An Optical Resection Local Positioning System for an Autonomous Agriculture Vehicle

Kevin H. Murray

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science

In

Mechanical Engineering

Alfred L. Wicks, Chair

Kathleen Meehan

John P. Bird

September 28, 2012

Blacksburg, VA

Keywords: resection, optical, positioning system, autonomous agriculture, precision agriculture

ABSTRACT

An Optical Resection Local Positioning System for an Autonomous Agriculture Vehicle

Kevin H. Murray

Obtaining accurate and precise position information is critical in precision and autonomous agriculture. Systems accurate to the centimeter-level are available, but may be prohibitively expensive for relatively small farms and tasks that involve multiple vehicles. Optical resection is proposed as a potentially more cost-effective and scalable positioning system for such cases.

The proposed system involves the placement of optical beacons at known locations throughout the environment and the use of cameras on the vehicle to detect the apparent angles between beacons. The position of the vehicle can be calculated with resection when three or four beacons are identified. In addition, the system provides precise orientation information, so a separate inertial measurement unit is not required.

The system is seen as potentially cost-effective by taking advantage of the precision and low cost of digital image sensors. Whereas the components in other positioning systems tend to be more specialized, the widespread consumer demand for inexpensive and high quality cameras has allowed for billions of dollars of research and development to be spread across billions of image sensors.

ACKNOWLEDGEMENTS

I am grateful to many people for not only their help in bringing this project together, but for making the entire graduate program such a great experience. This past year has been by far the most interesting, exciting, and rewarding experience as a student and as an engineer. I can't overstate how much this has meant to me.

On the academic side, I would foremost like to thank my committee. Without working on a specific contract (and thus with a topic of my own choosing), Dr. Wicks, Dr. Meehan, and Dr. Bird derived little benefit in taking me on, but did anyway. I am very much indebted to them. In particular, I would like to thank Dr. Wicks for being my advisor and teacher. It has been an absolute privilege and pleasure working for you. Like many theses before me, special thanks go to Dr. Bird for the immense amount of help and time he has offered. I had the pleasure of bugging Dr. Bird for help four years ago as an undergrad, and I still remember how excited I was to see him teaching one of my classes on my first day back as a grad student.

I would also like to thank the other students in the mechatronics lab. I had such a great time working with you all and I regret that it was only for one year. I hope we cross paths again. Best of luck to all of you.

On the personal side, I would like to thank my family. This experience would not have happened without the opportunity and unending support provided by my parents, John and Laura Murray. I could not have asked for better parents.

To my fiancée, Jessica, who was willing to spend the first year of our engagement two-hundred and fifty miles apart: thank you. Thank you for the thousands of miles you logged, the pounds of food you cooked for me, and the countless hours spent playing Civilization V with me. I cannot wait to start the rest of our lives together.

Photos by author unless otherwise cited, 2012.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The ways humans have grown food has been continually transformed by technology; from the oxen-powered plough in the 6th millennium B.C. to the mechanized tractors, planters, and combine harvesters of today. The future of agriculture may involve the application of autonomy to every phase of crop production, and perhaps introduce entirely new processes, such as continual monitoring of the health of individual plants.

In 2007, total spending on farm labor reached $26.4 billion in the United States, which was 11% of total farm production expenses, and the third greatest expense next to feed and livestock expenses (together accounting for 36% of expenses) [1]. The potential benefits of autonomous agriculture mirror the benefits autonomy has brought to other fields: reduction of human labor, increased reliability, and the ability to fundamentally change processes that no longer have to account for human involvement.

One of the fundamental requirements of an autonomous agriculture vehicle is for the vehicle to determine its position within the field. While very precise solutions are commercially available, they tend to be very expensive at roughly five to ten thousand dollars per unit, in addition to subscription services or a user-installed base station. This paper proposes optical resection as a potentially more cost-effective positioning technique for certain agriculture settings, particularly small to medium sized farms with hand-picked produce.

The following sections will give a brief overview of the field and description of the proposed system, followed by an outline of the thesis.

1.1 AUTONOMOUS AGRICULTURE OVERVIEW

Research in autonomous agriculture is not a new field of study. In the 1950s and 1960s, researchers explored mechanical means to provide autonomy for farm equipment, such as leader cables to guide driverless tractors [2].

With the advent of information technology and satellite positioning systems in the 1980s, the concept of 'precision agriculture' emerged. Precision agriculture aims to manage farms by observing and responding to data (e.g. moisture, fertilizer, and pesticide levels) collected at a very fine level, down to individual plants. Although autonomy is not the major feature of precision agriculture, many of the concepts of precision agriculture are shared with autonomous agriculture. Perhaps the most significant shared concept is the use of precision localization. With precision localization, farmers can distinguish individual plants based on their location and then collect relevant data for each plant.

Autonomous grain harvesting [3] and planting [4] systems are beginning to appear in the marketplace, but autonomous agriculture is still mostly in the domain of research. Real-world applications are widely expected in the near future, though, as the popularity of research in autonomous agriculture testifies. Research covers every phase of crop production for virtually every type of crop; from planting, watering, and nutrient and pesticide delivery, to harvesting and packaging.

1.2 OPTICAL RESECTION

Resection is a method of determining the position of an unknown point by measuring angles to known points, from the perspective of the unknown point. By measuring the angles between four known point locations, a unique three-dimensional location can be calculated. Figure 1-1 shows the tetrahedron described by these points and angles.

