P_{TR}, 1-P_{FTR}, and *c* to generate the curves in Figure 1.

$$1 - P_{FTR} = \frac{P_{TR}}{(1-c)P_{TR} + c} \tag{1}$$

Notice that the value of *c* drives the relationship between P_{TR} and P_{FTR} in Equation 1. To increase the value of *c*, parameters such as area search rate, pixel density, sensor algorithms, and the characteristic size of the targets can be altered. The actual ROC curve for an ATR based system must be determined experimentally, so Equation 1 merely represents an approximation to an actual ROC curve.

In the past, most of the target detection in simulations was completed through a confusion matrix. If the UAV came across what appeared as a target, its simulated sensor would run through a confusion matrix to determine if the detection was a true target given a known distribution of targets. While this technique provided useful simulation data, it treated the sensor as just a set of probabilities instead of an actual piece of hardware. Further, it did not allow for experimentation on hardware platforms.

Other keys to success in the cooperative control of autonomous UAV fleets include communication, decision control/task allocation, and management of uncertainty. Developing technology for UAVs to communicate, allocate the search and destroy parts of the mission, and know when a target is legitimate or not work is critical to making the battlefields of the future not only safer for our troops, but also safer for the innocent civilians caught in the crossfire. While not a focus of this research, future work must address the use of multiple experimental search vehicles to demonstrate the use of cooperative algorithms to identify targets.

In this research, the ATR system including the actual sensor and ATR algorithm will be part of a surrogate vehicle that will serve as a test bed to conduct wide area search missions. The ATR system will be characterized by experimentally determining both P_{TR} and $1-P_{FTR}$ for various conditions at a given threshold. Target size, shape, and color will all factor into this characterization for different operating conditions. Once P_{TR} and $1-P_{FTR}$ are known for a given threshold and operating condition, they can be artificially increased and decreased by simply changing the threshold. Doing this for a variety of thresholds will produce a ROC curve for the given operating condition.

1.2.2 Sensor Footprint Characteristics

As the vehicle conducts its wide area search, its sensor will have a footprint size that depends on the sensor specifications, mounting geometry, vehicle position, and altitude. For this research, a similar geometry to that of Abeygoonewardene (2006) will be used. The sensor will be mounted on the vehicle such that it has a trapezoidal footprint with length, z, and with front width, w_f, and rear width, w_b. The elevation view in Figure 2 shows the footprint length in relation to the position of the vehicle in the vertical dimension as well as the other angles and dimensions in the vertical plane.

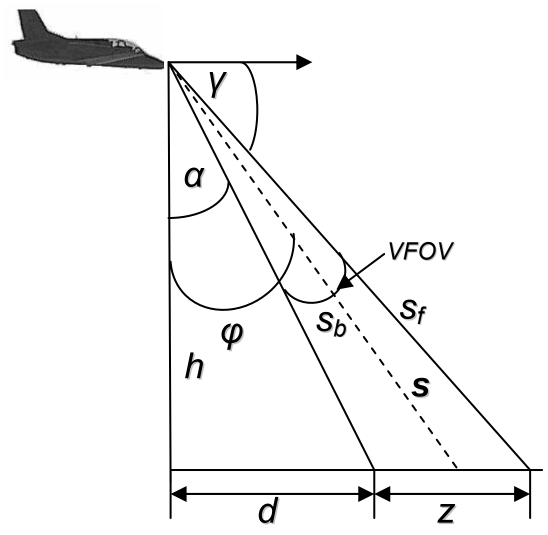


Figure 2. Elevation View of Ground Vehicle During Search

In the elevation view, the vertical field of view (VFOV), sensor height above the search area h, and the bore angle φ drive the depression angle γ, footprint length z, dead range d, slant range s, and slant angle α. Of these parameters, VFOV can be experimentally determined or obtained from manual specifications, and should stay relatively constant for a single sensor, and the depression angle, as well as the slant angle can both be determined once a bore angle is set. See below for the development of all of the necessary equations to solve for the vertical geometry of the sensor footprint.

$$\gamma = 90 - (\varphi + \frac{1}{2} VFOV) \tag{2}$$

$$\alpha = \varphi - \frac{1}{2} VFOV \tag{3}$$

$$d = h \tan(\alpha) \tag{4}$$

$$z = \frac{h}{\tan(\gamma)} - d \tag{5}$$

$$s_f = \frac{h}{\cos(VFOV + \alpha)} \tag{6}$$

$$s_b = \frac{h}{\cos(\alpha)} \tag{7}$$

The azimuthal footprint shown in Figure 3 illustrates the width of the front and rear footprints with respect to dead range and footprint length, both determined above.

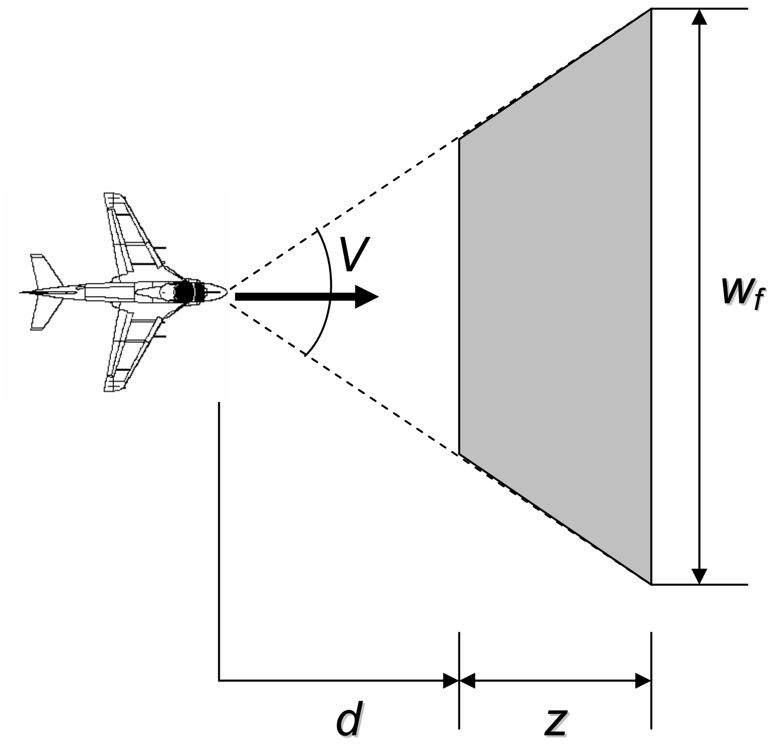


Figure 3. Azimuthal View of Sensor Geometry

Figure 4 shows the frontal view of the search vehicle's geometry. To actually determine the sensor footprint width, the two needed additional parameters are the sensor swath angle, θ, and the sensor front slant, s_f, and back slant, s_b, distances. Because the swath angle is a property of the sensor, it must be experimentally determined or obtained from specifications in a similar fashion to the VFOV angle.

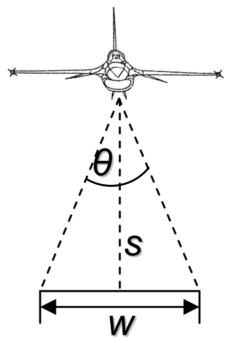


Figure 4. Frontal View of Sensor Geometry

Notice that from the geometry of the footprint in the plane, it is assumed that half of the swath angle encompasses half of the footprint width. Once the swath angle is known, trigonometry can be used to determine the sensor footprint back and front widths as seen below.

$$w_b = 2s_b \tan^{-1}(\frac{1}{2}\theta) \tag{8}$$

$$w_f = 2s_f \tan^{-1}(\frac{1}{2}\theta) \tag{9}$$

Lastly, area search rate is can be determined by taking the product of the rear footprint width and the velocity of vehicle normal to the footprint width. The rear width is selected due to the trapezoidal shape of the footprint even though the front width is wider.

Equation 10 will give a conservative area search rate value and will not account for any objects that are whole or partially located outside the rear width of the footprint.

$$\frac{dA}{dt} = w_b V \tag{10}$$

In addition to the size and shape of the sensor footprint, another consideration in targeting applications is frame overlap O_L for maximum coverage of the search area. By overlapping frames, the target can be guaranteed to be contained wholly within a single frame if its largest dimension is smaller than the overlap. Cameras with slower processing time might not be able to overlap, but if they could capture frames fast enough to ensure that each frame abuts the next, the target would still be wholly captured, but in two adjacent frames. Frame overlap is much better than abutment, but sometimes sensor processing speed and minimum vehicle speed make it infeasible. When feasible, overlap can be calculated using frame length, F_L, and frame separation, F_S, as seen below.

$$O_L = z - F_s \tag{11}$$

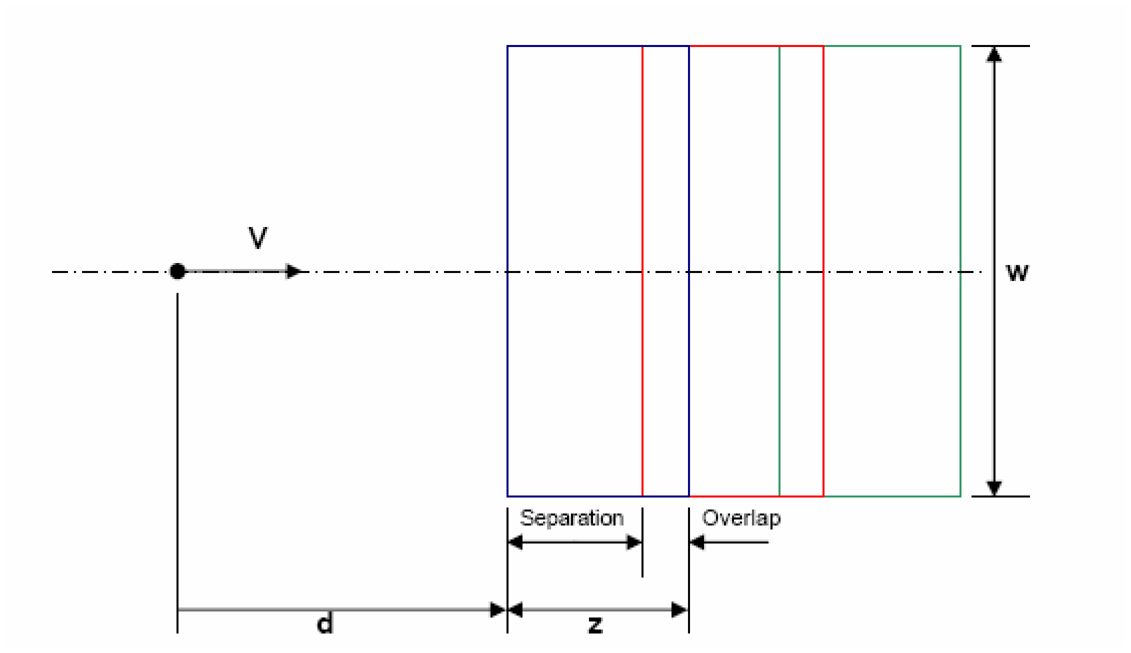


Figure 5. Frame Overlap for Straight Line Search

1.2.3 Dynamic Scaling

In the development of UAV systems, simulations are normally conducted using dynamic models from the actual vehicle being simulated. These vehicles are often quite large and, due to both cost and safety, can be prohibitive to test in the early stages of development of systems. However, there are a number of guidance and control systems that could be tested earlier in development if the vehicle was ready. To solve this problem, a surrogate vehicle can be used during the initial real world testing as long as it is dynamically similar to the actual system. These surrogate vehicles can be small less expensive UAVs or unmanned ground vehicles that match the characteristics of a larger or more expensive UAV, i.e. are dynamically similar.

The proper dynamic scaling of an experiment should produce predictable results and the vehicle should have multiple configuration capability to closely match its larger counterpart. If it can meet these criteria, it should give an accurate representation of the performance of the full size air vehicle it is representing. Once the initial surrogate vehicle is configured properly, future researchers can use this test bed to complete experiments without spending the majority of the time on the critical yet laborious task of designing & building the system.

In this particular research, there are three possible ways to conduct a real world experiment to validate the single UAV ATR computer simulation. The first and most expensive is to fly the actual UAVs on a test range with actual targets. The next choice would be to fly scale models UAVs on a test range with the targets, using dynamic scaling to ensure the integrity of the experiment. This choice is cheaper and safer than using full size UAVs. However, since the technology is still maturing, this is also risky due to the chance of losing a UAV with thousands of dollars of equipment integrated into its fuselage. The third and safest choice is to use dynamically scaled ground vehicles to represent the UAVs in a two dimensional space. The lack of an altitude dimension will be considered the same as assuming that the altitude is constant. With the current state of the technology, it makes sense to start with the scaled ground vehicles and work up to the full size UAVs when it is safe and cost effective.

In September 2006, Jeevani Abeygoonewardene showed how smaller and less complex surrogate vehicles can be used to conduct experiments that will predict the performance of their nominal counterparts (2006).
These dynamic scaling techniques, based heavily upon the Buckingham Pi theorem (1914), provide the mathematical proof that matching certain parameters between two

vehicles is enough to consider the surrogate as an accurate representation of the actual full scale vehicle.

The Buckingham Pi theorem stipulates that the solution to any differential equation, regardless of its order or nonlinearity, can be made invariant with respect to dimensional scaling as long as appropriate ratios of parameters are maintained. If these ratios of the independent variables can be maintained, two systems of different size can be said to be "dynamically similar." Even though it sounds like a simple process, the independent variable must first be identified so that non-dimensional pi groups can be developed.

The physically meaningful equation below,

$$f(q_1, q_2, \ldots q_n) = 0$$

shows each q as one of the n physically meaningful independent variables expressed in terms of k independent physical units. The above equation can be rewritten as shown below,

$$F(\Pi_1, \Pi_2, \Pi_n) = 0$$

where the Π_i are dimensionless parameters built from q_i in the form of

$$\Pi_i = q_1^{m_1} q_2^{m_2} \ldots q_n^{m_n}$$

where the exponents m_i are rational numbers. The number of Π equations is calculated from the equation below.

$$p = n - k$$

Abeygoonewardene (2006) determined that the following 9 variables in Table 3 accurately represent both the vehicle and sensor dynamics using the wide area search sensor geometry developed earlier in this thesis.

Table 3. Variables Representing Vehicle and Sensor Dynamics

d	Sensor Dead Band
V	Vehicle Velocity
g	Vehicle Required Acceleration
w	Sensor Footprint Width
$\hat{c}$	Simplified ROC Curve Parameter
ρ_t	Pixel Density
z	Sensor Footprint Length
L_{targ}	Target Characteristic Length
O_L	Frame Overlap

.

Since there are 9 physically meaningful independent variables

$$n = 9$$

The two physically meaningful independent dimensions associated with these variables are length, L and time, T. Therefore,

$$k = 2$$

Applying Buckingham's Theorem, the number for dimensionless equations (p) is,

$$p = n\text{-}k = 9 - 2 = 7$$

Since d and V cannot form a dimensionless group by themselves, they are selected as the set to use to non-dimensionalize the rest of the parameters. These variables have the following dimensions:

$$d \Rightarrow L$$

$$V \Rightarrow LT^{-1}$$

Substituting d into the equation for V and then solving for T,

$$L = d$$

$$T = dV^{-1}$$

Now each of the 9 variables can be non-dimensionalized by multiplying/dividing it by either d, V, or some combination of the two. Table 4 below shows the 9 variables, their pi group, and which variable(s) they are multiplied/divided by to form the pi group.

Table 4. Dynamic Scaling Pi Groups

Variable (units)	Pi Group #/Ratio
$z\ (L)$	$\Pi_1 = z/d$
$w\ (L)$	$\Pi_2 = w/d$
$g\ (L/T^2)$	$\Pi_3 = g(n^2-1)^{1/2}d/V^2$
$\hat{c}\ (TL^{-1})$	$\Pi_4 = \hat{c}\,V$
$\rho_T,\ (L^{-2})$	$\Pi_5 = \rho_T\, d^2$
$L_{targ}\ (L)$	$\Pi_6 = L_{targ}/d$
$O_L\ (L)$	$\Pi_7 = O_L/d$

With defined pi groups, it is now possible to attempt to match the dynamics of a surrogate vehicle (ground or air) with those of a full scale UAV (nominal). If a surrogate vehicle is chosen such that its pi groups match or closely match the pi groups of the nominal vehicle and the two vehicles share the same governing differential equations, then the vehicles have dynamic similitude.

1.3 Research Statement

The primary goal of this research is to design, build, and test a wireless, radio controlled surrogate autonomous search vehicle to physically demonstrate single vehicle wide area search techniques. This surrogate search vehicle will demonstrate the ability to identify objects as either targets or false targets through the use of ATR algorithms including the development of confusion matrices and ROC curves for the static case and for a given velocity. A secondary goal of the research is to demonstrate that the surrogate vehicle can be dynamically scaled to the nominal Sig Rascal 110 RC aircraft performing at normal operating conditions (100 feet AGL, 60-90 ft/sec). The airspeed window is the same as used by Capt Nidal Jodeh, USAF, in his research (2006) presented in March 2006. Using the same airspeed window will give future researchers performance data to use when testing the algorithms on the nominal vehicle.

Two separate theoretical calculations will be developed to predetermine search parameters for the system. The first is the calculation of the maximum number of pixels the camera will return when it has a colored target object aligned with the middle of the bottom of its field of view. This parameter will feed into the ATR threshold calculation as well as validate the geometry of the experimental setup. The second calculation will determine a steering correction angle to give the surrogate vehicle the capability to navigate directly over the top of objects it classifies as targets (i.e. engage). Although this angle will not be used during the research presented here, it can be used in future experiments that use algorithms to guide the search vehicle through a search area.

1.4 Summary

Autonomous UAV research is coming more into the spotlight as the United States continues to fight in asymmetric conflicts around the world. The development of this technology will help keep US forces further away from the dangers on battlefields around the world and more aware of the environment in which they are fighting. To more quickly field these unmanned systems, a surrogate vehicle will be developed to demonstrate the guidance and control systems on a smaller scale resulting in quicker and safer testing of the system.

Autonomous target recognition and dynamic scaling will be used to design the surrogate vehicle. Implementing these two concepts into the surrogate will allow its sensor to closely match the performance of an operational system and allow the guidance and control systems to be developed and tested to meet the warfighter's needs prior to the vehicle's first flight. To design and build this surrogate vehicle system, its component hardware needs to be identified. Chapter 2 will describe each of the components used in the surrogate as well as the hardware and software integration necessary to make the system functional. Chapter 3 will discuss the development of the ATR algorithm used in this research, including the initial ATR threshold and the actual wide area search procedure. Chapter 4 will describe the results of static and dynamic search experiments to determine experimental ROC curves for the surrogate, as well as a dynamic scaling analysis using theory developed in Chapter 1. Finally, Chapter 5 will offer a summary of the research presented in this thesis and also recommendations for future work to further develop the wide area search surrogate vehicle system.

2. Search Vehicle System Architecture

The hardware for this ground based autonomous search and destroy surrogate include the Tamiya RC Mammoth Dump Truck (Tamiya, 2007), Kestrel Autopilot (Procerus, 2007), Aerocomm 4790-1000M OEM wireless transceiver Software Development Kit (Aerocomm, 2007), and the CMUcam2 vision sensor camera (CMUcam2, 2007). All products, with the exception of the truck, which is no longer in production, and their accompanying software/hardware are available commercially through their respective manufacturer's websites on the World Wide Web. Each piece of hardware will be discussed in more detail in this section, including the features that make them all good choices to fulfill the necessary functions for this research.

2.1 Tamiya RC Mammoth Dump Truck

The Tamiya RC Mammoth Dump Truck (2007) is a radio controlled 1/20 scale dump truck with shaft driven all time 4 wheel drive, sturdy suspension, and a 540 motor. This motor is powered by a single 6 cell 7.2 V nickel-metal hydride (Ni-MH) battery pack. This robust platform is roughly 20.6 inches long with an 11 inch wheelbase and an 11.6 inch front and rear track. With a 1.6 inch minimum clearance, the vehicle stays very low to the ground so it must be used on even terrain.

Figure 6 shows a side view of the truck as well as the large 6.14" x 2.36" rubber tires used to help move the 12.2 pound vehicle. According to the manufacturer's website it is capable of speed from a slow crawl to cruising speed, which we estimate to be at least 5 mph. With this span of controlled speeds, this vehicle is a good candidate for this ground based experiment because it can operate in the slower range of speeds needed to scale to the 30-40 knot cruise speed of a Sig Rascal 110 (Jodeh, 2006).

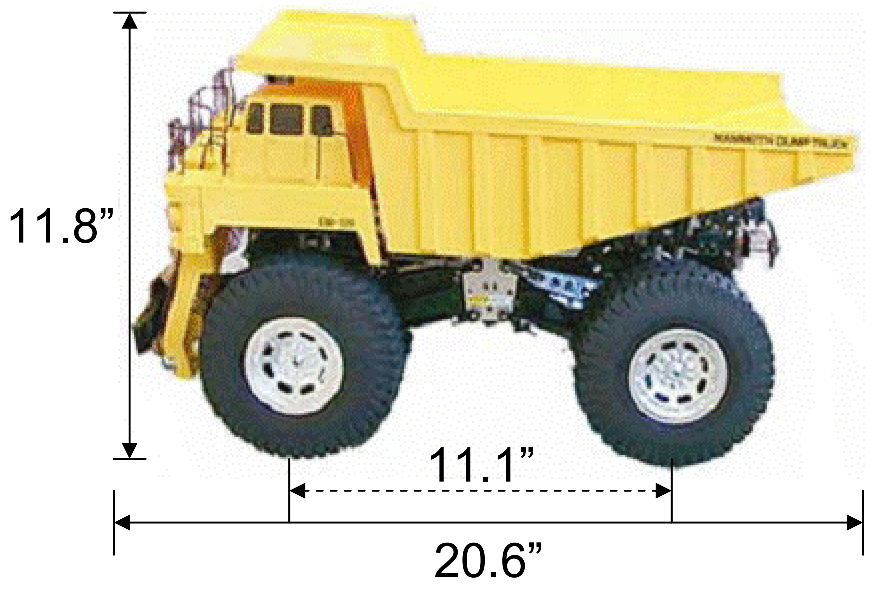


Figure 6. Tamiya 1/20 Scale RC Dump Truck

2.2 Kestrel Autopilot System v 2.2

The guidance for the search system comes from the Kestrel autopilot system, manufactured by Procerus Technologies in Vineyard, Utah (Procerus 2007). This autopilot provides the vehicle with its autonomous guidance and control ability with its GPS (Global Positioning System) and INS (inertial navigation system). The system is comprised of the actual onboard autopilot system and the ground station.

One of the main reasons the Kestrel system was selected for the system is the small size and weight of its onboard autopilot box. Since this system is normally integrated into UAV systems, where size and weight are restrictions are more stringent, the autopilot box was designed to easily fit into the palm of a hand. It weighs only 16.65 grams and is 2.375” L x 1.5” W x .875” H (Figure 7). An autopilot of this size can be easily integrated into any one of multiple free cavities in the frame most radio controlled trucks.

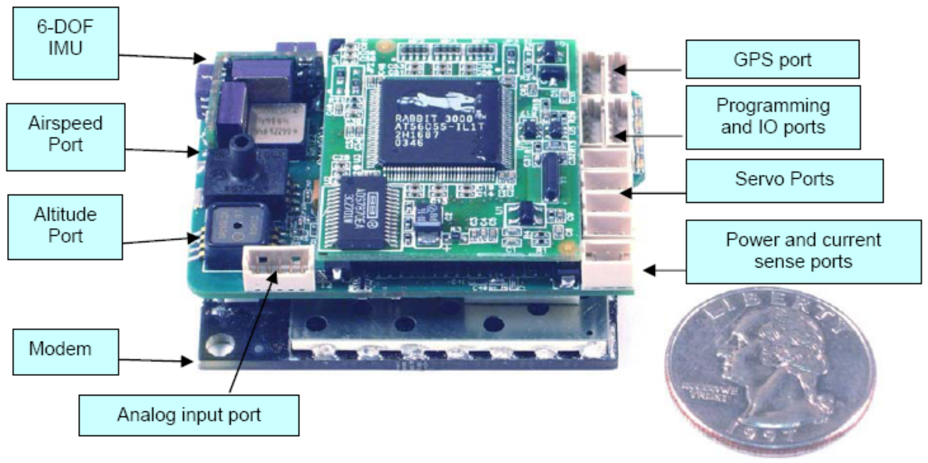


Figure 7. Kestrel Onboard Autopilot Box Input/Output Port Description, (With Permission © Copyright 2006 - 2007. Procerus Technologies. All Rights Reserved.)

As mentioned above, the Kestrel is normally designed to provide navigation and real time telemetry to UAVs, but it should also work for this experiment since the ground vehicles can be related to air vehicles flying at a constant altitude. The onboard portion of the autopilot system (Figure 8) includes not only the autopilot box (differential and absolute air pressure sensors, 3-axis rate gyros, accelerometers), but also a GPS receiver and a dipole antenna to wirelessly transmit telemetry to a 4.5” L x 3.675” W x 2.25” H Commbox transceiver.

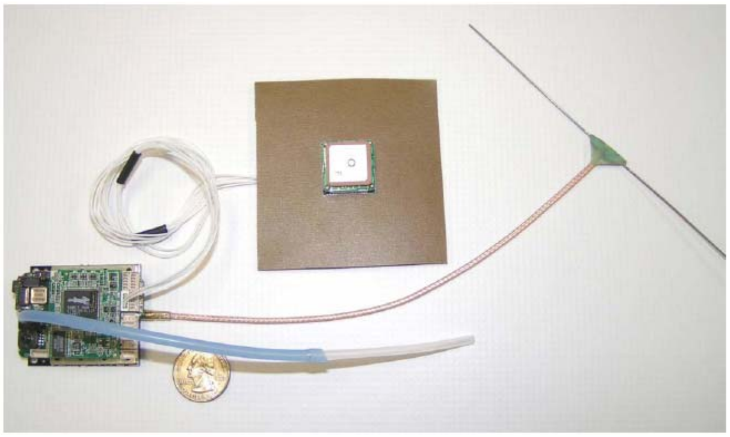


Figure 8. Kestrel Autopilot with GPS receiver, dipole antenna, and pitot tube,
(With Permission © Copyright 2006 - 2007. Procerus Technologies.
All Rights Reserved)

The ground based portion of the Kestrel Autopilot System consists of a Commbox receiver, RC transmitter, and the Virtual Cockpit software loaded onto a laptop computer. (Figure 9). This ground station setup allows for all telemetry data to be relayed from the autopilot onboard the vehicle to the laptop via the Commbox through a R232 9-pin serial cable. If manual control of the vehicle is needed, an RC transmitter can be connected to the Commbox and when configured properly the vehicle will respond to transmitter commands instead of autopilot commands from the ground station.

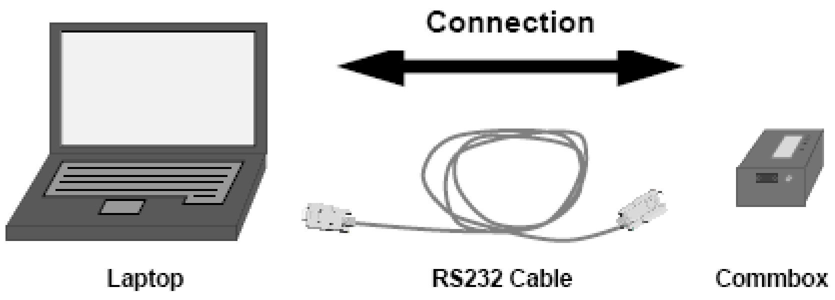
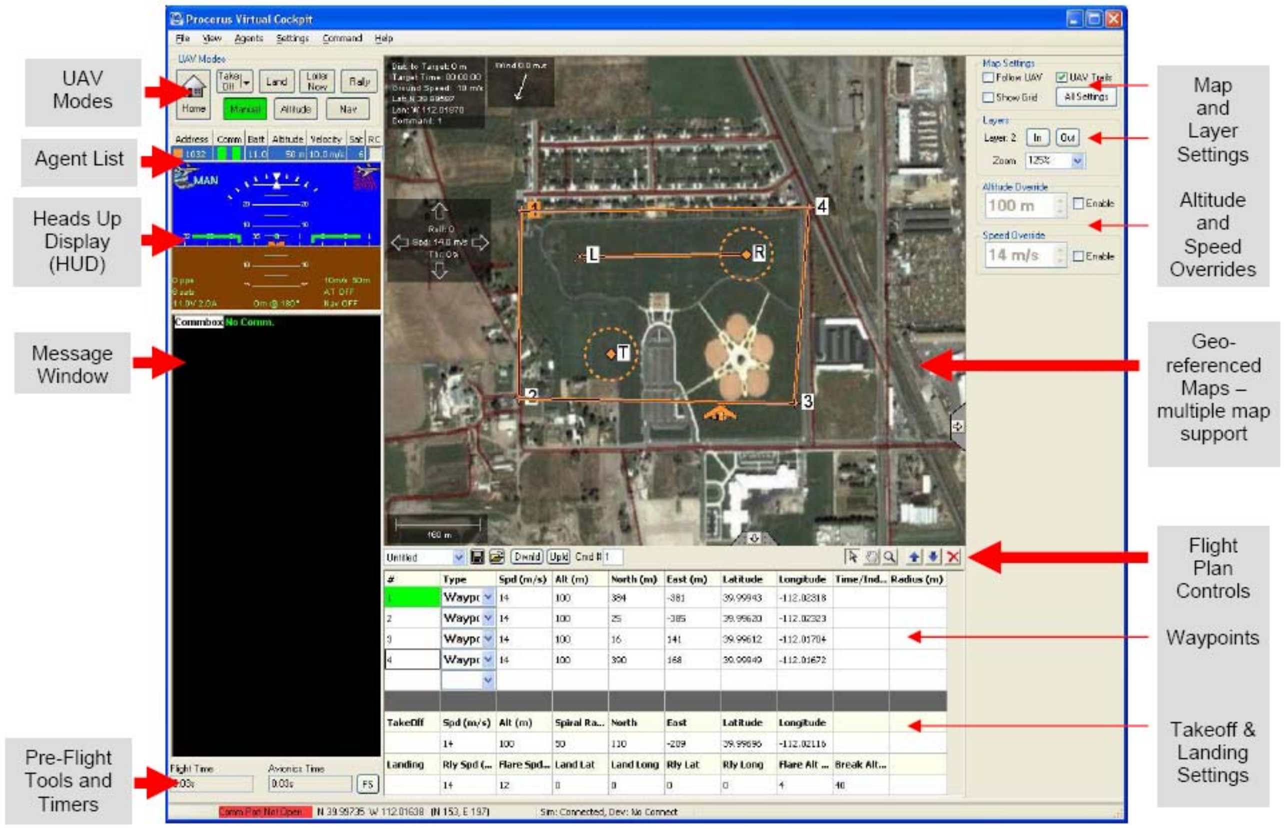


Figure 9. Kestrel Autopilot Ground Station Setup,
(With Permission © Copyright 2006 - 2007. Procerus Technologies.
All Rights Reserved)

The final portion of the Kestrel Autopilot ground station is the Virtual Cockpit software that acts as a graphical user interface (GUI) shown in Figure 10. The GUI can be used to set vehicle parameters and send speed and navigation commands to the vehicle as well as receive telemetry data from the vehicle. A short list of telemetry data available includes vehicle position, speed, acceleration, altitude, and heading information. Because the vehicle has both a GPS receiver and an INS, some of the telemetry data is received from two different sources.



#	Type	Spd (m/s)	Alt (m)	North (m)	East (m)	Latitude	Longitude	Time/Ind...	Radius (m)
1	Waypc	14	100	384	-381	39.99943	-112.02318		
2	Waypc	14	100	25	-385	39.99620	-112.02323		
3	Waypc	14	100	16	141	39.99612	-112.01704		
4	Waypc	14	100	390	168	39.99940	-112.01672		

TakeOff	Spd (m/s)	Alt (m)	Spiral Ra...	North	East	Latitude	Longitude	
	14	100	50	110	-209	39.99696	-112.02116	

Landing	Rly Spd (...	Flare Spd...	Land Lat	Land Long	Rly Lat	Rly Long	Flare Alt ...	Break Alt...
	14	12	0	0	0	0	4	40

Figure 10. Kestrel Autopilot System Virtual Cockpit Screenshot
(© Copyright 2006 - 2007. Procerus Technologies. All Rights Reserved)

While it seems like a busy interface, a large majority of the screen is a map to show the location of the vehicle in two dimensional space. Because the GUI is set up for UAV flight, several of the options will not be used in this ground based research. Gains and other parameters for both elevator/pitch and rudder/yaw are completely ignored due to the way the autopilot will be integrated into the steering mechanism of the truck. Also, because the vehicle(s) will be driven using the RC mode or the autonomous waypoint navigation mode for the majority of the time, the other modes including, takeoff, landing, loiter, home, rally, manual, and altitude, will be used rarely if at all.

2.3 CMUCam2 Camera and Processor

The Carnegie Mellon University Camera 2 (CMUcam2, 2007) was chosen as the sensor to complete the tasks required in this experiment. This camera is the second in the series of cameras developed by Carnegie Mellon University, following their CMUcam development in 2001. It is commercially available through Seattle Robotics and Acroname, Inc in the United States.