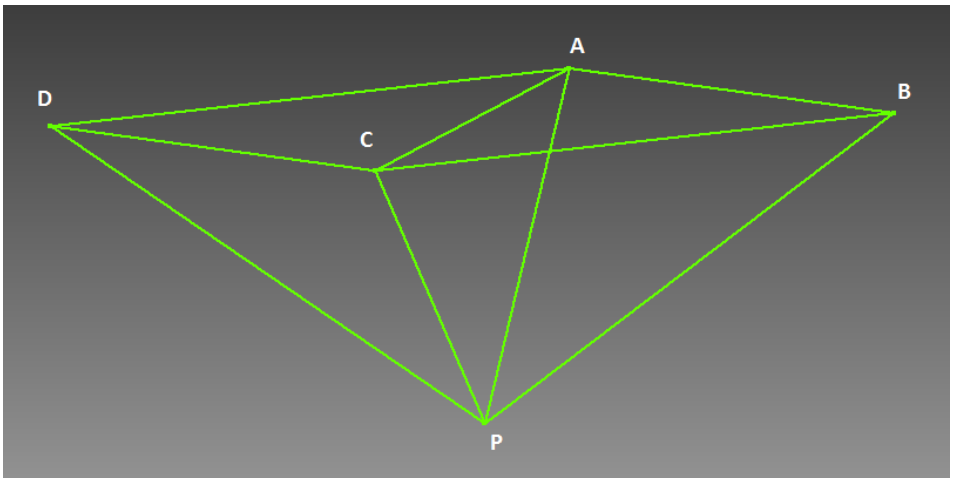


Figure 1-1. Known points A, B, C, D and unknown point P form a hexahedron. The hexahedron is completely defined if the coordinates of points A, B, C, and D and the four angles centered on point P are known. The four known points are not required to be coplanar.

This is the basis for the proposed positioning system. A grid of visible beacons will be placed at known locations throughout the environment; and the system will identify beacons, measure the apparent angles between them, and use resection to calculate its position. As long as at least three beacons are visible to a mobile platform, the position of the platform can be calculated. An additional benefit of this approach is that the absolute angles to the beacons can provide complete and precise orientation measurement (yaw, pitch, and roll) of the platform.

The precision of the position calculation with this setup is dependent on the angular resolution of the image sensors and the spacing between beacons (and thus total number of beacons). Economically, the balance between image sensor resolution and number of beacons is dependent on the total size of the farm and the number of independent mobile vehicles.

1.3 OUTLINE

The order of this thesis matches the sequence that occurred in research. The first stage was to recognize the opportunity of autonomous agriculture and the requirement of a high-precision positioning system, which has been described in the preceding sections of this chapter.

Chapter 2 describes many of the existing positioning systems that were investigated as possible solutions. Ultimately it was decided that development of a new technique (optical resection) should be pursued because of the perceived benefits of high precision and low cost. The factors motivating this decision are described in detail in Chapter 3.

Development of the technique began with the geometric derivations that provide the actual position information. This, as well as a study of the precision that could be obtained, are shown in Chapter 4.

In order to verify the theories developed in Chapter 4, a proof-of-concept system was developed and tested. The design of the system and the experimental results are detailed in Chapters 5 and 6, respectively. The thesis closes with conclusions and areas of future work in Chapter 7.

2 BACKGROUND

The optical resection approach described in this report is motivated by the successes and short-comings of other positioning systems. In addition, the usefulness of the system cannot be fully evaluated without comparing it to other systems. For these reasons, the first section of this report will discuss some of the most common approaches used in agriculture positioning.

The positioning system described in this paper is closely related to a class of indoor positioning systems called 'visible light positioning,' which is unlike most of the systems currently found in agriculture research. A variety of approaches in this category will be briefly discussed in the second section.

Shortly before this project was complete, a group of researchers using an approach very similar to the optical resection method used in this project were discovered. While the central idea is the same, a few important parts of the technique are different, and the scope of this research covers some of the ideas discussed in their future work. Their research will be discussed in detail in the third section of this chapter.

2.1 POSITIONING METHODS CURRENTLY USED IN AGRICULTURE

Accurate positioning is required for practically any autonomous agriculture platform. Systems already exist that largely meet even the most stringent precision requirements, but their generally high costs may not make them feasible for some applications. This section will discuss the benefits, limitations, and costs of some of the more widely-used positioning methods in agriculture commerce and research.

Four of the most common positioning methods used in agriculture include differential global positioning system (differential GPS or DGPS), real time kinematic GPS (RTK GPS), GPS dead-reckoning fusion, and computer vision row detection. DGPS and RTK GPS are well-developed and currently used in agriculture, while GPS dead-reckoning fusion and computer vision row detection are in varying stages of research. Each of these approaches will be described in the following subsections.

2.1.1 DIFFERENTIAL GPS

The United States' global positioning system (GPS) and other global navigation satellite systems (GNSS) enable receivers to estimate their distances (pseudoranges) to multiple satellites. Combining the pseudoranges with the known satellite positions, multilateration is used to determine the position of a receiver. There are a number of error sources in the pseudorange calculation, including ionosphere delay, multipath signals, and receiver clock error. The error in the pseudorange calculation limits positioning accuracy to around ten meters, however, much of the error is spatially and temporally correlated. If the error can be calculated at a certain position and time, then nearby receivers can use this calculated error to correct their position estimates. This is the basis for differential GPS. [5]

Differential GPS involves one or more GPS receivers at fixed, known locations and one or more mobile GPS receivers. The fixed receiver (reference station) can calculate the error in the pseudoranges by comparing the pseudoranges to the known distances between the receiver and satellite positions. The receiver then must have some means of communication to send that information to mobile receiver(s) for their own error correction. This concept is illustrated in Figure 2-1 [6].

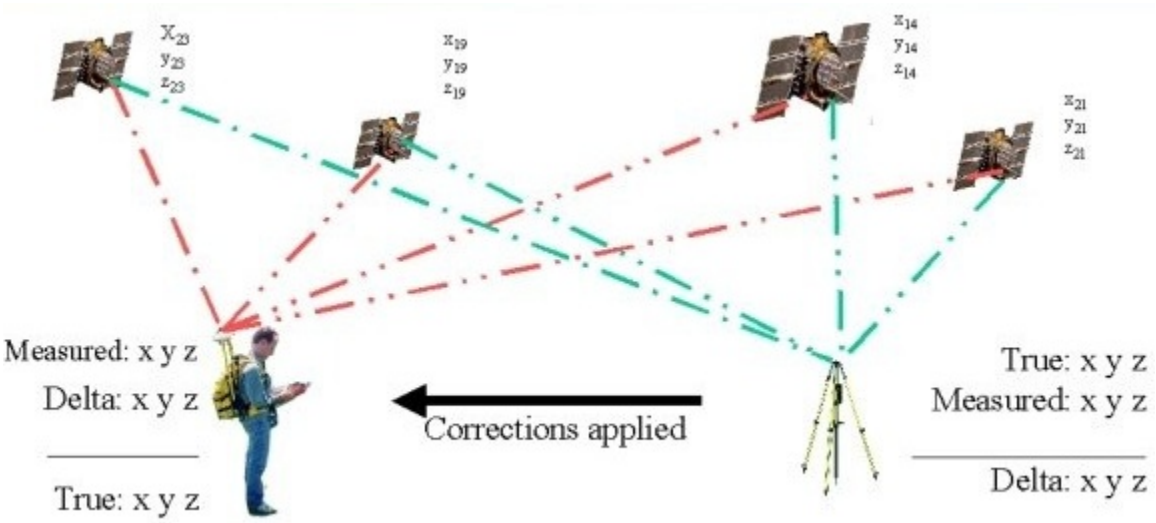


Figure 2-1. Differential GPS.

A differential GPS system can be set up with its own local reference station, or it can make use of reference station networks created by governments or corporations. A local reference station has an advantage of being closer to the mobile receiver, but network stations may use more sophisticated equipment and algorithms.

One network of reference stations is the Wide Area Augmentation System (WAAS), created by the United States Federal Aviation Administration. WAAS has 38 reference stations (each with 3 antennas) covering most of North America and a network of satellites to communicate calculated errors to receivers. According to the National Traffic Safety Board, WAAS has a 95% horizontal accuracy of 0.504-0.922 meters and a 95% vertical accuracy of 0.879-1.373 meters [7]. Access to the WAAS corrections is free of charge and included in many off-the-shelf GPS units.

Another network of reference stations is commercially available through OmniSTAR®, a subsidiary of Trimble Navigation, Ltd. The OmniSTAR® HP and XP services have 95% horizontal accuracies of about 10 cm [8]. The major drawback of these services with this precision is cost: receivers are generally sold for around $5,000 and access to the error corrections requires a yearly service fee of around $1,500 per receiver.

2.1.2 REAL TIME KINEMATIC GPS

Real time kinematic (RTK) satellite navigation is a positioning technique based on GPS carrier phase measurements, and is sometimes called carrier-phase enhancement GPS. RTK systems are a type of differential GPS and thus require a reference station in addition to the mobile receivers. Accuracies as small as one centimeter are possible. References for the description of RTK GPS in this section can be found in [9].

GPS pseudoranges are normally calculated by comparing the received GPS signal with an internally generated copy of the same signal. The internal copy is shifted in time until it aligns with the received signal. That shift in time is the time of flight of the GPS signal, which provides the pseudorange to the satellite when the known speed of light is considered. In general, signals can be aligned to within 1% of their wavelength. For example, since the wavelength of the L1 GPS signal is 293 meters, pseudoranges calculated with the L1 signal have a maximum accuracy of about 2.9 meters (before other errors are introduced). The repeating GPS signals are long enough that there is no risk of mistakenly lining up a signal with a signal repeated at a later time.

RTK systems work in a similar way, but instead of only aligning the GPS signal, the RTK receiver aligns the carrier wave of the GPS signal. The L1 carrier wave has a wavelength of 19 centimeters, which corresponds to a maximum pseudorange accuracy of 1.9 millimeters. However, each cycle of the carrier wave is similar to every other cycle, so it is difficult to know whether the same cycles are being aligned or if the alignment is off by an integer multiple of cycles. Addressing this ambiguity requires complex statistical analysis. As with differential GPS, RTK systems require a local reference station or access to a reference station network.

RTK systems are well-developed and are already used to help tractor drivers precisely plant seeds and harvest grains. They also appear to be the most popular positioning system used in autonomous agriculture research, with demonstrated mobile accuracies of 10 cm standard deviation [10], 6 cm RMS [11], and 3.28 cm standard deviation [12], to name just a few.

High costs are the major drawback: receivers generally cost between $5,000 and $10,000, reference stations are in the tens of thousands of dollars, and subscription services are in the low-thousands per receiver, per year.

2.1.3 GPS DEAD-RECKONING FUSION

Dead-reckoning is the process of calculating a position by continually measuring changes in position. While error in each change of position measurement can be extremely low, the error in the absolute position is the summation of each of these individual errors; thus the position error is generally a percentage of total distance travelled. While that often rules out dead-reckoning for long distances, distances within a few meters may be accurate to within a few centimeters. Meanwhile, standard GPS is not very accurate (error in meters), but its accuracy is not dependent on distance travelled. The complementary nature of these two systems has motivated much research in their fusion, typically using a Kalman filter [13].

Dead-reckoning is usually carried out with either wheel odometry or an inertial measurement unit (IMU). Because a large and highly variable amount of wheel slipping occurs in outdoor, unpaved environments, inertial measurements are often preferred in agriculture settings. However, wheel odometry may be competitive at low price levels. For instance, [14] used two 16,000 PPR relative motor encoder (roughly $150 each today) on a differentially steered robot, and fused their measurements to a $130 GPS unit using a Kalman filter. An experiment was performed to calculate the position error over time; the results of which are found in Figure 2-2.

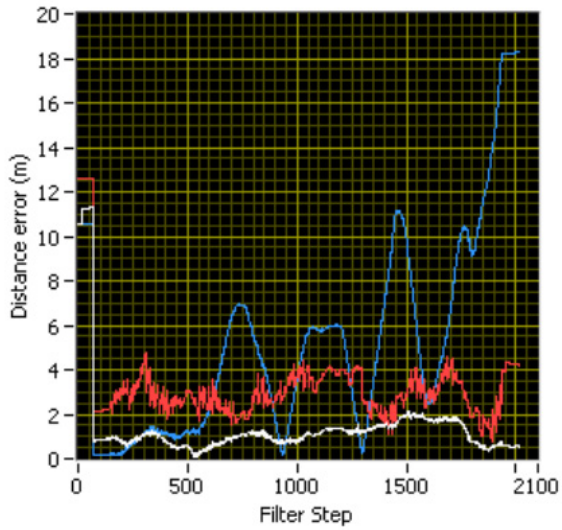


Figure 2-2. Error of wheel odometry and GPS fusion. Blue line represents the error in wheel odometry only, red line is error in GPS only, and white line is error in fused wheel odometry and GPS data. Used under fair use, 2012.