The CMUcam2 system (Figure 11) is made up of an OV6620 Omnivision CMOS (complementary metal-oxide semiconductor) camera interfaced with a Ubicom SX52 microcontroller. Some of its several features useful to targeting applications include onboard image processing, video output, color tracking, and motion detection.

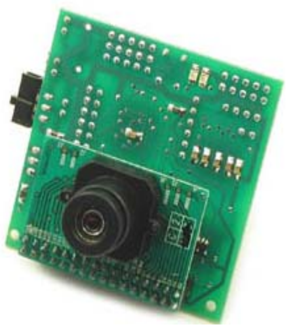


Figure 11. CMUcam2 Vision Sensor
Courtesy of Acroname Inc, www.acroname.com

The ability to process images in real time (or as close to real time as possible) gives the targeting vehicle the ability to act on this information almost instantly (multiple images per second). Image processing speed is critically important in this research because the system will have less time to make a decision on a target before it leaves the field of view due to the smaller scale of this research. If the camera can process multiple images per second, the targeting algorithm can make essentially real time target decisions while the potential target is still in the field of view of the camera. It also opens up the opportunity for other vehicles to be called in to make a determination if necessary before the object is classified as a target or as a false target. This capability should be able to greatly reduce the FTAR.

Other features of the CMUcam2 that are useful to search and targeting applications include video output, color tracking, and motion detection. The video output feature of the CMUcam2 allows for the operator to view the search area as the surrogate is actively pursuing targets. While this second pair of eyes would not be in keeping with the concept of a truly autonomous search, it is extremely helpful during experimental validation of ATR algorithms.

More tools embedded into the CMUcam2 include color tracking and motion detection. Both can be useful if target size, shape, or color information is previously known and can be "taught" to the ATR system. If the target does not need to be eliminated, but instead followed to help produce bigger targets, tracking it using color and motion detection will ensure that it is kept in the field of view. This type of surveillance can be helpful in picking up travel patterns and it gives time to identify the object as a high or low priority target.

2.3.1 CMUCam2 Field of View Experiment

Similar to the process used in by Mike (Mike, 2006) in his thesis, the field of view for the CMUcam2 was determined by capturing an image of a grid with the camera (bore angle aligned normal to the grid) at a known distance from the grid. Knowing the horizontal and vertical dimensions captured by the image, and the distance of the camera from the image, a simple arctangent can be used to determine both the vertical field of view, VFOV, and the swath angle, θ. Table 5 shows the results obtained from this experiment with the CMUcam2 used for this research as well as those calculated in (Mike, 2007:7).

Table 5. CMUcam2 Field of View Angles

	Vertical FOV	Horizontal FOV
Rufa - MS Thesis	45.13°	30.79°
Mike - BS Thesis	44.91°	29.76°

The results from this experiment correlate closely to the experiment conducted by Mike to determine the CMUcam2 horizontal and vertical field of view. Since both fields of view were off by 1 degree or less, they are sufficient to use when calculating specific sensor geometry information in Chapter 4.

2.4 Aerocomm 4790-1000M OEM Wireless Transceiver

To make the system truly wireless, a wireless serial connection between the camera and the ground station is necessary. While the Kestrel Autopilot has extra data ports to send wireless signals, it was decided that giving the camera its own dedicated transceiver set would provide the best results since each set of wireless transceivers could operate independently. The Aerocomm 4790-1000M 900 MHz Transceiver (Aerocomm, 2007) was selected to provide wireless transmissions between the camera and ground station due to its ease of use and range. According to the Aerocomm website, this transceiver has a range of up to 20 miles with a high gain antenna. Although, this research will not require that type of range, future applications of these wireless serial radios might require a larger range.

The kit ships from the factory with two transceivers mounted to an adapter board as shown in Figure 12 below. These adapter boards give the designer the capability to integrate these wireless serial radios with USB, RS-232, or RS-485 type peripheral equipment and ground stations.

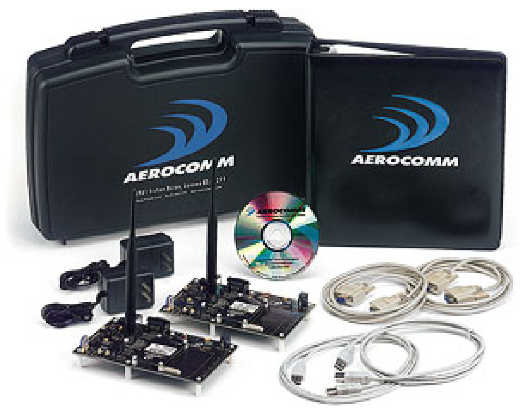


Figure 12. Aerocomm 4790-1000M Wireless Transceiver SDK
(Reproduced with permission of Aerocomm, Inc)

In most applications, the two boards work together on one serial port to provide a two way wireless data flow from one peripheral device, but the introduction of a third board gives the capability for another peripheral device to be added to the system on its own serial port. The board wired to the ground station can be configured to receive signals from both peripheral devices through two separate serial ports.

2.5 Ground Vehicle System Vehicle Electronics Integration

The integration of the CMUcam 2 with its wireless serial connection and Kestrel Autopilot System into the Tamiya radio controlled truck was completed in two parallel phases. The first phase was the physical integration of the camera (with transceiver) and autopilot into the sensor deck of the truck, while the second phase was the writing and integration of the ATR software to run the camera, receive and process its data, and make a targeting decision.

2.5.1   System Hardware Integration – Kestrel Autopilot System & CMUcam2 Vision Sensor System

Due to volume constraints within the dump truck, the autopilot box, dipole antenna, GPS receiver, camera, and wireless serial transceiver were installed on the sensor deck seen in Figure 13.

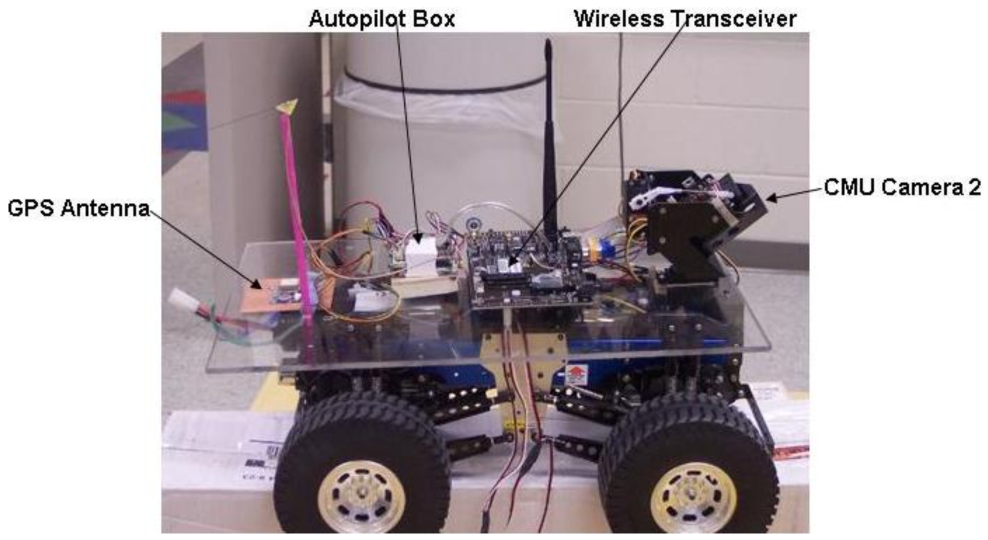


Figure 13. Ground Vehicle Sensor Deck with all Components Installed

For the truck to be driven autonomously, its power and steering mechanisms need to be connected directly to the onboard autopilot box because this device takes over the role of the receiver that normally sends steering and throttle commands to the servos. The truck steering cable is connected to the aileron channel on the autopilot (Channel 1), while the truck's throttle is connected to the throttle channel on the autopilot (Channel 4). The final necessary connection is from the autopilot to a pair of 3 cell lithium polymer (LiPo) batteries that power both the autopilot and the other sensor deck devices. Figure 14 shows all of the necessary connections to the autopilot.

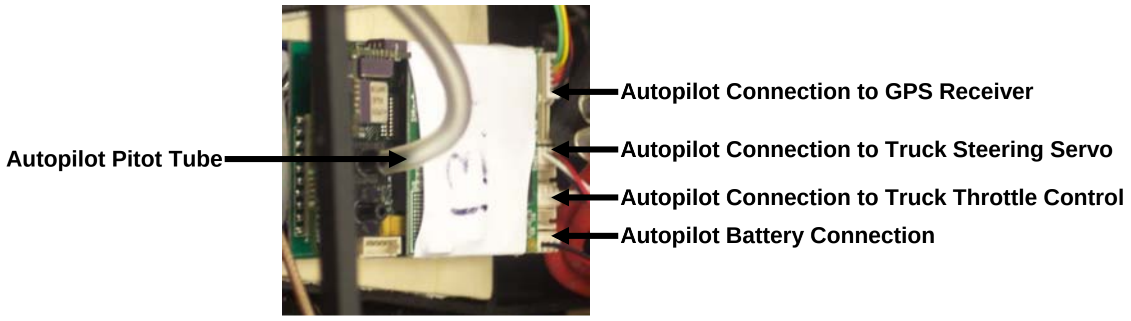

Autopilot Connection to GPS Receiver
Autopilot Pitot Tube
Autopilot Connection to Truck Steering Servo
Autopilot Connection to Truck Throttle Control
Autopilot Battery Connection

Figure 14. Kestrel Autopilot Box with Steering and Throttle Connections to Truck

The autopilot's GPS receiver is secured with velcro to the rear end of the sensor deck with the antenna facing skyward so that when the truck is upright it will have a direct line of site to its satellites. The dipole antenna is secured to an antenna mast that is mounted through a hole in the rear part of the sensor deck.

The other system integrated into the truck frame is the CMUcam2 and its wireless serial transceiver. Both devices are powered by the LiPo batteries, but only the camera has its own power switch. As soon as the transceiver is connected to the battery, it becomes energized. Due to space constraints under the body, both the camera and transceiver are placed on the sensor deck as shown in Figure 13.

2.5.2 System Software Integration – Kestrel Autopilot System & CMUcam2 Vision Sensor System

As with any hardware installation, the companion software must be properly configured to ensure each of the components work as expected. To make the complete system work properly, the Kestrel Autopilot Software and CMUcam2 software both needed to be configured to communicate with the ground station and also with each other. For ease of use, Matlab was selected as the software programming tool for the CMUcam2, while the Kestrel autopilot used Procerus' own Virtual Cockpit 2.2 software (Procerus, 2007).

The CMUcam2 software integration consists of a Matlab routine designed to communicate directly with the ground station passing preprocessed information from the camera. This preprocessed data comes through as packets that must be fully captured to use the information for targeting purposes. These packets come through as tracking data, RGB histogram data, raw image data, or mean frame data. With four different types of data packets, there are many different ways to use the frame data for processing. Two processes are shown in the following paragraphs.

The first processing option is to simply capture the raw pixel data with full frames and use the red, green, and blue pixel data in the ATR algorithm to determine whether the frames included the target or not. Since the camera captures the frames in raw byte format, a Matlab program is needed to decode this binary data and discard certain non pixel information passed with each frame. This non pixel information includes frame synchronization bytes, frame size, and column synchronization bits. The process to capture a frame and get its pixel information into usable format for both analysis and viewing is shown below. Upon completion of this process, the image matrix can be fed into an ATR algorithm for a targeting decision.

CMUcam2 Frame Capture Process

1. Open camera's serial port
2. Send the "SF" command to the camera to capture a frame
3. Send a command to the camera to read raw frame data to Matlab
4. Close the camera's serial port
5. Remove non pixel information from frame capture data stream (147 bytes for low resolution capture)
6. Reformat pixel information into format compatible with Matlab's image command (87 rows x 143 columns x 3 colors). If only one color is used, the matrix will be 87 x 143 x 1.

The second processing option is to predetermine the color of the targets and then set the camera to find that specific color within each frame. The camera accomplishes this task by returning a T packet (CMUcam2, 2007) with data shown in Table 6.

Table 6. Tracking Packet Description for CMUcam2

T	denotes tracking packet
mx	x-centroid of tracked blob (pixel #)
my	y-centroid of tracked blob (pixel #)
x1	x-upper left hand of blob (pixel #)
y1	y-upper left hand of blob (pixel #)
x2	x-lower right hand of blob (pixel #)
y2	y-lower right hand of blob (pixel #)
pixels	tracked pixels in FOV (capped at 255)
confidence	confidence of tracked pixels (capped at 255)

This whole process occurs at 15 frames per second when the camera is connected to the ground station (e.g. laptop) via a serial cable. When the camera and laptop are connected via a wireless serial connection through the transceivers, the frame rate is reduced to 5-6 frames per second, but is still adequate for tracking stationary targets. This process will enable the vehicle to move faster during the search and scale better with a Sig Rascal

110, but it cannot feed actual images without a secondary video camera mounted onboard. However, the speed of the data coming into the ground station made this option more compatible with the type of data this research is looking to gain. The process to capture tracking data is shown below.

CMUcam 2 Target Tracking Process

1. Open camera's serial port
2. Send the "TC [Rmin Rmax Gmin Gmax Bmin Bmax]" command to the camera with RGB min and max values
3. Send "fscanf" command to the camera to read the "T" packet information into Matlab
4. Determine if a complete packet was received. If a packet is missing information, the algorithm will insert a place holder into its place.
5. Plot the location of the tracked color using the "mx" and "my" values to get an idea of where the target is in the camera's field of view.
6. Use the location of the tracked color to steer the vehicle towards that point by determining the position of the target relative to the nose of the vehicle.
7. Close the camera's serial port

When the search vehicle is set to just search the area and not act on any target information it receives, it is possible for the two programs to run independent of each another. In this case, only steps 1-5 in the tracking process are used. However, if the vehicle needs to change waypoints based on its search results, the two different interfaces will need to work together to share information to update waypoints in the Virtual Cockpit, thus using all seven steps in the tracking process.

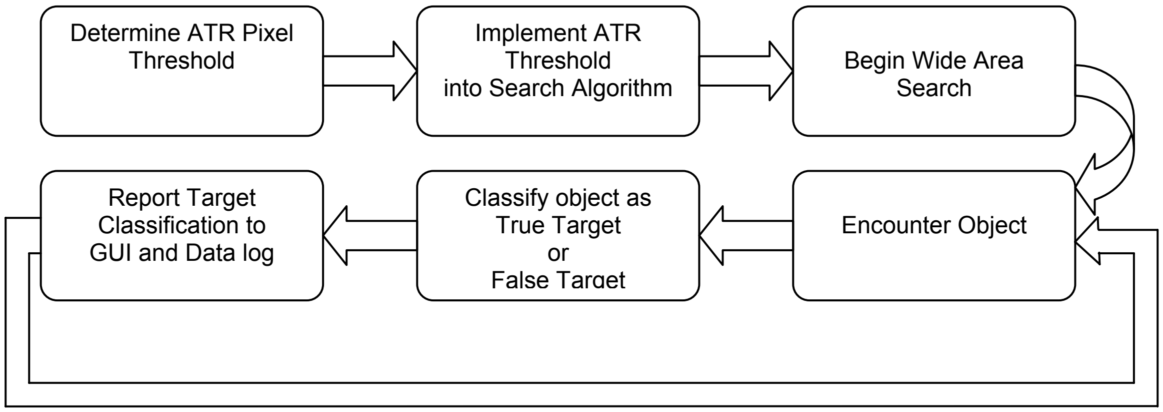
2.6 Summary

Integrating an RC truck with a camera, wireless transceiver, and autopilot yields a surrogate system that can be used to complete a wide area search of an area. While the hardware integration was fairly straightforward, determining the type of frame data needed from the camera made the software integration more complex. An author modified script (von Kraus, 2007) utilized Matlab's prebuilt serial port communication commands to enable to the camera to send frame data wirelessly to the ground station at roughly 6 frames per second. This script can be found in Appendix B.1.

With the surrogate vehicle search system built, the next step in completing the experiment is to determine the process the system will use to turn sensor frame data into useful targeting information (i.e. develop the ATR algorithms). The specific wide area search algorithm used to make targeting decisions for the experiments in this research will be discussed in Chapter 3.

3. Wide Area Search Algorithm Development

The process to build an algorithm that can predict whether an object in a sensor's field of view is a target or not consists of several steps that will be discussed in the following pages of this chapter. The steps to developing the algorithm include setting the ATR pixel threshold, searching the area, classifying an object upon encounter, and reporting the object as a target or false target. Figure 15 shows the general flow of the ATR algorithm, however, the steps will be explained in further detail in the following sections.