As would be expected, the position error of the wheel odometry grows over time, while the GPS error is consistently near three meters. The fusion of wheel odometry and GPS data reduced the error in the GPS-only data by roughly half.

GPS/IMU sensor fusion appears to be more accurate, but possibly only when highly accurate (and thus expensive) IMUs are used. In 2002, researchers [15] developed a tractor positioning system by combining a $200 off-the-shelf GPS unit and a $12,000 IMU, using a Kalman filter. The system reduced the mean bias of the GPS unit to 0.48 m latitude (from 1.28 m) and 0.32 m longitude (from 1.48 m), and allowed for an increase in the update frequency from 1 Hz to 9 Hz. Similar research [16] in 2004 with an off-the-shelf GPS unit and a high-performance IMU yielded a 95% confidence level in the error of 1.37 m (from 3.89 m with just the GPS unit). In 2006, researchers [17] developed a “low cost” positioning system using an off-the-shelf GPS unit in combination with a high-quality IMU. The price of the IMU is not listed, but the specifications indicate a cost of several thousand dollars. Like the two preceding publications, the Kalman filter fusion of the sensors reduced the position error by roughly 2/3, compared to the GPS-only solutions.

In general, GPS dead-reckoning sensor fusion appears able to cut GPS error in half, but unable to match the decimeter and centimeter precision of high-performance DGPS and RTK systems. Much of the research in this area is currently focused on improving GPS accuracy in unknown environments where high-performance DGPS is unavailable (such as roadways), or allowing for continual position data in areas where GPS data is unreliable.

2.1.4 COMPUTER VISION ROW DETECTION

Computer vision aims to provide computers with the ability to understand images at a high-level, similar to the way humans perceive high-level information from sight. A human picking produce on a farm generally does not use raw distance information like the previous localization techniques, but instead relies on the perception of individual plants and crop rows in order to determine location. Many researchers are attempting to replicate this process with computers.

The most common goal appears to be the detection of crop rows, but some researchers have attempted to detect soil tillage lines and the edges of harvested crops [18]. All of the goals are similar in that they attempt to detect line paths for an agriculture vehicle to follow autonomously. A number of vision-based techniques have been proposed, but they can generally be classified as either color/intensity recognition using monocular cameras or elevation recognition using stereovision.

Because lines are detected rather than individual points, the processes aimed for automation with these techniques are generally processes that do not require absolute position. Instead, they tend to be simple processes that occur uniformly as the vehicle traverses the lines.

Hague and Tillet developed [19] a method for detecting the rows of young wheat plants. The goal was to enable an autonomous vehicle to drag a hoe between the rows of wheat in order to remove weeds. With the knowledge that the plants were brighter than the soil, the algorithm was designed to look for relative changes in brightness in order to handle shadows, as shown in Figure 2-3. The center locations of rows were identified with high accuracy; to within 25 mm in 94% of the measurements.

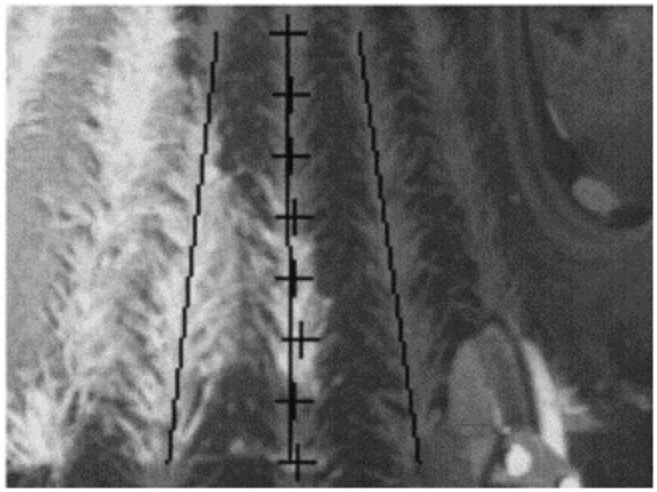


Figure 2-3. Wheat row detection in partially shadowed image. Used under fair use, 2012.

Another method of row detection is to recognize changes in height between plant rows and bare soil. In [20], researchers used stereo cameras to create a depth map in front of a tractor, as shown in Figure 2-4. The system was able to recognize bare soil based on its relatively low elevation, and use that information to autonomously navigate part of a soybean field. Compared to a baseline autonomous navigation using RTK GPS, the autonomous navigation using row detection had an RMS error of 5 cm.

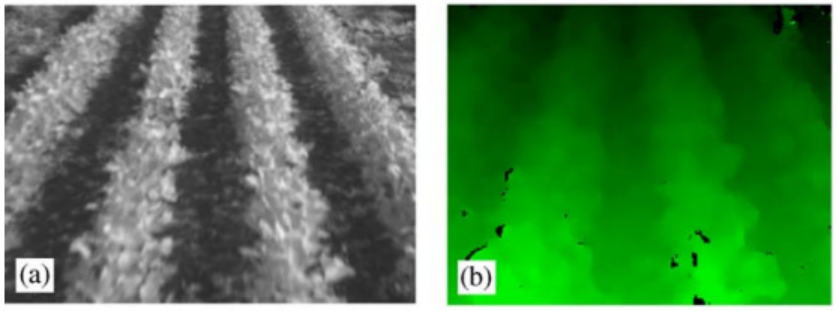


Figure 2-4. Soybean row detection with (a) single image from left camera; and (b) a disparity map calculated from images from both cameras. Used under fair use, 2012.

Crop row detection research is promising, but still confined to research. When it does work, the precision is very high, often comparable to RTK GPS. With some plants, the row spacing is short enough to require precision at this level. However, as noted by [21] in their literature review of autonomous agriculture vehicle guidance, there are still major challenges: dealing with a variety of plants species, growth stages, and lighting conditions; in addition to dealing with weeds in or between rows.

Furthermore, detecting individual plants along rows is an entirely separate and deeper challenge. While row detection may indeed be a valuable technique, there may be some limitations in the processes it can be applied to.

2.2 VISIBLE LIGHT POSITIONING

Visible light communication (VLC) and positioning (VLCP) has recently become a popular field of research. As the name suggests, information is transmitted in the visible region of the electromagnetic spectrum rather than the more common radio region. The popularity of this field is largely driven by the emergence of light emitting diodes (LEDs) as a ubiquitous light source. Unlike incandescent and fluorescent light sources, LEDs can be modulated very rapidly, allowing for demonstrated bandwidths of 130 megabits per second and theoretical bandwidths higher than 10 gigabits per second [22]. High-bandwidth photodiodes are needed to receive data at these speeds, but image sensors are also being considered for low-bandwidth applications.

Visible light positioning (VLP) is often seen as a supplemental feature to VLC. As such, it can be limited by the hardware used for communication. For instance, small numbers of high-bandwidth photodiodes can achieve high bitrates, but may not offer the position precision that an image sensor could provide. In other cases, VLP is seen as a viable indoor positioning system, where GPS signals are unavailable or unreliable, in which case communication may be subordinate to positioning.

VLP typically uses visible LED beacons and either image sensors or photodiodes. Positions are calculated in a variety of ways using the data collected by the image sensors or photodiodes. The two main measurements used within VLP are time of flight and light intensity. The following sections will present methods and results for each of these measurements from the research literature.

2.2.1 TIME OF FLIGHT

VLP using time of flight measurements operates similar to GPS: time of flight measurements from multiple beacons translate into distances, and multilateration of these distances provides a position calculation. Unlike GPS, however, most of the VLP literature using time of flight measurements uses the time difference of arrival (TDoA) method rather than the absolute time of arrival (ToA) method. While the TDoA method requires at least one extra beacon or receiver, it eliminates the need to synchronize the high-resolution clocks between the beacons and receiver.

A typical setup involves multiple LED beacons at known locations that output self-identifying signals via brightness modulation. Photodiodes on the mobile platform receive the optical signals and convert them into electrical signals for analysis by a signal processor, which identifies individual beacons and calculates time differences.

In [23], researchers studied the theoretical performance of TDoA under multiple visible light communication modulation schemes. In their approach, four overhead lighting LEDs broadcast their positions and are received by a single photodiode. Their simulation resulted in an average error of 14 centimeters.

The approach in [24] attempted to provide accurate outdoor positioning in cities (where GPS performance is poor), by taking advantage of the increasing number of LED-based traffic lights. Two sets of photodiodes are placed on a

vehicle, separated by a known distance. The LED in the traffic light communicates its position and is received by both photodiodes at slightly different times based on their distance from the traffic light. This provides the first time difference. When the car travels a known distance (measured with wheel odometry), another time difference is calculated. Some constraints allow for the two-dimensional position to be calculated with only two time differences. The first is that the photodiodes are assumed to only receive signals in front of the vehicle, so that the vehicle is known to be approaching the traffic light. The second is that the photodiodes are assumed to be perpendicular to the road, so the order in which the photodiodes receive the signal indicates whether the vehicle is to the right or left of the traffic light.

Alternatively, the system can recognize two different traffic lights and calculate a position without having to move the vehicle. The theoretical accuracy of this system is very high, but no experiment was performed. A significant challenge in using photodiodes outdoors is that there is much more lighting noise, so it is difficult to recognize LED lights, especially at large distances. The authors acknowledge that background noise was neglected in the development of their models.

2.2.2 INTENSITY

The observed intensity of a light source drops in relation to the squared distance from the source. Therefore, light intensity can be used as an alternative distance measurement to time of flight. Like time of flight methods, multilateration is used to calculate an actual position.

The major issues with intensity measurements are that they are susceptible to noise and can vary based on receiver orientation. In time of flight measurements, noise makes signal detection difficult, but does not necessarily add error to the distance measurement. With intensity measurements, noise can add significant error to the distance measurement in addition to making signal detection difficult.

In [25], researchers developed a calculation model for determining position based on received light intensity. The researchers note that not only does the output intensity of an LED vary based on angle; the reception angle of a photodiode significantly affects the received light intensity. In their experiments, a horizontal displacement resulted in a 50% greater change in received light intensity than an equivalent displacement directly toward or away from the LED. Without further information, distances cannot be estimated and trilateration cannot be performed. The researchers propose using nine LEDs rather than three, and compute a position that best fits the received light intensities, accounting for reception angle. They estimate that a position accuracy of 30 mm can be achieved when the receiver is rotating no more than 7 °/s.

2.3 OPTICAL RESECTION IN AN AGRICULTURAL SETTING

In section 2.1.4, positioning based on computer recognition of natural objects (i.e. crop rows) was presented. One way to deal with the challenge of recognizing natural objects in changing environments is to avoid it altogether by placing artificial objects in the environment. Prominent and consistent artificial objects (fiducials) can convey location information and be much more easily detected than natural objects. They are often used when navigation is confined to a relatively small and known environment. A notable example is the Kiva Systems® warehouse automation system, which uses teams of robots to retrieve shelves of merchandise. Location information is encoded into barcode tiles, which are captured with cameras and decoded by the robots. An image of the system is shown in Figure 2-5 [26] followed by an example image of the marker [27].

Figure 2-5. Kiva Systems® warehouse automation system.

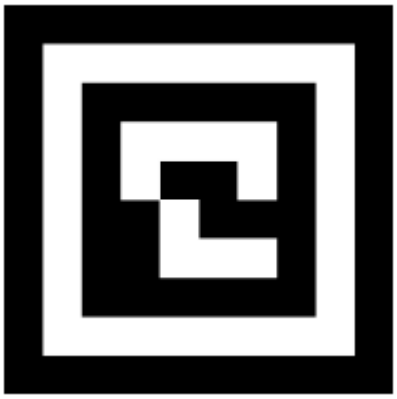


Figure 2-6. Unique marker provides position information.

While some research has applied this technique to agriculture settings (e.g. by placing encoded lines between crop rows), the large areas of farms compared to warehouses likely makes it preferable to reduce the density of fiducials and increase the precision of the image sensors on the robots.