Determine ATR Pixel Threshold
Implement ATR Threshold into Search Algorithm
Begin Wide Area Search
Encounter Object
Classify object as True Target or False Target
Report Target Classification to GUI and Data log

Figure 15. Flow of Wide Area Search ATR Algorithm

A useful piece of information that can be implemented into the algorithm in the future is a vehicle steering correction angle. Solving for this angle will give the vehicle the ability to engage the target it has identified by steering towards to target. Because this research will not experimentally implement the steering correction angle, its derivation will be shown in Appendix C.

3.1 Search Vehicle Object Pixel Geometry

If an object's characteristic length is known, it is possible to calculate an estimate of the maximum number of pixels the camera can put on the object when it is aligned with the vertical centerline of the field of view and the rear horizontal edge of the field of view as shown in Figure 16. Also, given an object's upper left and lower right bounding coordinates from the sensor, it is also possible to calculate the angle, ψ, between the velocity vector of the surrogate vehicle and the centroid of the object. This measurement can be used in future surrogate guidance and control work.

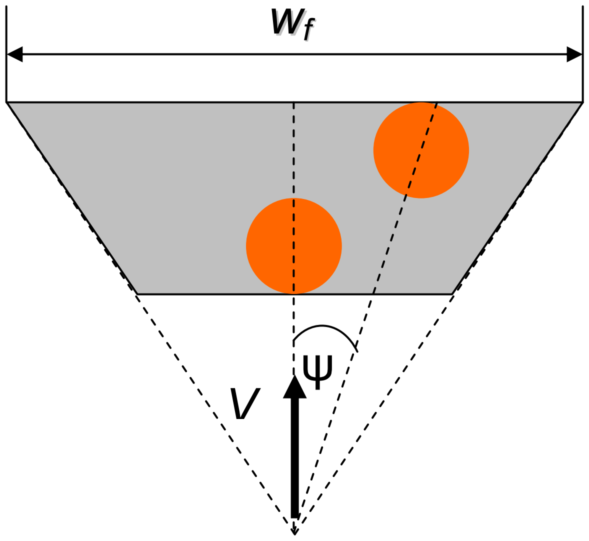


Figure 16. Sensor Frame Ground Projection with Objects in Field of View

3.1.1 Object Area Vertical Pixels Calculation

The first step to calculating the maximum number of pixels the camera can put on the object is to determine its vertical angle, β_{obj}. This angle subtends the distance between the rear edge of the search footprint to the front edge of the object as seen in

Figure 17. Knowing the object's characteristic diameter, D_t, the equation below will give its vertical angle.

$$\beta_{obj} = \tan^{-1}\left(\frac{d + D_t}{h}\right) - \alpha \qquad (12)$$

Once the vertical object angle is calculated, the number of vertical pixels on object can be determined by the following equation knowing that the camera has 143 vertical pixels.

$$P_{ObjVert} = \beta_{obj}\frac{CameraVerticalPixels}{VFOV} \qquad (13)$$

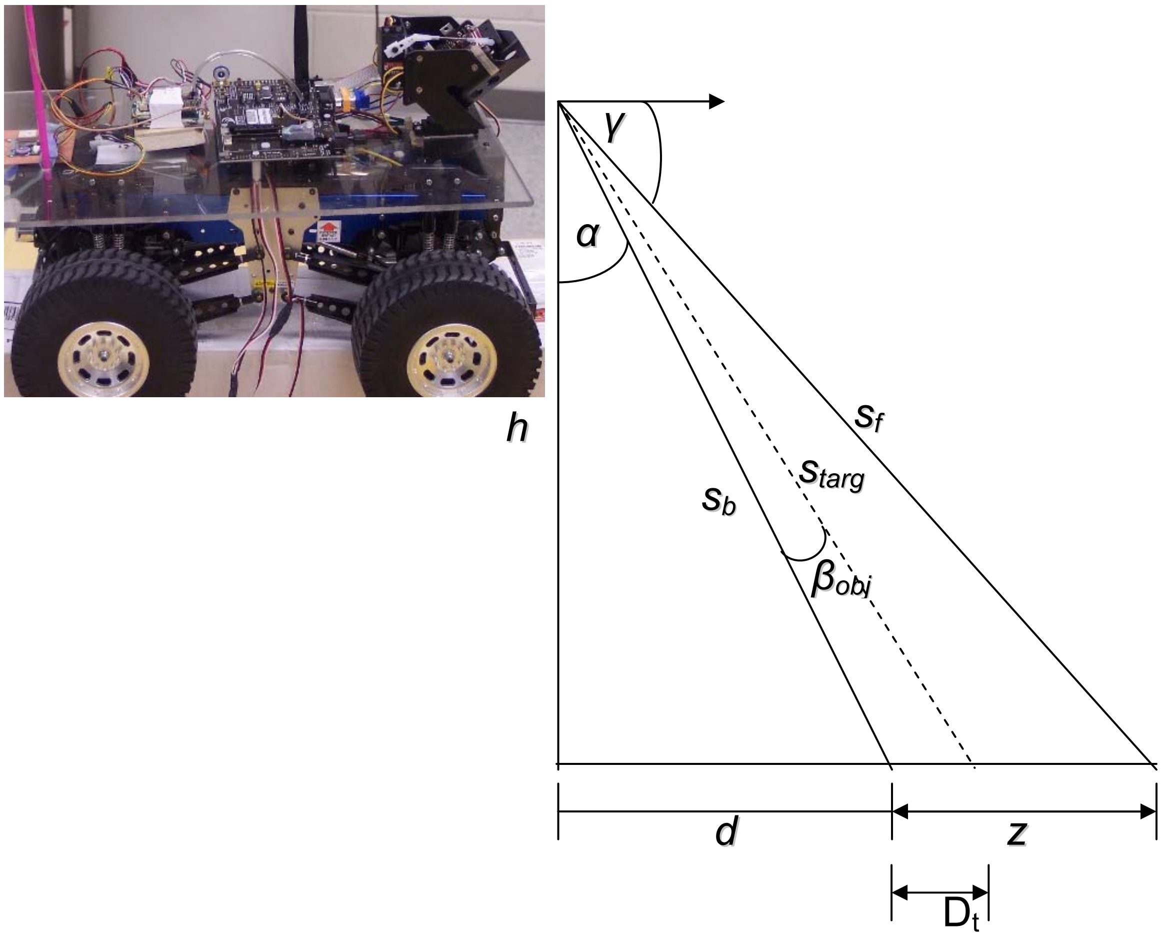


Figure 17. Estimated Vertical Object Angle, β_{obj}

3.1.2 Object Area Horizontal Pixels Calculation

Once the number of vertical object pixels is known, the next step is to determine the number of horizontal object pixels at the rear and front of the object to properly correlate this estimate with the data given using the sensor. The angles used to determine the number of horizontal pixels are shown in Figure 18 and are calculated below.

$$\chi_{obj} = 2\tan^{-1}\left(\frac{\frac{1}{2}D_t}{s_{obj}}\right) \tag{14}$$

$$\chi_b = 2\tan^{-1}\left(\frac{\frac{1}{2}D_t}{s_b}\right) \tag{15}$$

Similar to the calculation of the number of vertical pixels on the object, the number of horizontal pixels on the object can be calculated by knowing the above two angles and horizontal pixels to swath angle ratio. The two equations below represent the number of horizontal pixels at the rear edge of the object and the number of pixels at the leading edge of the object knowing that the camera has 87 total horizontal pixels.

$$P_{ObjHoriz} = \chi_{obj}\frac{CameraHorizontalPixels}{\theta} \tag{16}$$

$$P_{BHoriz} = \chi_b\frac{CameraHorizontalPixels}{\theta} \tag{17}$$

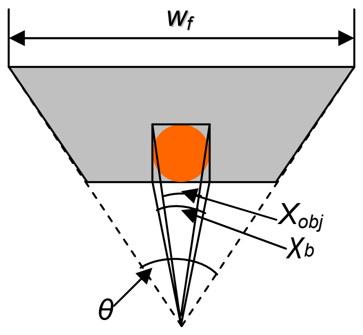


Figure 18. Estimated Horizontal Object Angles, χ_b and χ_{obj}

Now that the number of vertical pixels and the number of horizontal pixels on object are known, the pixel area can be determined by correlating these values to the location of the object's upper leftmost pixel and the lower rightmost pixel in the field as shown in Figure 19. When the bore angle of the camera is not equal to 0 or 90 degrees, these two pixel locations will not be the same distance from the vertical centerline of the frame. However, this is not a problem because the camera's raw output provides both pixel location coordinates. Therefore any theoretical area calculation using those values can also be verified experimentally. The upper left and lower right pixel coordinate x and y equations, U_{Lx}, U_{Ly}, L_{Rx}, and L_{Ry} respectively, as well as the object pixel, A_{ObjPix} area equation are shown below.

$$U_L x = \frac{CameraHorizontalPixels - P_{ObjHoriz}}{2} \tag{18}$$

$$U_L y = CameraVerticalPixels - P_{ObjVert} \tag{19}$$

$$L_R x = \frac{CameraHorizontalPixels + P_{BHoriz}}{2} \tag{20}$$

$$L_R y = CameraVerticalPixels \tag{21}$$

$$A_{ObjPix} = (L_R x - U_L x)(L_R y - U_L y) \tag{22}$$

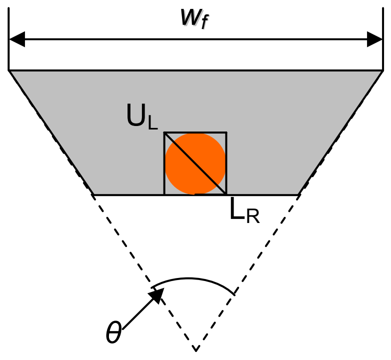


Figure 19. Upper Left and Lower Right Object Pixels used to Calculate Object Area

3.2 System Target Search Algorithm

Once the target pixel threshold has been set, the surrogate vehicle can begin the wide area search. This search consists of encountering, classifying, and reporting objects in the sensor's field of view as it moves through the search area. In chronological order, the system must first search for objects in its field of view that have a target characteristic (color will be used as the primary target characteristic in this research). Upon encountering an object with the target characteristic, it must then classify it as either a target or false target. The final step is to report its classification to the operator through some type of interface so that a disposition can be made depending on whether it has determined the object is a target or not.

3.2.1 Searching

For the surrogate to start searching, it must be given either an initial heading and speed (for straight line type search patterns) or waypoints for other types of search patterns. With the current system, this can be accomplished by giving the vehicle a direction and velocity command using a radio transmitter or through setting waypoints using the Kestrel's Virtual Cockpit software. Once the vehicle starts moving, the sensor must be activated by starting the CMUcam2 Matlab script. Upon completion of these two steps, the system is ready to start classifying objects as they are encountered.

3.2.2 Classifying

As the sensor captures frames in the wide area search, it will encounter objects in its field of view that have the target color characteristics. When it detects these objects, it must classify them as targets or false targets. If the ATR system is properly characterized, it would be expected that it will properly classify each object it encounters most of the time. However, there should be instances where it improperly classifies targets as false targets or false targets as targets to account for the realism that should be expected on a battlefield.

For the CMUcam2 to classify an object as either a target or a false target, it must first detect that object in it is field of view. If it sees an object in the field of view with the target characteristic color, the system will calculate the object's number of target colored pixels and compare that value to the pixel threshold determined by the maximum number of false target pixels. A pixel count higher than the threshold returns a target classification while a pixel count lower than the threshold returns a false target classification.

3.2.3 Reporting

For the user, the most visual step in the algorithm is how the search results are reported. This tracking script will use a Matlab figure scaled to the size of the sensor footprint to symbolically display locations of true and false targets. If the system thinks that it is seeing a target, it will display a red star at the object's centroid as shown in Figure 20.

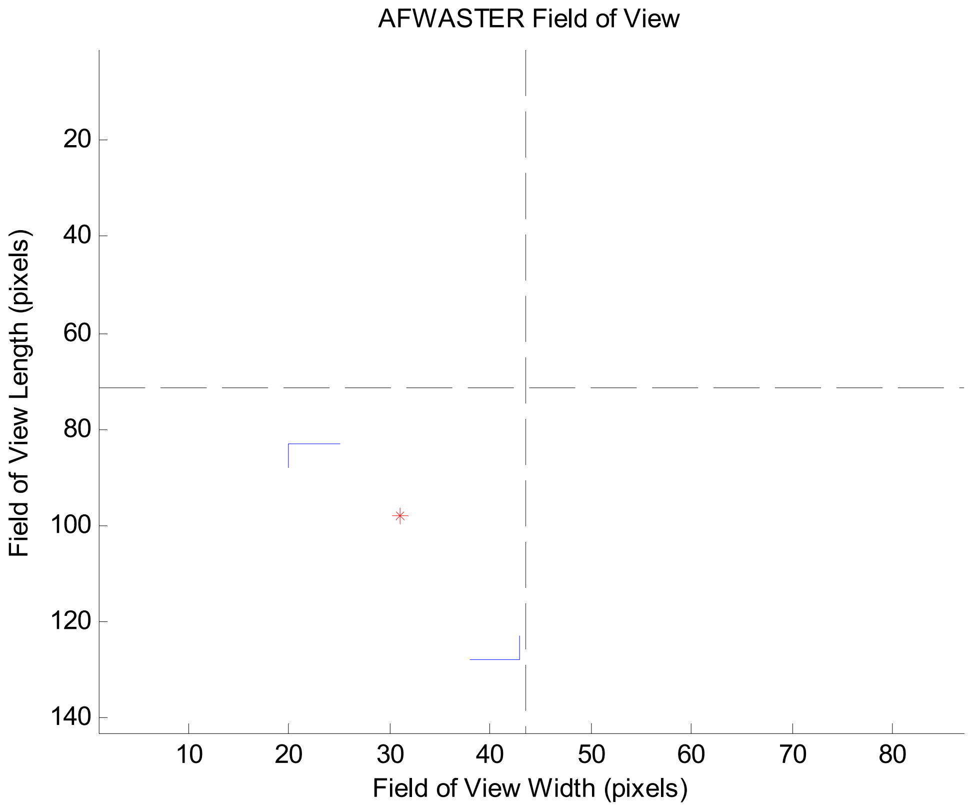


Figure 20. CMUcam2 GUI: True Target Detection

If the system thinks that it is seeing a false target, the object's centroid will be represented as green star as shown in Figure 21.

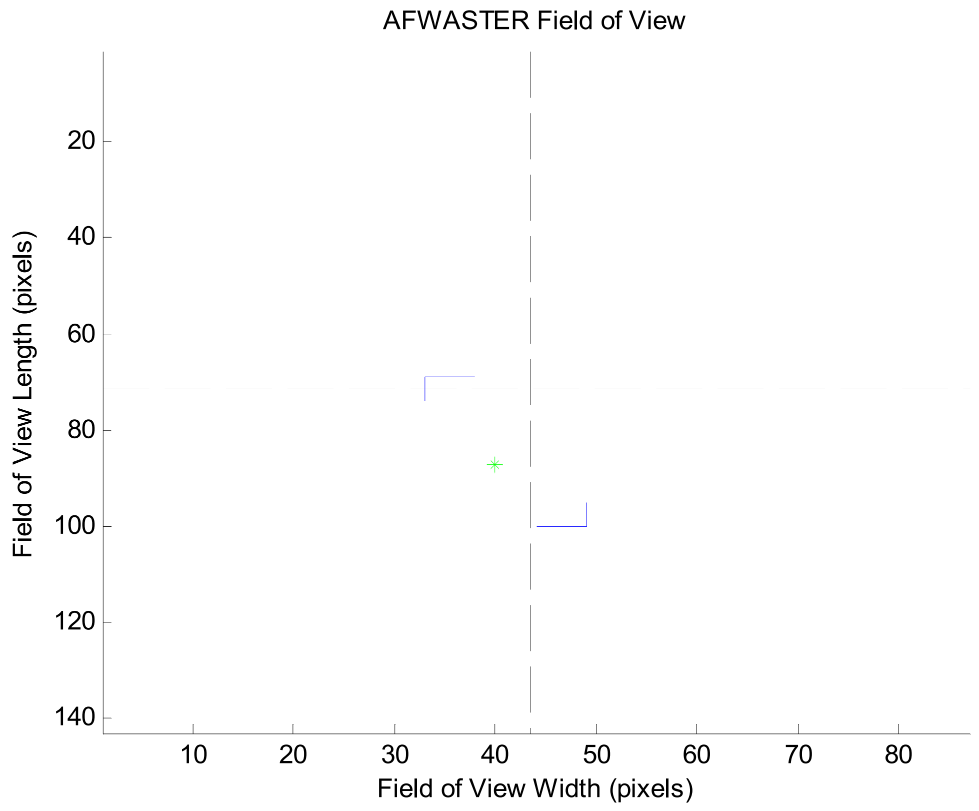


Figure 21. CMUcam2 GUI: False Target Detection

Other target information, such as brackets showing the target's upper left and lower right bounds will also be displayed. To avoid confusion and overly busy figures, once an object leaves the sensor's field of view, it will disappear off of the figure. However, the tracking algorithm will still keep a record of the number of target colored pixels for that frame capture as well as the classification of that object.