A group of researchers from China and Japan have recently (2010 [28] and 2012 [29]) published two papers detailing an optical resection technique for agricultural settings, much like the approach used in this report. Their system and experimental results will be discussed in detail in the following subsections.

2.3.1 SYSTEM DESCRIPTION

This section summarizes and provides images from Sections I-III of [29]. Four beacons are located in the corners of a field at known positions. An omnidirectional camera mounted on top of a vehicle detects the four beacons. The angles between detected beacons are used to solve for the position of the vehicle, relative to any one of the four beacons.

The beacons are bright red, passive cylinders. The color red was chosen to provide contrast with the typical farm backdrop, and the cylindrical shape was chosen so that its two-dimensional image representation would not change based on the vehicle's two-dimensional position. The coordinates of the four beacons must be known.

The omnidirectional camera is composed of a camera and a hyperbolic mirror facing each other and with their principle axes aligned. The image sensor can capture at resolutions of 2048x1536 at 6 FPS and 648x480 at 20 FPS. Images are transferred to a computer using USB 2.0 with image format RGB24.

Beacon pixels are detected in an image with adaptive thresholding, which changes based on the brightness of the image. Once a beacon is detected, its center point is assumed to be the center of area of a group of pixels that passed through the threshold. When four beacons are detected, the four azimuth angles between the beacons are calculated, based on the center point of each beacon.

With the four azimuth angles and beacon coordinates, a two-dimensional location can be calculated. The calculation procedure is shown in Figure 2-7 [29].

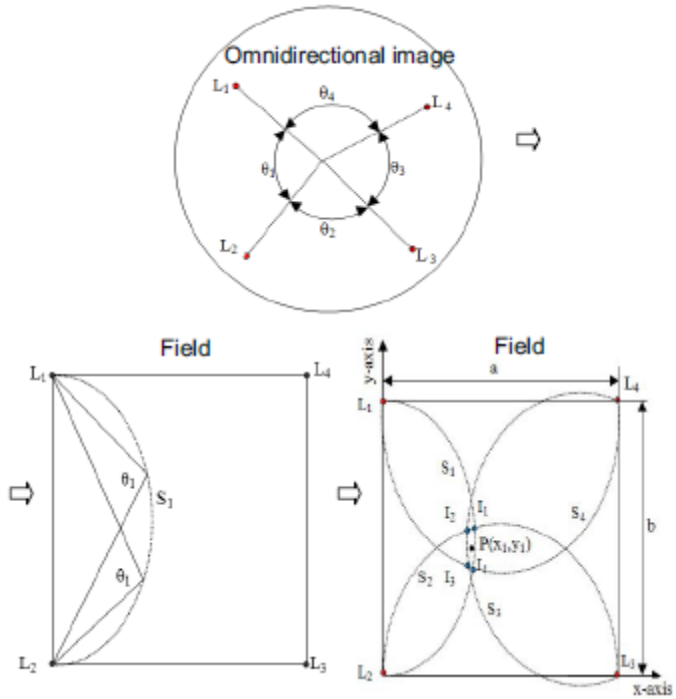


Figure 2-7. Location calculation. Used under fair use, 2012.

The circle at the top shows how a typical image from the omnidirectional camera would appear. Four azimuth angles (θ_1, θ_2, θ_3, θ_4) are calculated between the detected beacons (L_1, L_2, L_3, L_4). The bottom left square shows how a single azimuth angle between two beacons constricts the possible locations to an arc (S_1). The bottom right square shows four intersections (I_1, I_2, I_3, I_4) produced by the four arcs (S_1, S_2, S_3, S_4). The two-dimensional position solution is defined to be the center of the four intersections. With the two-dimensional position known (and thus the distance to each beacon), the height of the vehicle is calculated based on the zenith angle of one of the beacons and the known camera-mirror geometry.

2.3.2 EXPERIMENTAL RESULTS

This section summarizes and provides images and tables from Section IV of [29]. An experiment was performed on a flat, 50m x 50m field in natural sunlight. Four beacons were placed in the four corners of the field. The experiment made 16 different static position calculations, which resulted in a position RMS error of 24.51 cm and a maximum error of 38.94 cm

A second, dynamic experiment was performed on a 50m x 10m corner of the same field. The camera was set to record 1024 x 768 resolution images at 6 FPS. The camera was mounted on a vehicle that drove 50 meters down the center of the corner at 5.5 km/hr. Two-hundred images were captured by the camera during the experiment, and resulted in an RMS error of 8.34 cm.

2.3.3 POTENTIAL IMPROVEMENTS

This section summarizes Section V of [29]. The research showed that it is possible to obtain accurate positioning calculations using optical resection. The authors have pointed out some potential issues and future work. The most pressing future work is to develop a way for the system to calculate positions while the omnidirectional camera is tilted, which would occur in any non-flat environment. The authors also mention that installing a light in the beacons could improve results and allow for operation in low or no sunlight. In addition, the authors note that there is room for lowering the number of false positive beacon detections as well as an improvement in angle accuracy.

While this literature was not discovered in time to impact the thesis research, solutions to some of the potential improvements mentioned in this literature have been attempted. In particular, self-illuminating beacons were tested and a solution to obtaining position irrespective of orientation was produced.

3 MOTIVATION

The goal of this research is to develop an inexpensive, scalable, and reliable positioning system for labor-intensive farms that can approach the precision and accuracy of the differential GPS and RTK GPS systems discussed in Section 2.1. Compared to the alternative methods described in Chapter 2, optical resection appeared to be the best method to meet these goals.

There are three main factors that motivate the optical resection approach described in this report:

1. Image sensors provide increasingly cheap and accurate angle measurements
2. There are a large number of low-acreage farms, for which this method is particularly economical
3. Automation would be most beneficial to labor-intensive crops, which tend to be grown on small farms

These three factors will be discussed in detail in the following three sections.

Unless otherwise cited, all of the agricultural data presented in this chapter comes from the 2007 USDA Census of Agriculture [1].