3.3 Summary

Creating an ATR algorithm to complete the wide area search is the most critical step in the successful development of the surrogate system. This process outlines how the sensor will see the target and what characteristic will be used to make targeting decisions. Further, it lays out the process that the software must follow to scan the search area and classify as well as report any object encounters. Although, the size, shape, and color of the objects are important, the software used with the sensor is the heart of the autonomous target recognition because it must process sensor data and make the ultimate target or no target decision.

Chapter 4 will combine the concepts from Chapter 1, the hardware from Chapter 2, and the ATR algorithm development from Chapter 3 to run surrogate vehicle experiments to build ROC curves as well as complete a dynamic scaling analysis.

4. Surrogate Vehicle ROC Curve Development and Dynamic Scaling Analysis

4.1 Surrogate Vehicle Search System Initialization

Each time the surrogate vehicle search system is used to collect data, all of its component must be initialized. To complete this process, each of the system components must be powered on and checked to ensure they are communicating properly with one another. Below is a short description of each step in the initialization process.

System Initialization Process

1. Ensure that the laptop has the Kestrel Autopilot's Commbox plugged into the bottom USB port and the ground station Aerocomm wireless transceiver is plugged into the top USB port. With both peripherals plugged in, open the Virtual Cockpit Software and "tracker.m" script in Matlab (provided in Appendix B.2).

2. Power on Commbox and check voltage reading in message window (Voltage should be greater than 10 V and in green font)

3. Power on Radio Transmitter and verify the "RC" box is checked in the panel above the message center in the Virtual Cockpit GUI.

4. Power on the onboard vehicle electronics using the single power switch wired to the Li-Po batteries. When the main power switch is turned on, the Kestrel Autopilot, Aerocomm wireless transceiver, and CMUcam2 should all power on.

5. Check the Tamiya 1/20th scale radio controlled dump truck steering by toggling channel 1 (right joystick on transmitter) left and right. Check throttle by moving left joystick forward and back 2-3 detents (Check throttle only when the vehicle is in an open space where there are no obstructions).

4.2 Sensor Characterization

To experimentally build a ROC curve for the surrogate vehicle, the target must first be characterized using the surrogate vehicle's sensor so that the ATR algorithm can properly classify objects against the background of the area it will be searching. This process consists of determining the color of the objects that should be classified as targets or false targets and then determining the pixel threshold that must be exceeded for the object to be classified as a target.

4.2.1 Target Color Characterization

To determine the target/false target color, a target disk is put within the field of view of the camera with the camera bore angle set to the same angle to be used during the searches. Its RGB minimum and maximum values are read after capturing a frame. The process is repeated at 6 different points within the field of view to capture all variations in the target's color as seen in Figure 22.

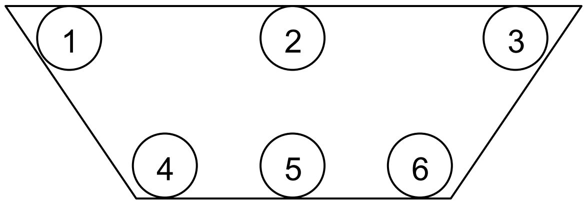


Figure 22. Target Color Characterization Locations

Since the CMUcam2 comes with a GUI that already calculates the minimum and maximum RGB values for pixels within a certain range, it will be utilized to find the target color. It is important to complete this process with the target set out against the background to be used during the search as the RGB ranges of the target color will change against different backgrounds due to changes in the amount of light reflected. The red, green, and blue pixel intensity ranges for the orange objects to be used in this research are shown in Table 7.

Table 7. Target Color Minimum and Maximum RGB Values

	Red		Green		Blue	
Position	Min	Max	Min	Max	Min	Max
1	210	255	97	157	0	46
2	210	255	137	197	0	46
3	210	255	115	175	0	46
4	210	255	92	152	0	46
5	210	255	80	140	0	46
6	210	255	85	145	0	46
Average	210	255	101	161	0	46
Extreme	210	255	80	197	0	46

It should be noted that since the object's color is orange, its highest color intensity will be red. The biggest color intensity variation comes from the green intensity values as they range from 80 to 197 depending on the object's location within the sensor footprint. The blue intensity stays relatively constant regardless of where the object is placed within the frame. It is important to use the whole range of intensity values for all three colors in the search algorithm because the object should be fully tracked regardless of its position within the frame.

4.2.2 Surrogate Vehicle ROC Curve Determination

For this research, multiple ROC curves will be created for comparison. The first will be created by searching an area the size of the sensor footprint with zero velocity with a false target that is 2.5 inches in diameter and the true target is 3.25 inches in diameter. To explore the effect of target size, the second curve will be developed from a false target of the same size with a true target that is 4.75 inches in diameter. For these cases, a target will be placed in each of the six spots in Figure 22 and 100 frames will be captured for each placement. The process is then repeated with the false targets for a total of 1200 frame captures.

In the third case, the surrogate vehicle complete 5 straight line runs across the length of the search area while searching for the 4 true targets and 4 false targets and then 1 true target and 1 false target will be removed to verify the results are repeatable. To ensure the sensor works correctly with the algorithm, all targets and false targets will be distributed along the search path such that there is only object in the sensor's field of view at any one time. The width of the search path will be restricted to the front width of the sensor footprint so each true and false target has a chance for detection.

Once the detection data is collected for both cases, the next step in the analysis is to vary the threshold so that the ROC curve points can be determined. For both cases, the initial P_{TR} and 1-P_{FTR} characterization will come from setting the pixel threshold to be 10% below the maximum number of pixels that the sensor can detect for a false target. The reason to set the initial threshold smaller than the maximum size of the false target is to ensure that the experimental ATR system will give false positives so that that a confusion matrix can be developed.

Theoretically, this value can be calculated from the equations (12) through (22) in sections 3.1.1 and 3.1.2 as shown below for a false target with a 1.25 inch radius (2.5 inch diameter). The Matlab code for this calculation is shown in Appendix B.1.

Object vertical angle from equation (12),

$$\beta = \tan^{-1}\left(\frac{d + L_{obj}}{h}\right) - \alpha = 9.27^{\circ}$$

Vertical pixels on object from equation (13),

$$P_{ObjVert} = \beta * \frac{CameraVerticalPixels}{VFOV} = 43.07$$

Front object horizontal angle from equation (14),

$$\chi_{obj} = 2 * \tan^{-1}(\frac{.5 * L_{obj}}{s_{obj}}) = 10.59^{\circ}$$

Rear object horizontal angle from equation (15),

$$\chi_{b} = 2 * \tan^{-1}(\frac{.5 * L_{obj}}{s_{b}}) = 11.82^{\circ}$$

Front horizontal pixels on object from equation (16),

$$P_{ObjHoriz} = \chi_{t\arg} * \frac{CameraHorizontalPixels}{\theta} = 20.41$$

Rear horizontal pixels on object from equation (17),

$$P_{BHoriz} = \chi_{b} * \frac{CameraHorizontalPixels}{\theta} = 22.78$$

Upper left object bounding pixel x coordinate from equation (18),

$$U_{L}x = \frac{CameraHorizontalPixels - P_{ObjHoriz}}{2} = 33.29$$

Upper left object bounding pixel y coordinate from equation (19),

$$U_{L}y = CameraVerticalPixels - P_{ObjVert} = 99.92$$

Lower right object bounding pixel x coordinate from equation (20),

$$L_{R}x = \frac{CameraHorizontalPixels + P_{BHoriz}}{2} = 54.89$$

Lower right object bounding pixel y coordinate from equation (21),

$$L_{R}y = CameraVerticalPixels = 143$$

Theoretical object pixel area from equation (22),

$$A_{ObjPix} = (L_{R}x - U_{L}x) * (L_{R}y - U_{L}y) = 930.3$$

Experimentally, the simplest way to calculate the maximum number of pixels detected for a false target is to place a false target within the CMUcam2's field of view as shown in Figure 23.

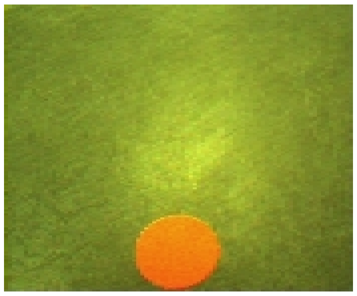


Figure 23. False Target Location for Maximum Pixel Detection

The frame capture in Figure 23 resulted in 800 pixels on the false target as well as very similar centroid and bounding coordinates to those determined theoretically. Table 8 summarizes both the theoretical values and the experimental values for the maximum number of pixels that the sensor can put on a false target. Since the number of pixels is within 15%, the theoretical equations seem to be accurate.

Table 8. Maximum False Target Pixels

	Maximum False Target Pixels	
	Theoretical	Experimental
# of Pixels	930.3	800
x-centroid	43.5	44
y-centroid	121.46	121
x-upper left	33.29	34
y-upper left	99.93	102
x-lower right	54.89	54
y-lower right	143	142

With an experimental value of 800 for the maximum number of false pixels, the initial threshold used to experimentally determine a P_{TR} and $1\text{-}P_{FTR}$ is 720 pixels. Table 9 shows the P_{TR} and 1-P_{FTR} values at the initial threshold for both static cases where the target size was varied and the velocity cases where the number of targets and false targets were varied. P_{TR} was calculated by dividing the total number of true target detections (when a true target was in the footprint) by the total number of true target encounters. 1-PTR was calculated by dividing the total number of false target detections (when a true target was in the footprint) by the total number of true target encounters. P_{FTR} was calculated by dividing the total number of false target detections (when a false target was in the footprint) by the total number of false target encounters. 1-P_{FTR} was calculated by dividing the total number of true target detections (when a false target was in the footprint) by the total number of false target encounters.

Table 9. Initial Threshold Sensor Characterization

	Target Threshold=720 pixels			
	Static 1	Static 2	Velocity 1	Velocity 2
P_{TR}	55.77%	77.78%	80.00%	78.57%
1-P_{FTR}	11.89%	0.00%	10.53%	20.00%
P_{FTR}	88.11%	100.00%	89.47%	80.00%
1-P_{TR}	44.23%	22.22%	20.00%	21.43%

To create the experimental ROC curves for each of the cases, the target thresholds were varied enough to get the (0,0) point and the (1,1) point on the curve. The largest variation occurred with the Static 2 case where the target diameter was 4.75”. It required the threshold to be dropped to 8 pixels to get the (1,1) point and due to the large target size, the threshold needed to be raised to 2560 to get the (0,0) point. The three other scenarios had threshold windows less than this case.

The resulting ROC curves are presented in the next two figures. Figure 24 was created from the first two runs with the sensor in a static position at a 45° sensor bore angle. The first ROC curve was built from data collected for a 3.25” true target and a 2.5” false target. The second ROC curve was built from data collected for a 4.75” true target and 2.5” false target. These two curves demonstrate that as the target size gets bigger while using the same ATR algorithm, the sensor will have better performance. This is expected because *c*, the ROC parameter, normally increases with more pixels on target.

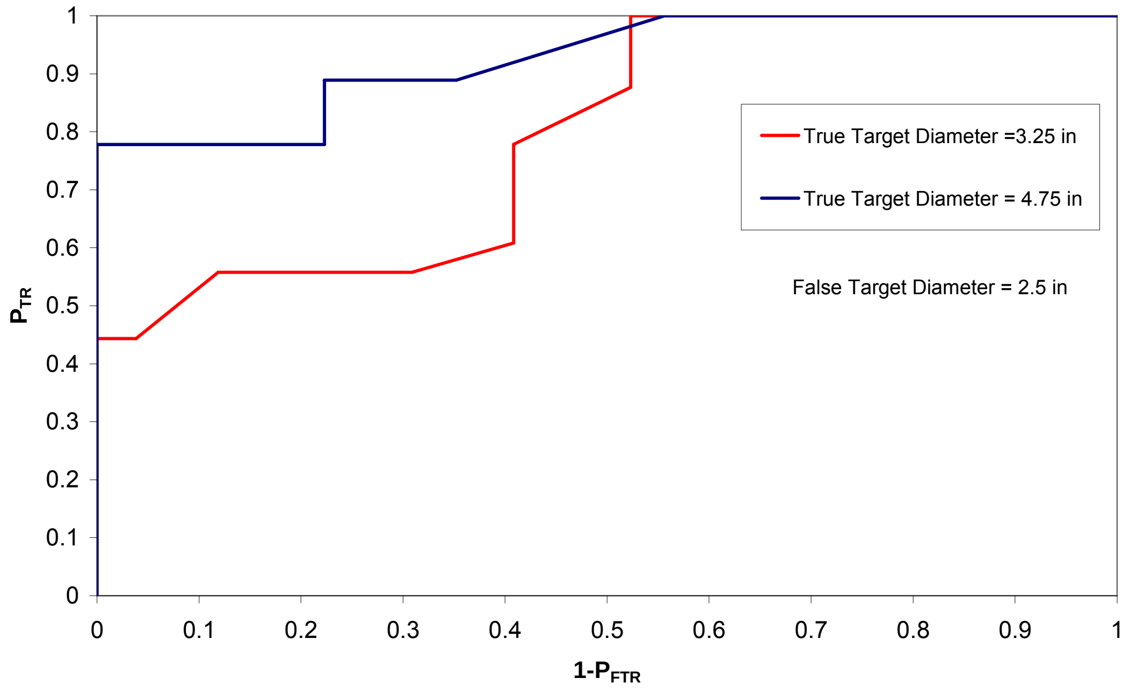


Figure 24. Surrogate Vehicle Static ROC Curves

Figure 25 shows a pair of ROC curves for the surrogate vehicle conducting a straight line search at 0.5 ft/s with 3.5” true targets and 2.5” false targets. The slow velocity was chosen to ensure that the sensor was able to capture the objects in multiple frames. These two curves produced similar P_{TR} results for $1-P_{FTR}$ values between 0 and 0.6 giving confidence that the results are repeatable for the CMUcam2 with the same sensor geometry, target size, and search speed. Also, for reference, Equation 1 was used to characteristic plot a ROC curve with $c = 10$.

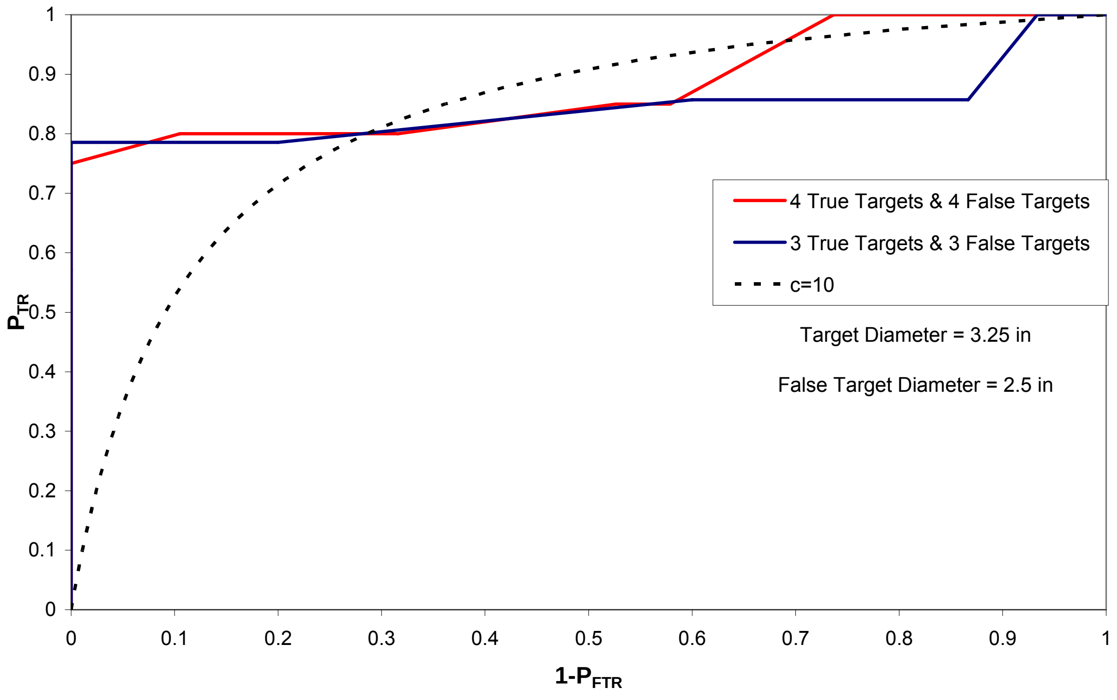


Figure 25. Surrogate Vehicle ROC Curves for Vehicle Velocity = .5 ft/s

To truly see the effects of target size and sensor velocity on the sensor performance, all four ROC curves have been plotted together in Figure 26. When the vehicle is moving at a slow velocity, it seemed to perform slightly better for lower values of 1- P_{FTR} than the static case with 3.25" diameter true targets. However, further right on the ROC curves, the static cases reached a P_{TR} of unity when $1-P_{FTR}$ was less than 0.6 while the velocity case got to a $1-P_{FTR}$ of 0.7 before P_{TR} reached unity. The best sensor performance found during this research seemed to be the static case with 4.75" diameter true targets. As mentioned above, this is to be expected since c is a function of pixels on target and characteristic target length divided by search vehicle velocity.