3.1 INEXPENSIVE ANGLE MEASUREMENTS WITH DIGITAL IMAGE SENSORS

As described in Chapter 2, the positioning systems used in agriculture commerce and research are commonly some form of GPS augmentation and thus rely on time-of-flight measurements of radio waves. The primary advantage of time-of-flight measurements is that distance accuracy is not directly diminished as distance is increased; an uncertainty in the elapsed time leads to a constant uncertainty in length, regardless of total distance. In practice, atmospheric distortion of the speed of light becomes significant at large distances, but in general, time-of-flight measurements can offer high accuracy across large areas. The main disadvantage of these measurements is that they generally require expensive equipment.

An alternative to time-of-flight measurements is to use angle measurements and calculate distances based on the intersections of multiple angles. In this case, the distance accuracy is directly dependent on the distance: a doubling in distance roughly corresponds to a doubling in the distance error. The advantage of angle measurements is that they can be done very precisely with very inexpensive equipment, thanks to the widespread use of digital cameras.

High consumer demand for small, inexpensive, and high-resolution cameras embedded into laptops and smartphones has created a large and rapidly growing image sensor market, as shown in Figure 3-1 [30].

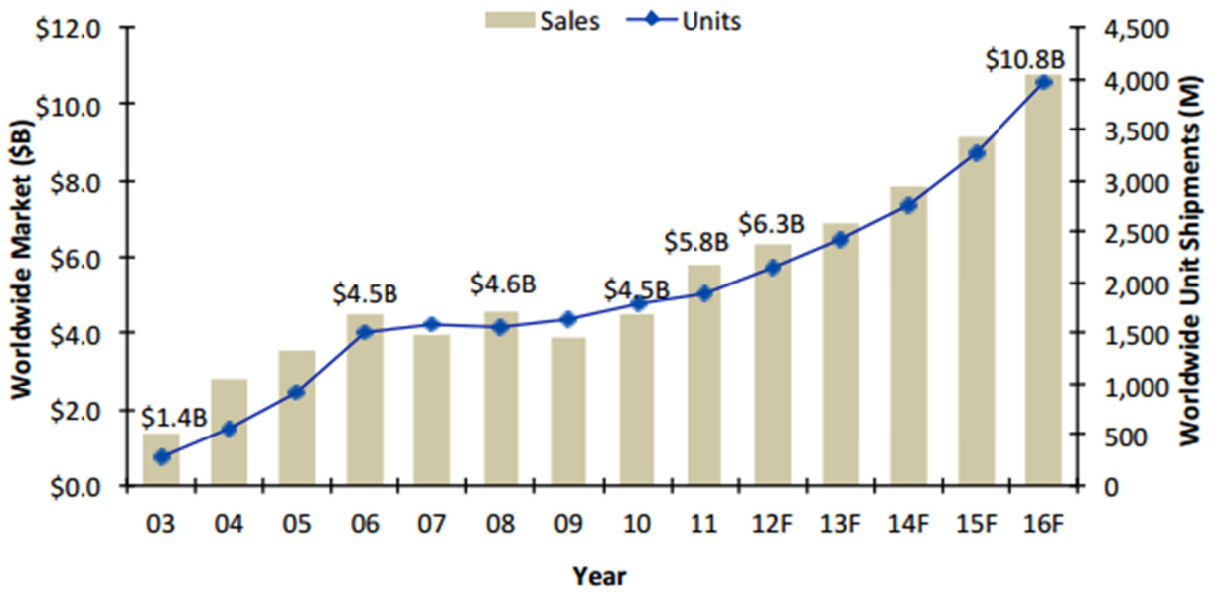


Figure 3-1. Historic and projected CMOS sensor sales.

The size and growth of the image sensor market has led to large investments in research and design, with a corresponding pace of advancement similar to microprocessors. In fact, CMOS image sensors utilize the same photolithography manufacturing process as modern microprocessors [31]. As an example of the rapid growth of image sensors, the first Apple® iPhone™ shipped in June, 2007 with a 2.0 megapixel image sensor and no video recording function. Four years later, the iPhone 4S™ shipped with an 8.0 megapixel image sensor and the ability to record 1080p video at 30 frames per second.

As of July, 2012, individual 10 megapixel image sensors can be obtained for less than $35. With a typical 55° field of view lens, such an image sensor could have an angular resolution of about 0.014°, which could translate into a roughly 2 cm distance precision at a distance of 50 m, for example.

Wafer-level cameras may cause a further acceleration in functionality and value. With wafer-level cameras, the entire camera assembly (including lens and filters) is created using existing photolithography manufacturing processes. Low resolution (640x480) wafer-level cameras measuring just 2.5mm x 2.5mm are commercially available.

3.2 SMALL FARMS

With resection, the number of beacons required to maintain a certain accuracy directly depends on the size of the area to be covered. This may be prohibitively expensive for large farms. For instance, as will be shown later, a typical horizontal and vertical spacing between beacons could be around 100 meters. Covering a square 2,000 acre farm would require about 841 beacons at this spacing. In contrast, differential GPS and RTK GPS may not

produce significant error until a receiver is 1-10 km away from a base station, which translates into areas between 775 and 77,500 acres. Therefore, GPS systems are likely to be more cost-effective for large farms.

However, small or medium-sized farms appear to offer a compelling market for optical resection. Table 3-1 shows the total number of farms in the US, based on acreage.

Table 3-1. Quantity of US farms and required number of beacons based on acreage.

Size (acres)	1 to 9	10 to 49	50 to 179	180 to 499	500 to 999	1,000 to 1,999	2,000 or more
Quantity	232,849	620,283	660,530	368,368	149,713	92,656	80,393
Beacons required	4 to 9	9 to 36	36 to 100	100 to 225	225 to 441	441 to 841	841+

The number of beacons required is calculated based on a 100 m spacing on a square farm. With a conservative estimate of $50 per beacon, about 1.5 million farms could be covered for less than $5,000. This is a very rough figure and does not include setup cost and power distribution, but should at least provide some idea of cost.

3.3 HAND-PICKED PRODUCE

Grain crops such as wheat and corn are harvested extremely efficiently, largely because they can be reaped, threshed, and winnowed with combine harvesters. Since the introduction of the self-propelled combine harvester in the early twentieth century, the output per hour of labor for wheat and corn has increased by an order of magnitude, as shown in Figure 3-2 [32].