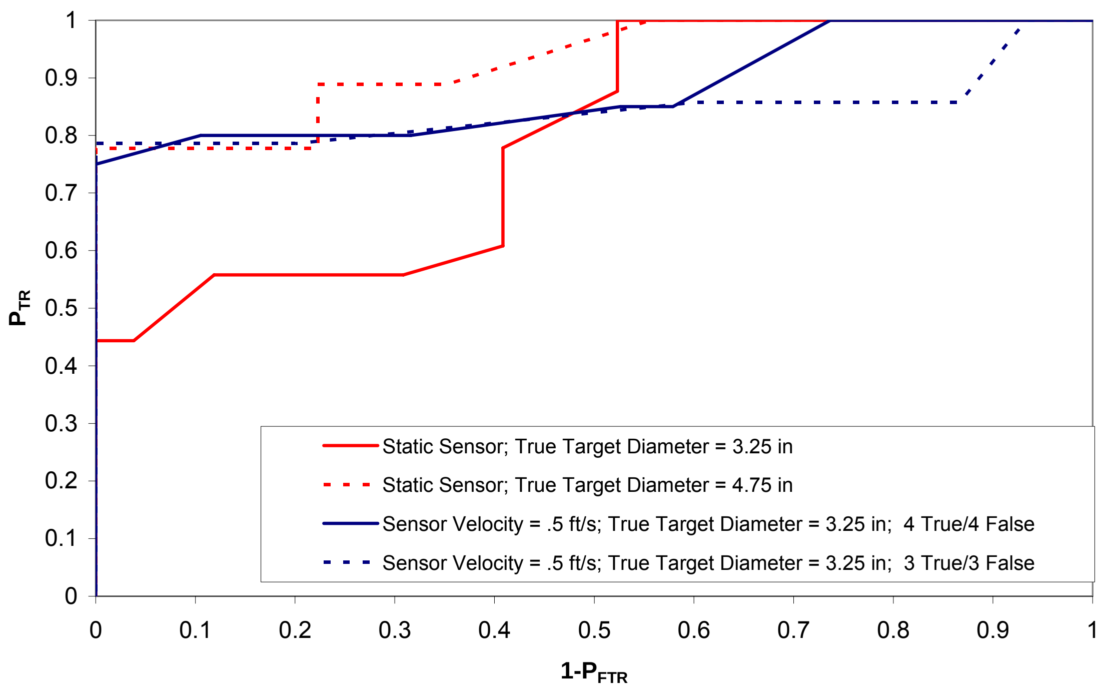


Figure 26. Surrogate Vehicle ROC Curves

4.3 Surrogate Vehicle Dynamic Scaling

4.3.1 Dynamic Scaling Overview

As discussed earlier, the purpose of this research is to build a surrogate vehicle that is dynamically similar to an actual UAV with known sensor performance. To verify the dynamic scaling concepts previously presented, a dynamic scaling analysis will be performed assuming that both the surrogate and nominal vehicle use the same sensor at the same bore angle. Note that the nominal vehicle has a variable sensor rate (i.e. frames per second), but the surrogate was assumed to be a constant 6 frames per second. Furthermore, this assumption enables target size to be scaled between the vehicles. If the sensors were different, then matching pixels on target would be the only parameter that needed to be matched to be able to match sensor performance.

From Chapter 1, the nominal vehicle for this analysis will be the Sig Rascal 110 Radio Controlled aircraft. It will be operating at 100' AGL, 40 kts, with the sensor bore angle of 45 degrees, searching for targets with an 8' characteristic diameter. Two different overlap cases will be examined. The first case will determine the surrogate velocity needed to ensure full search area length coverage, but no overlap. The second case will include frame overlap at least as long as the target's lengthwise dimension.

4.3.2 Case 1: No Frame Overlap

Because this search must exhaustively cover the whole length of search area, the maximum airspeed of the Rascal can be no faster than what is needed to ensure where consecutive frames will abut as shown in Figure 27. While this guarantees 100% coverage of the length of the search area, the search will not cover 100% of the width of the search area due to the triangular dead spots on each of side of the footprint due to a sensor bore angle not equal to 0°. If targets are wholly or partially encompassed in these dead spots, at least some part of them will be missed by the sensor resulting in a false target categorization or no detection.

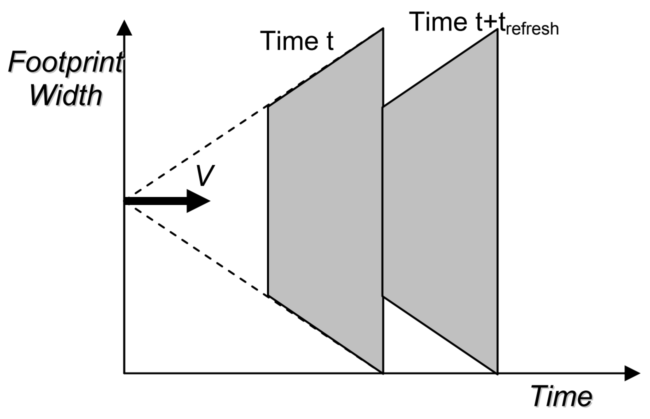


Figure 27. Vehicle Sensor Footprint in Two Consecutive Frames Without Overlap

For searches with no target overlap, Equation 22, taken from (Mike, 2006), can be used to solve for the sensor refresh rate needed to conduct a search at the specified operating speed once the footprint length for the sensor has been established, through the sensor geometry.

$$V_{req-no-overlap} = \frac{z}{t_{refresh}} \tag{23}$$

This same equation can be used to determine the velocity of the surrogate vehicle once its sensor footprint is known, since it has a fixed sensor refresh rate by assumption.

4.3.3 Case 2: Target Lengthwise Overlap

Figure 28 shows consecutive frames when the vehicle is moving slow enough for each frame to overlap part of the previous frame. Notice that when overlap exists in the sensor footprint geometry, each frame will capture a portion of the previous frame near its bottom. As the overlap becomes larger, the dead spot triangles on the outer edges of the footprint will become smaller. Further, making the overlap at least as long as the target will ensure that the target's lengthwise dimension is wholly captured in at least one frame.

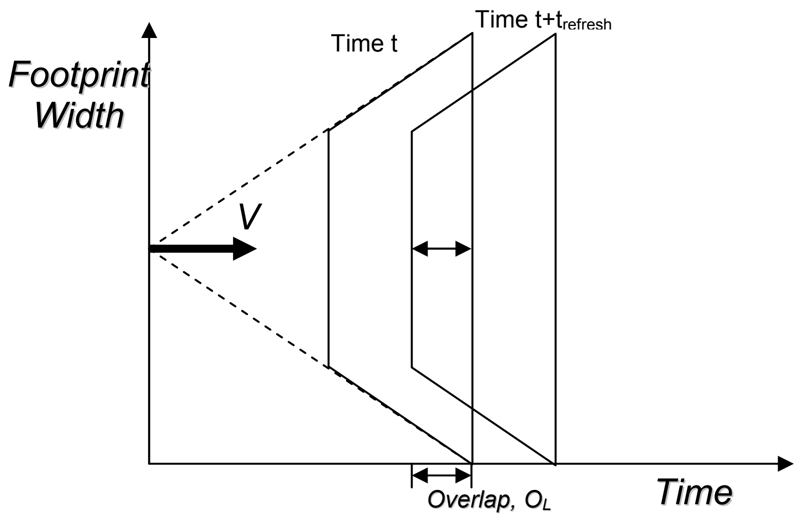


Figure 28: Vehicle Sensor Footprint in Two Consecutive Frames With Overlap

For searches with overlap, the maximum velocity equation is a bit more complicated because it contains an overlap factor on the footprint length. This factor is equivalent to the percent of the sensor footprint length that the vehicle will move in the time it takes to process one sensor frame, $t_{refresh}$. Equation 24 below shows the overlap factor and Equation 25 shows the required velocity given this overlap.

$$x_{overlap} = z(1 - \frac{O_L}{z}) \tag{24}$$

$$V_{req-overlap} = \frac{x_{overlap}}{t_{refresh}} \tag{25}$$

4.3.4 Π₈ Development: Search Vehicle Velocity Frame Overlap Factor

In order to maintain dynamic similarity between the nominal vehicle and the surrogate, the frame overlap must be accounted for in the scaling process. Thus, given the operating velocity of the nominal vehicle and its frame overlap (or equivalently, its sensor refresh rate), the surrogate must be scaled to have an equivalent frame overlap. Therefore, another pi group (Π_8) is developed to account for frame overlap and is shown in Table 10. Then, given the geometry and sensor refresh rate of the surrogate, the required search velocity of the surrogate can be obtained from Π_8.

Table 10. Search Velocity Frame Overlap Factor, Π_8

Variable (units)	Pi Group #/Ratio
V (L/T)	$\Pi_8 = 1-(Vt_{refresh}/z)$

4.3.5 Pi Group and Surrogate Vehicle Dynamics Calculations

Upon developing Π_8, the nominal vehicle's given operating conditions can be scaled to an operating condition for the surrogate vehicle that will be dynamically similar and hopefully within the surrogate vehicles operating range. Table 11 summarizes the values of seven of the eight pi groups for the 100% overlap case with the Sig Rascal 110 using the equations in the far right column of Table 4. The no overlap case has equivalent values for Π_1- Π_6, but Π_7 and Π_8 will go to zero because there will be no frame overlap. The ROC Curve factor, Π_4, presented in (Abeygoonewardene, 2006), was omitted for this analysis, which was possible due to the fact that the same sensor and ATR algorithm are used on the normal and surrogate vehicle.

Table 11. Pi Group Scaling Factors for Sig Rascal 110

Pi Group	Name	Ratio	Value
1	Normalized Elevation Field of Regard	*z/d*	2.098423447
2	Normalized Azmuthal Field of Regard	*w/d*	2.674130978
3	Normalized Vehicle Turn Capability	*d/r*	0.152548604
4	ROC Curve Factor	*V*c/(kc_bar)*	N/A
5	Pixel Density Range	ρ_T*d^2	2,828
6	Target Detection Size	D_t/d	0.140818713
7	Footprint Overlap	O_L/d	0.140818713
8	Velocity Factor	1-(V*$t_{refresh}$/z)	0.06710691

Once the values of the pi groups are known for a specified nominal case, the operating conditions for the surrogate search vehicle can be directly calculated. These calculations start with determining its sensor dead band by multiplying Π_3 by the vehicle's minimum turn radius (assuming that minimum turn radius is independent of velocity). A simple turn radius test at low speeds (< 1 mph) showed the minimum turn radius to be roughly .996 meters.

From Π_3, the surrogate vehicle sensor dead band,

$$d = \Pi_3 * r = .152 \text{ meters}$$

As seen above in Table 11, the surrogate vehicle's footprint length, front footprint width, pixel density, true target, frame overlap, and velocity can be determined by multiplying the appropriate pi group by the surrogate sensor's dead band as shown in the following calculations for 100% target overlap.

From Π_1, the surrogate vehicle footprint length,

$$z = \Pi_1 * d = .319 \text{ meters}$$

From Π_2, the surrogate vehicle footprint length,

$$w_f = \Pi_2 * d = .406 \text{ meters}$$

From Π_5, the surrogate vehicle sensor pixel density,

$$\rho_T = \frac{\Pi_5}{d^2} = 122{,}256 \text{ pixels/meter}^2$$

From Π_6, the surrogate vehicle desired true target characteristic diameter,

$$D_T = \Pi_6 * d = .021 \text{ meters}$$

From Π_7, the surrogate vehicle sensor frame overlap length,

$$O_L = \Pi_7 * d = .021 \text{ meters}$$

From Π_8, the surrogate vehicle velocity needed for scaled frame overlap,

$$V = z * \frac{(1 - \Pi_8)}{t_{refresh}} = 1.79 \text{ meters/second}$$

Table 12 and Table 13 summarize the parameters for the nominal vehicle and the corresponding surrogate vehicle parameters required for the surrogate to perform a dynamically similar search. Table 12 shows the results when no frame overlap is required, while Table 13 mimics the above calculations to show both vehicles operating conditions when 100% target lengthwise frame overlap is needed.

Table 12. Vehicle Dynamics (No Overlap)

Vehicle/Sensor Parameter	Sig Rascal -Nom	Truck -Surg
Velocity (V), m/s	20.57776977	1.914811365
Needed g limit, m/s^2	1.07	1.06809794
Turn Radius (r), m	113.5105791	0.99695
Normal Operating Altitude of the Sensor (h), m	30.48	0.267702238
Frame Overlap (O_L), m	0	0
Pixel Density (ρ), pixels/m^2	9	122,256
Dead Range of Sensor (d), m	17.3158804	0.152083331
Footprint Front Width (w), m	46.30493219	0.406690747
Footprint Length (z), m	36.33604943	0.319135227
Swath Angle (θ), degrees	45.13401816	45.13401816
Vertical Field of View (VFOV), degrees	30.7976395	30.7976395
Sensor Bore Angle (χ), Degrees	45	45
Desired Target Characteristic Diameter (D_t), m	2.4384	0.021416179
Camera Refresh Rate, s	1.765791426	0.166666667

Table 13. Vehicle Dynamics (100% Target Overlap)

Vehicle/Sensor Parameter	Sig Rascal -Nom	Truck -Surg
Velocity (V), m/s	20.57776977	1.786314291
Needed g limit, m/s^2	1.07	1.051982707
Turn Radius (r), m	113.5105791	0.99695
Normal Operating Altitude of the Sensor (h), m	30.48	0.267702238
Frame Overlap (O_L), m	2.4384	0.021416179
Pixel Density (ρ), pixels/m^2	9	122,256
Dead Range of Sensor (d), m	17.3158804	0.152083331
Footprint Front Width (w), m	46.30493219	0.406690747
Footprint Length (z), m	36.33604943	0.319135227
Swath Angle (θ), degrees	45.13401816	45.13401816
Vertical Field of View (VFOV), degrees	30.7976395	30.7976395
Sensor Bore Angle (χ), Degrees	45	45
Desired Target Characteristic Diameter (D_t), m	2.4384	0.021416179
Camera Refresh Rate, s	1.64729462	0.166666667

Since the surrogate's sensor frame rate is fixed at 6 frames per second, its velocity changes from 4.28 miles per hour in the no overlap case to 3.99 miles per hour in the 100% target overlap case. Both of these cases have reasonable surrogate velocities that can be demonstrated in later research as well as actually flying the nominal vehicle to fully validate this dynamic scaling model. Appendix A shows different variations of surrogate and nominal vehicle parameters as nominal operating conditions change to give an idea of how the sensor geometry changes with increasing and decreasing altitude and velocity of the nominal vehicle.

4.4 Summary

Chapter 4 brought together ATR, dynamic scaling, and the actual surrogate vehicle to show that it is not only possible to experimentally characterize the performance a sensor, but it is also possible to adjust the performance by changing search parameters such as target size and search vehicle velocity. Theoretical calculations showed that object pixel information can be accurately predicted by knowing the sensor bore angle, object size, and its position within the footprint. The dynamic scaling analysis also demonstrated that the surrogate vehicle developed for this research should dynamically scale to existing ANT Center UAVs flown with an identical sensor at nominal operating conditions. Using the same dynamic scaling analysis, any other UAV that can match the nominal conditions (altitude, airspeed, and sensor) will also be dynamically similar to the surrogate vehicle.

5. Conclusion and Recommendations

5.1 Summary

The research presented in this thesis can be broken down into four categories. The first is the development of an experimental platform that will meet the needs of future AFIT autonomous wide area search studies to include cooperative autonomous wide area search studies. The second is the development of an autonomous target recognition algorithm (ATR), incorporating sensor geometry, sensor characteristics, and target sizing to build ROC curves for a given operating condition. Multiple ROC curves were developed for the sensor to show the effects of different variables on ATR performance. The third category of research is the further investigation into the dynamic scaling of wide area search vehicles, based on the work of Captain Jeevani Abeygoonewardene. The last category is the development of further sensor geometry calculations to predict the maximum number of pixels a sensor will return with an object horizontally centered at the bottom of its field of view.

5.1.1 Experimental Platform Development

This research successfully developed a surrogate vehicle test bed that can be used to conduct autonomous single UAV experiments as well as multiple UAV cooperative control experiments. Because the vehicle was built in such a manner that the UAV autopilot, wireless transceiver, and camera are mounted together on the sensor deck, this vehicle electronics package can be installed on any radio controlled vehicle with a steering and throttle servo. Furthermore, since the autopilot was intended for aircraft, it will be possible to transition this to a UAV surrogate.

5.1.2 ATR Algorithm and ROC Curve Development

An ATR algorithm was developed to search, classify, and report targets during an experimental wide area search. This algorithm used the object size and color against the search background to determine if it was a target or false target based on the threshold set by the size of the known false targets. Multiple runs of a static search and moving search collected data to build four experimental ROC curves for the ATR system developed at different operating conditions. While these curves were not nearly as smooth as the traditional ROC curves seen in Chapter 1, they confirmed the same general trend that as P_{TR} increases, $1-P_{FTR}$ also increases. Also, the plots validated theoretical results claiming that as target size and pixels on target increase, the ATR performance improves. This finding demonstrated that the algorithm used in this thesis, although not refined, is a good starting point for future wide area search studies. Hopefully, with more research and data collection, the experimental ROC curves will eventually become smooth enough to better fit theoretical curves with specific *c* values using Equation1.

5.1.3 Dynamic Scaling

The dynamic scaling analysis showed that it is feasible to use the Tamiya 1/20th scale RC Dump Truck as a surrogate vehicle to the Sig Rascal or any other UAV with similar operating conditions. The truck can conduct a 100% target overlap search at 3.99 miles per hour with 0.84" diameter targets that scales to a Sig Rascal flying at 40 knots, 100' AGL searching for 8' diameter targets. The only change to be made to the Rascal would be to manually adjust the refresh rate on its sensor so that it can maintain the specified 40 knot airspeed regardless of the required frame overlap percentage.

A search vehicle velocity overlap pi group was developed to ensure that the surrogate vehicle and nominal vehicle both have footprints that overlap the same percentage of the target length. While this research used surrogate vehicle velocity as the control to change the percentage of frame overlap, similar to the nominal vehicle, the surrogate vehicle's velocity could be fixed and its sensor frame refresh rate could be altered if the sensor had this capability.

5.1.4 Further Sensor Geometry Development

While the majority of this research focused on building and completing preliminary testing of an experimental platform, some theoretical concepts were also investigated. The sensor footprint geometry was examined for the case when the sensor is not normal to the surface it is viewing. This trapezoidal footprint required both a front and back footprint width to be calculated and also skewed the shape of each sensor pixels. Also, the determination of the maximum number of pixels of an object in the frame turned into a very cumbersome process of finding angles, pixel densities, and coordinates. The fact that these theoretical calculations were able to accurately predict experimental results demonstrated that the sensor system geometry used in this research is well modeled.

5.2 Recommendations for Future Research

Since this research resulted in both a wide area search surrogate platform and a dynamic scaling analysis, future students have several possibilities when continuing wide area search studies. However, the first recommendation is to run several more experiments with the existing surrogate wide area search vehicle to show that the ROC curves presented in this research accurately depict the performance of the ATR system developed for the given operating conditions. Upon validation of those curves, the experiments can be taken one step further to determine the probability of target attack (P_{TA}) as a function of P_{TR} to determine how well it matches up with wide area search simulations.

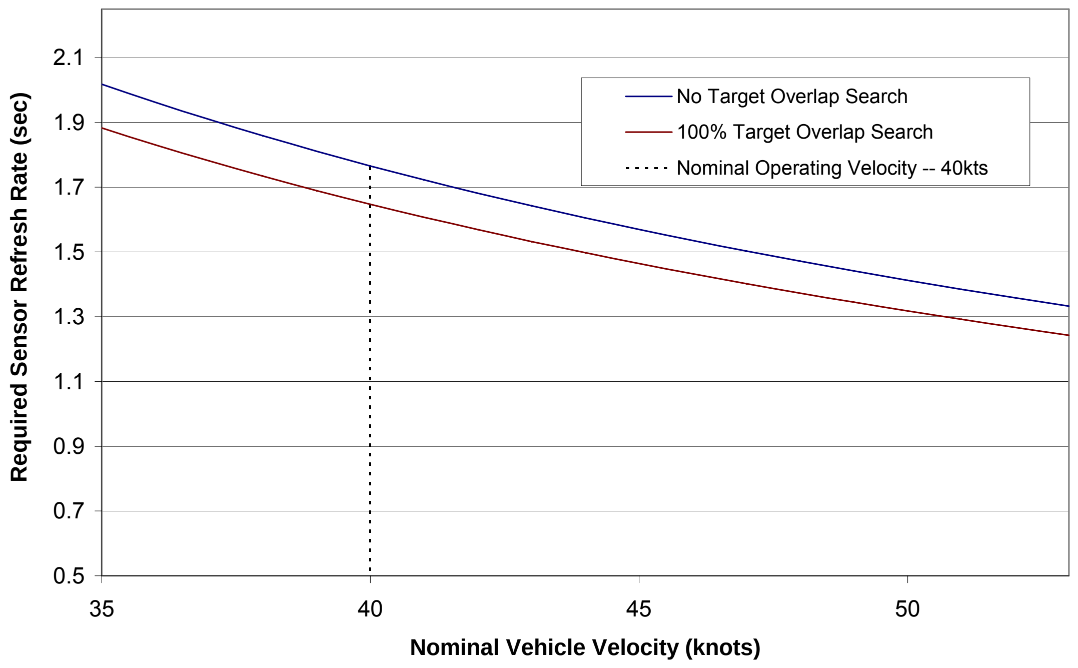
A second recommendation is to examine how the performance of the sensor changes when using object color to set the target/false target threshold instead of object size. In cases where the targets need to be very small to get the needed sensor performance to scale to the nominal vehicle, using different colored targets and false targets might provide an easier route to matching sensor performance. The target/false target threshold would be set by changing the red, green, and blue pixel intensity values in the sensor tracking script and running searches with targets and false targets of equal size, but different closely matching colors. As the pixel intensity values are changed, the P_{TR} and $1\text{-}P_{FTR}$ values will change so that ROC curves can be developed.

Another useful recommendation is to implement technology to detect the surrogate sensor's bore angle, pan angle, and height off of the surface to be searched. These measurements would ensure that the sensor's actual experimental operating conditions always match the conditions used in any theoretical calculations, simulations, or dynamic scaling analyses. Specifically, it is necessary to match given nominal Sig Rascal operating conditions to the surrogate ground vehicle's actual operating conditions through dynamic scaling with a high level of accuracy to make a dynamically similarity claim. With this technology, ROC curves can be developed for both the surrogate vehicle and the nominal Sig Rascal using the theoretically calculated operating velocities, altitudes, and target sizes. With identical sensors and sensor bore angles, the correlation between these two sets of curves will give future researchers additional insight into the dynamic scaling techniques discussed in this research.

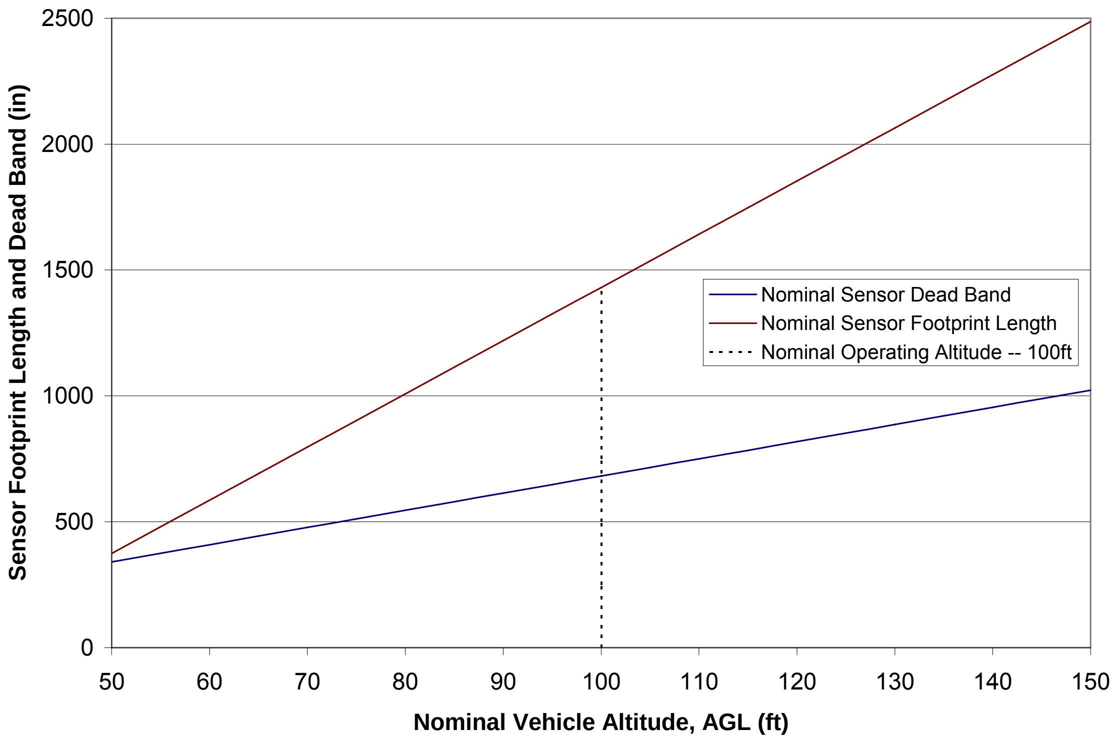
Integrating target classification feedback into the autopilot using a steering correction angle will enable the surrogate vehicle to engage a target by navigating towards it. Upon driving over the target, another steering correction should be given to the vehicle redirecting it back in the original search direction parallel to the original search path. This implementation will experimentally test Capt Abeygoonwardene's wide area search simulation (Abeygoonewardene, 2006) so that the results for a similar target/false target field can be compared and contrasted.

Appendix A. Dynamic Scaling Variation of Parameters

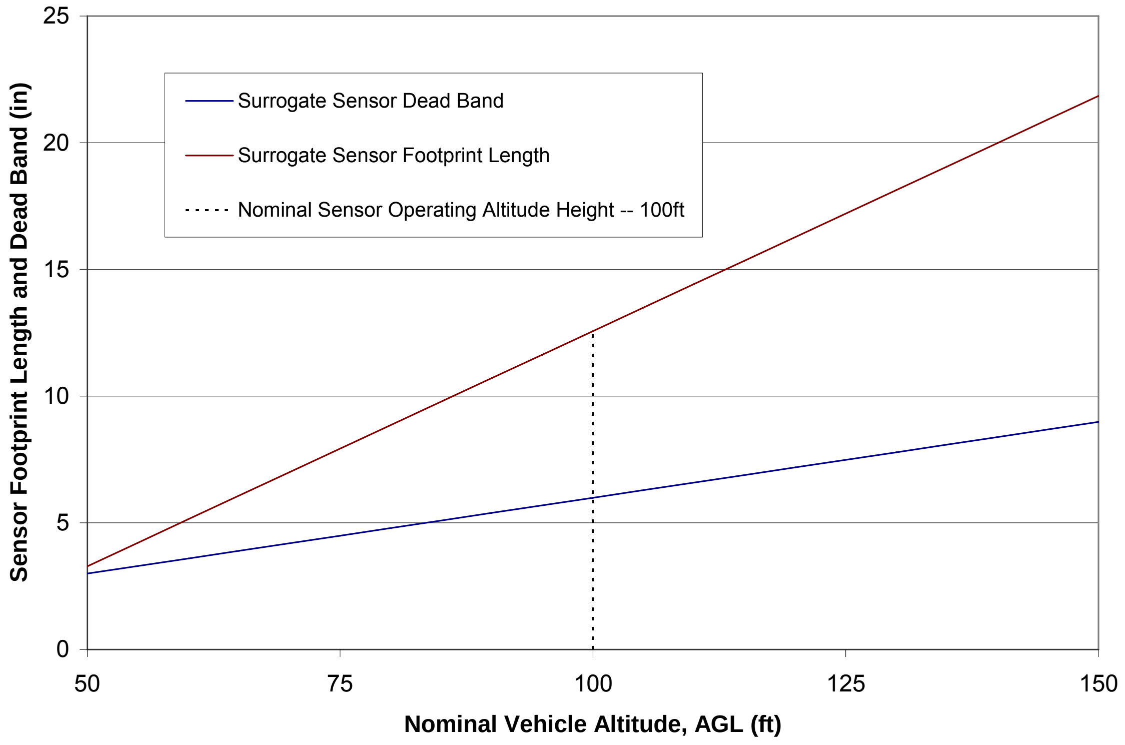
A.1 Nominal Vehicle Required Sensor Refresh Rate as a Function of Velocity



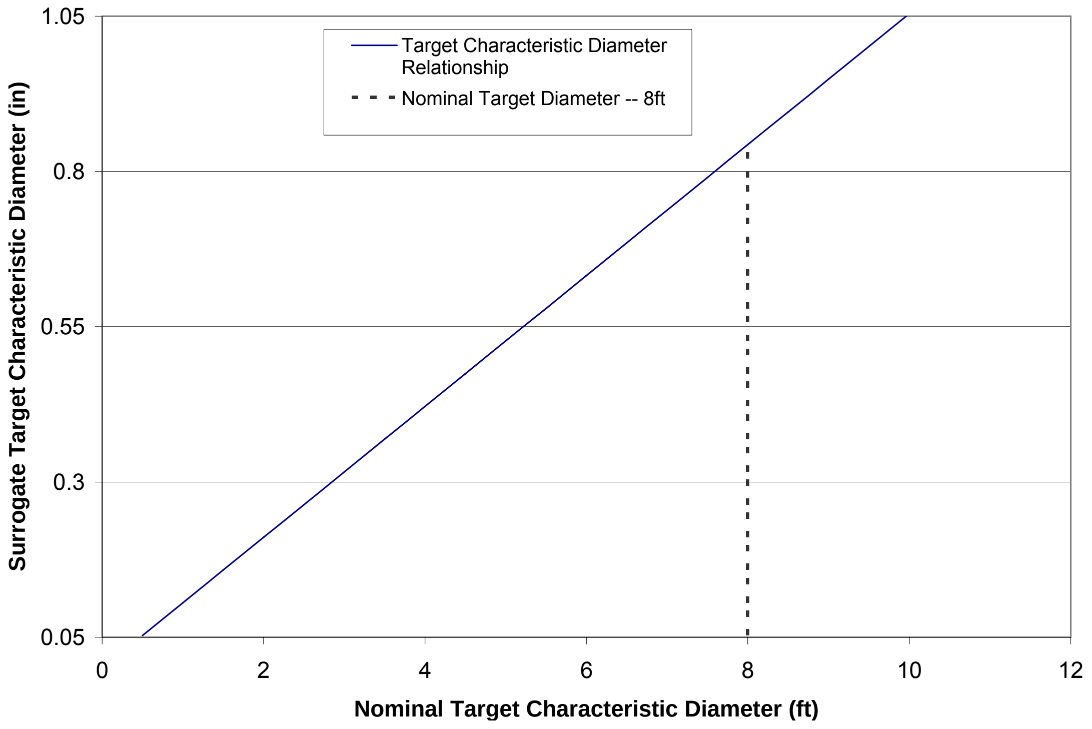
A.2 Nominal Vehicle Footprint Size as a Function of Altitude (AGL)



A.3 Surrogate Vehicle Footprint Size as a Function of Altitude (AGL)



A.4 Surrogate Vehicle Target Characteristic Diameter as Nominal Vehicle Target Characteristic Diameter



Appendix B. Matlab Code

B.1 False Target Maximum Pixel Predictor Code

```
% Capt Justin Rufa
% ENY Thesis Winter 2007 False Target Max Pixel Predictor
clc; clear all; close all;

% Camera's Vertical Field of View Properties
pixels_length=143; % Camera's vertical lines of resolution in pixels
VFOV=deg2rad(30.79); % Camera's vertical field of view in degrees
vert_pixels_per_degree=pixels_length/rad2deg(VFOV); %Camera's vertical
pixels per degree
bore_angle=deg2rad(45); % Camera's Bore Angle (Vertical Centerline) in
degrees
camera_height=10.5; % Camera's height off ground in inches
depression_angle=deg2rad(90)-bore_angle-.5*VFOV; % Camera's depression
angle measured from horizontal in radians
slant_angle=bore_angle-.5*VFOV; %Camera's slant angle measured from
vertical in radians
dead_band=camera_height*tan(slant_angle); % Camera's deadband in inches
footprint_length=camera_height/tan(depression_angle)-dead_band; %Camera's
footprint length in inches
slantback=sqrt(camera_height^2+dead_band^2); %Camera's Slant Length to
back of footprint in inches
slantfront=sqrt(camera_height^2+(dead_band+footprint_length)^2); %Camera's
Slant Length to front of footprint in inches

% Camera's Horizontal Field of View Properties
pixels_width=87; % Camera's horizontal lines of resolution in pixels
theta=deg2rad(45.13); % Camera's horizontal field of view in degrees
horiz_pixels_per_degree=pixels_width/rad2deg(theta); %Camera's horizontal
pixels per degree
footprint_backwidth=2*slantback*atan(.5*theta); % Camera's rear footprint width
in inches
footprint_frontwidth=2*slantfront*atan(.5*theta); % Camera's front footprint width
in inches
rho_f=footprint_frontwidth/pixels_width; % Camera's front footprint width inches
per pixel
rho_b=footprint_backwidth/pixels_width; % Camera's rear footprint width inches
per pixel

% Circular Target Properties
target_radius=1.25; % Target Circular Radius in inches
    % Vertical Properties
    slanttarg=sqrt(camera_height^2+(dead_band+2*target_radius)^2); % Target's
Slant Length from back of footprint to front ot target in inches
    target_vertangle=atan((2*target_radius+dead_band)/camera_height)-
slant_angle; % Target's vertical angle in radians
    target_vert_pixels=rad2deg(target_vertangle)*vert_pixels_per_degree; %
Target's vertical pixels
    uly=143-target_vert_pixels; % Target's upper left y coordinate
    lry=143; % Target's lower right y coordintate
    % Horizontal Properties
    target_backangle=2*atan(target_radius/slantback); % Target's horizontal rear
angle in radians
    target_targangle=2*atan(target_radius/slanttarg); % Target's horizontal front
angle in radians
    target_backhoriz_pixels=rad2deg(target_backangle)*horiz_pixels_per_degree;
% Target's rear width in pixels
    target_fronthoriz_pixels=rad2deg(target_targangle)*horiz_pixels_per_degree;
% Target's front width in pixels
    ulx=43.5-target_fronthoriz_pixels/2; % Target's upper left x coordinate
    lrx=43.5+target_backhoriz_pixels/2; % Target's lower right coordinate

Area=(ulx-lrx)*(uly-lry); % Target's Predicated Number of Pixels
```

B.2 CMUcam2 Matlab Search and Classify Algorithm

```
%#################################################################
%#########--MATLAB to CMUcam2 Matlab Tracking Script-###############
%#################--by Lee von Kraus--########################
%#############--modified by Capt Justin Rufa--##############
%#################################################################
%Note #1:
%If using the CMUcam java GUI to grab a frame and find the # values for a color
you want to track,
%notice that the color #s (mins and maxs) are NOT in the same order as they
are in the TC command!
%Note #2:
%Make sure to turn on pole mode
%('PM 1') this way, you're constantly getting up to date data, and not some stuff
from the buffer.
clc; clear all; close all;

% Set up CMUcam2 Serial Port
ser=serial('COM6'); % Specify COM Port
set(ser, 'BaudRate',115200, 'DataBits', 8,'Parity','none','StopBits',1,
'FlowControl','none',...
'Terminator', 'CR','TimeOut', .5); %Set up serial port properties

% Initialize Serial Port Connection
fopen(ser);                     % Open Serial Port
fprintf(ser, '%s\r', 'RM 2');      % Turn Off ACKs and NCKs
fprintf(ser, '%s\r', 'PM 1');      % Turn Polling Mode On

% Create Camera Field of View Plot
errorInd=[];
hold on
xlim([1 87]);
ylim([1 143]);
set(gca,'YDir','reverse')
title('AFWASTER Field of View');
xlabel('Field of View Width (pixels)');
ylabel('Field of View Length (pixels)');

plot(43.5*ones(143,1), linspace(1,143,143), '--k'); % Plot Vertical Cross hairs
plot(linspace(1,87,87),71.5*ones(87,1),  '--k');    % Plot Horizontal Cross haris
pause(eps);
searches=125;        % Specify # of frames for the CMUcam 2 to capture
threshold=800*.9;   % Specify target threshold # of pixels
list=[];
```

```
list2=[];
list3=[];
list4=[];
list5=[];
list6=[];
tic     % Start Search Clock

% BEGIN SEARCH
for i=1:searches

          fprintf(ser, '%s\r', 'TC 210 255 80 197 0 46');   % Specify Target Color

         cam_data= fscanf(ser,'%*s %d %d %d %d %d %d %d %d')';
         if length(cam_data)== 8
             A(i,1:8)=cam_data;

            % Add Target Area to Tracking Data
            A(i,9)=(A(i,5)-A(i,3)).*(A(i,6)-A(i,4));

         else
             errorInd(end+1)=i;
             A(i,:)=[1 0 0 0 0 0 0 0 1];
         end

   if A(i,1) & A(i,2) > 0
% Plot Target Centroid
    if A(i,9) >= threshold
    list(end +1)=plot(A(i,1), A(i,2), '*r');
    %     % Plot Vector to Center of Target
    list2(end +1)=plot(linspace(43.5,A(i,1),10),linspace(143, A(i,2),10), '--r'); %
From Camera
%     theta=atan(((A(i,1)-3)-43)/(143-A(i,2)))*180/pi
    else A(i,9) < threshold
    list(end+1)=plot(A(i,1), A(i,2), '*g');
    end
% % Plot Target Bounds
    list3(end+1)=plot(linspace(A(i,5),A(i,5)-5,5), (A(i,6))*ones(5,1), 'b'); % Upper
Left Horizontal
    list4(end+1)=plot(ones(5,1)*(A(i,5)), linspace(A(i,6),A(i,6)-5,5), 'b'); % Upper
Left Vertical
    list5(end+1)=plot(linspace(A(i,3),A(i,3)+5,5), (A(i,4))*ones(5,1), 'b'); % Lower
Right Horizontal
    list6(end+1)=plot(ones(5,1)*(A(i,3)), linspace(A(i,4),A(i,4)+5,5), 'b'); % Lower
Right Vertical
```

```
  else
     if ~ isempty(list)| ~ isempty(list2)
        delete(list);
        delete(list2);
        delete(list3);
        delete(list4);
        delete(list5);
        delete(list6);
     end
     list=[];
     list2=[];
     list3=[];
     list4=[];
     list5=[];
     list6=[];

  end
  pause(eps)

   end
% END SEARCH

   fclose(ser);    % Close Serial Port
Target_Reports=length(find(A(:,9)>threshold))   % Report # of Targets Identified
a=toc;
FPS=searches/a  % Report Search Frame per Second Rate
%%%%%% END TRACKING SCRIPT%%%%%%
```

Appendix C. Steering Control Calculations

C. 1 Calculation of Object Centroid Angle, ψ, for Steering Correction

As mentioned in Chapter 3, the search algorithm has the ability to determine the object's centroid angle for steering correction. With this angle, future research can implement commands to steer the vehicle directly towards an object that is classified as a target. The process to calculate this angle is laid out in detail below.

Similar to determining the geometry of the object area, this calculation involves both vertical and horizontal distance calculations shown in Figure 29.

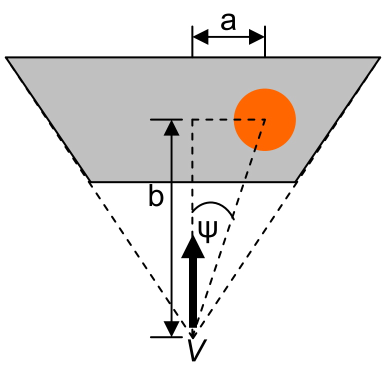


Figure 29. Object Centroid Angle Geometry

The first calculation determines the vertical angle between the centroid of the object and the rear edge of the footprint, $\beta_{centroid}$. To calculate this angle, the vertical pixel distance between the rear edge of the frame and the object centroid must be determined, given the centroid's x and y coordinates in pixels. Both calculations are shown below in Equations 25 and 26.

$$Y_{ObjPix} = CameraVerticalPixels - Y_{Centroid} \tag{26}$$

$$\beta_{centroid} = Y_{ObjPix}\left(\frac{CameraVerticalPixels}{VFOV}\right)^{-1} \tag{27}$$

Once $\beta_{centroid}$ is known, it can be substituted into Equation 27 to solve for the vertical distance between the centroid of the object and the point mass representing the front edge search vehicle, *b*. Note that *d* was previously determined by the geometry of the camera.

$$b = Y_{ObjIn} + d = h\tan^{-1}\left(\alpha + \beta_{centroid}\right) \tag{28}$$

To determine the horizontal distance from the object centroid to the vertical centerline, *a*, a relationship between the pixel density per horizontal line and distance from the rear of the frame must be determined. Since the number of pixels in reach row is known and the width of the rear edge and front edge of the frame are known, Equation 28 gives the pixel density value at any point between the rear and front of the frame.

$$\rho_{Yobj} = \frac{w_b}{HorizontalPixels} + \frac{\dfrac{w_f}{HorizontalPixels} - \dfrac{w_b}{HorizontalPixels}}{z} Y_{ObjIn} \tag{29}$$

With the pixel density, the ground distance, *a*, between the vertical centerline of the frame and the centroid of the object is calculated in Equation 29.

$$a = \rho_{Yobj}(X_{cent} - \frac{HorizontalPixels}{2}) \tag{30}$$

Finally, knowing *a* and *b*, the object centroid angle, *ψ*, is given by Equation 30.

$$\psi = \tan^{-1}(\frac{a}{b}) \tag{31}$$

If the centroid of the object is right of vertical centerline, it will result in a positive object centroid angle, while centroids left of the centerline will command a negative object centroid angle to steer the vehicle directly over top of the object.

Bibliography

1. "Aerocomm Instant Wireless Communications." http://aerocomm.com/rf_transceiver_modules/ac4790_mesh-ready_transceiver.htm. 2 February 2007

2. "The CMUcam2 Manual." http://www.cs.cmu.edu/~cmucam2/index.html. 2 February 2007

3. "iCasualties: OIF – Deaths by IED" http://icasualties.org/oif/IED.aspx. 2 February 2007

4. "Procerus Technologies : Kestrel Autopilot." http://procerusuav.com/productsKestrelAutopilot.php. 2 February 2007

5. "Tamiya America Item #58268 | RC Mammoth Dump Truck." http://www.tamiyausa.com/product/item.php?product-id=58268. 1 February 2007

6. Abeygoonewardene, Jeevani I. *Scaling Flight Tests of Unmanned Vehicles*. MS thesis, AFIT/GAE/ENY/06-S01. Graduate School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 2006.

7. Buckingham, E. "On physically similar systems; illustrations of the use of dimensional equations," *Physical Rev.*, 4: 345-376 (June 1914).

8. Gillen, Daniel P. *Cooperative Behavior Schemes For Improving The Effectiveness Of Autonomous Wide Area Search Munitions*. MS thesis. AFIT/GAE/ENY/01M-03. Graduate School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 2001.

9. Jacques, David. R. and Meir. Pachter. *A Theoretical Foundation for Cooperative Search, Classification, and Target Attack.* Norwell Massachusetts: Kluwer Academic Publishers, 2003.

10. Jodeh, Nidal M. *Development Of Autonomous Unmanned Aerial Vehicle Research Platform: Modeling, Simulating, And Flight Testing*. MS thesis. AFIT/GAE/ENY/06M-18. Graduate School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 2006.

11. Kish, Brian A. *Establishment Of A System Operating Characteristic For Autonomous Wide Area Search Vehicles*. Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 2005.

12. Marks Paul. "Airliner flown 'without pilot' in UAV test," *New Scientist Magazine,* (November 2006). 2 February 2007 http://www.newscientisttech.com/article/dn10675-airliner-flown-without-pilot-in-uav-test.html

13. Mike, Low C. M. (2006, *Automatic Targeting Systems For Unmanned Air Vehicle Operations*. BS Thesis, U024997B. Department of Mechanical Engineering, National University of Singapore, 2006. http://dynlab.mpe.nus.edu.sg/mpelsb/aeg/AutoTarget/Thesis.pdf

14. Moses, L. E, D. Shapiro, and B. Littenberg, "Combining Independent Studies of a Diagnostic Test into a Summary ROC curve: Data-analytic Approaches and Some Additional Considerations," *Statistics in Medicine*, 12: 1293-1316, (1993).

15. Rasmussen, Steven J., Jason W. Mitchell, Phillip R. Chandler, Corey J. Schumacher and Austin L. Smith. *Introduction to the MultiUAV Simulation and Its Application to Cooperative Control Research*. AFRL/VACA, Wright-Patterson AFB OH, June 2005.

16. Richards, Marc D, Darrell Whitley, and J. Ross. Beveridge. "Evolving Cooperative Strategies For UAV Teams," *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*. 2005. http://www.cs.mun.ca/~blangdon/biblio/cache/http___www.cs.bham.ac.uk__wbl_biblio_gecco2005_docs_p1721.pdf

17. Vachtsevanos, G., Liang Tang, and Johan Reimann. "An Intelligent Approach To Coordinated Control Of Multiple Unmanned Aerial Vehicles," *Proceedings of the American Helicopter Society 60th Annual Forum*. 2004. http://www.vtol.org/pdf/uav-60.pdf

18. von Kraus, Lee. "Color tracking w/ CMUcam and MATLAB." n. pag. http://www.instructables.com/id/EE0AV8DP75EP286TEW?ALLSTEPS. 1 February 2007.

Vita

Captain Justin Rufa was born in Massena, NY. He graduated from Massena Central High School in 1997 and began his Air Force career upon entering the United States Air Force Academy Preparatory School in July 1997. In June 1998 he entered the United States Air Force Academy and in 2002 he received a commission as a Second Lieutenant in the United States Air Force as well as a Bachelor of Science degree in Aeronautical Engineering.

He arrived at the Oklahoma City Air Logistics Center at Tinker AFB in Oklahoma City for his first assignment where he worked for the Engineering Directorate as a systems engineer. In October 2004 he was reassigned as a structural modifications engineer to the C/KC-135 System Program Office (327th Tanker Sustainment Group). In August 2005, Capt Rufa entered the Graduate School of Engineering and Management, Air Force Institute of Technology. Upon graduation with a Master of Science in Aeronautical Engineering he will be assigned to the Air Force Research Laboratory's Air Vehicles Directorate in the Control Science Division.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 074-0188*

1. REPORT DATE *(DD-MM-YYYY)* 22 Mar 07	**2. REPORT TYPE** **Master's Thesis**	**3. DATES COVERED** *(From – To)* Oct 2005 – Mar 2007

4. TITLE AND SUBTITLE Development of an Experimental Platform for Testing Autonomous UAV Guidance and Control Algorithms	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Rufa, Justin, R., Captain, USAF	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GAE/ENY/07-M20
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/VAC 2130 Eighth St, WPAFB, OH 45433 Dr. Jeffrey Tromp 937.255.3900 AFIT Proposal #2003-120, AFIT JON #05-186	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
With the United States' push towards using unmanned aerial vehicles (UAVs) for more military missions, wide area search theory is being researched to determine the viability of multiple vehicle autonomous searches over the battle area. Previous work includes theoretical development of detection and attack probabilities while taking into account known enemy presence within the search environment. Simulations have been able to transform these theories into code to predict the UAV performance against known numbers of true and false targets. The next step to transitioning these autonomous search algorithms to an operational environment is the experimental testing of these theories through the use of surrogate vehicles, to determine if the guidance and control laws developed can guide the vehicles when operating in search areas with true and false targets. In addition to the challenge of experimental implementation, dynamic scaling must also be considered so that these smaller surrogate vehicles will scale to full size UAVs performing searches in real world scenarios.

This research demonstrates the ability of a given sensor to use a basic ATR algorithm to identify targets in a search area based on its size and color. With this ability, the system's target thresholds can also be altered to mimic real world UAV sensor performance.

15. SUBJECT TERMS
Autonomous wide area search, dynamic scaling, dynamic similarity, autonomous target recognition, hardware integration

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Maj. Paul A. Blue - ENY
REPORT **U**	ABSTRACT **U**	c. THIS PAGE **U**	**UU**	98	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-6565, ext 4714; e-mail: paul.blue@afit.edu

Standard Form 298 (Rev: 8-98)