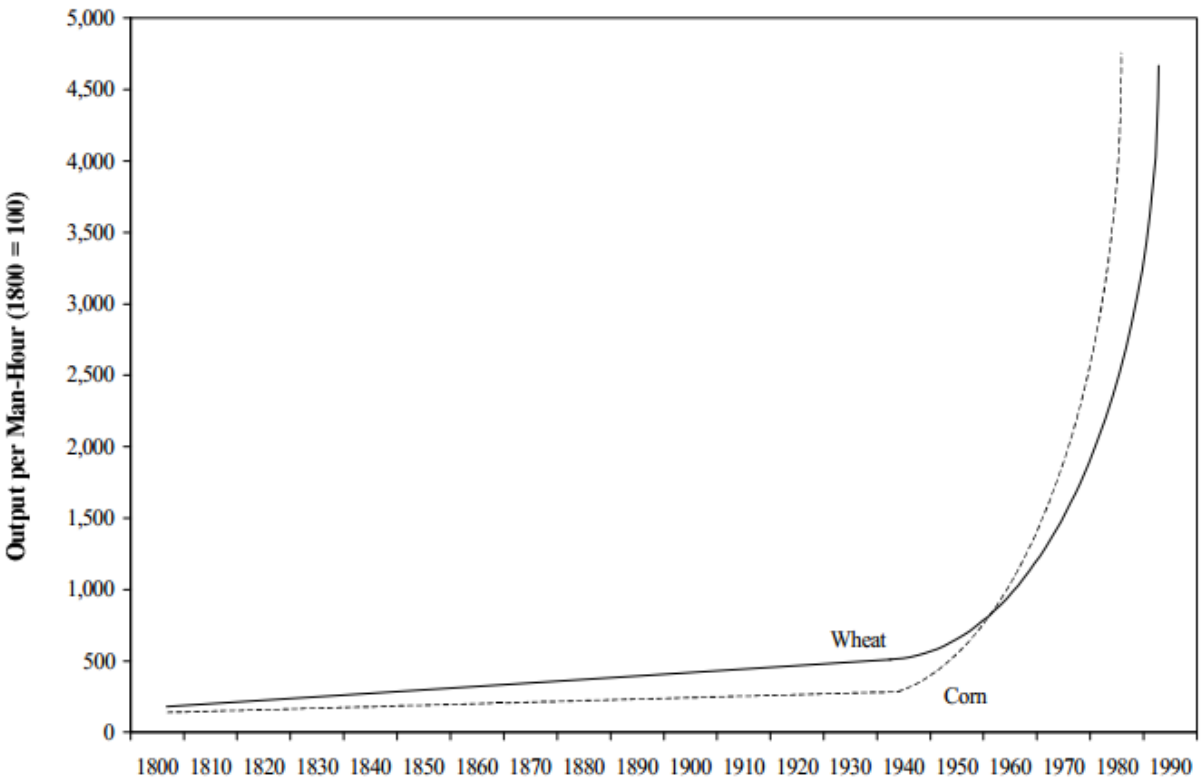


Figure 3-2. U.S. wheat and corn productivity over time.

Considering how little human labor is currently involved in the production of these crops, potential gains in productivity using autonomy may be limited. A more compelling application is likely to be for crops that are still hand-picked.

The USDA Census of Agriculture uses classifications from The North American Industry Classification System (NAICS). One of the classifications is farm type, which includes:

- Oilseed and grain
- Vegetable and melon
- Fruit and tree nut
- Greenhouse, nursery, and floriculture

Oilseed and grain farms are almost always harvested using machines (i.e. the combine harvester); while the latter three farm types often involve hand-picked crops. Table 3-2 shows the number of farms, average acreage, and average yearly labor costs for these four farm types.

Table 3-2. Quantity, size, and labor costs for select farm types.

Farm type	Oilseed and grain	Vegetable and melon	Fruit and tree nut	Greenhouse, nursery, and floriculture
Quantity	338,237	40,589	98,281	54,889
Average size (acres)	789	228	124	72
Average annual labor cost ($)	7,001	76,025	55,875	91,184

As can be seen, the three labor-intensive farm types have 8-13 times higher labor costs than oilseed and grain farms, despite being a tenth to a quarter of the size. So the farm types that can benefit the most from autonomy tend to be very small, providing a compelling motivation to develop the optical resection system.

4 THEORY

This chapter will detail the theories and principles of optical resection. The first section will describe the principle of obtaining positions based on resection, the second section will introduce the camera models that provide the angular measurements, and the third section will discuss the combination of these two concepts into optical resection.

4.1 RESECTION

Resection is the process of calculating the coordinates of an unknown point based on the angles to known points, observed from the unknown point. This section will show the derivation for the two-dimensional solution followed by the three-dimensional solution.

4.1.1 TWO-DIMENSIONAL SOLUTION

The two-dimensional solution always requires two angles measured to known points. The required number of known points depends on whether or not the orientation is known.

When the orientation (or simply heading, in two dimensions) is known, the solution is very simple. Figure 4-1 shows an example of this problem.

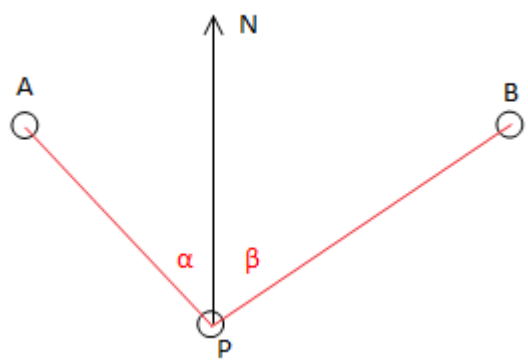


Figure 4-1. Two-dimensional resection with known heading.

An observer is at point P. The coordinates of point P are unknown but the heading of the observer is known to be north. Both the coordinates of points A and B and the angles to these points (α and β) are known.

The two red lines in Figure 4-1 are completely described by the angles α and β and the coordinates of points A and B. Thus, the coordinates of point P can be found by simply computing the intersection of these two lines.

If the heading was not known, only one angle would be known: the angle between A and B, which is the summation of α and β. In this case, an infinite number of positions along a curve between A and B are possible solutions, as shown in Figure 4-2.

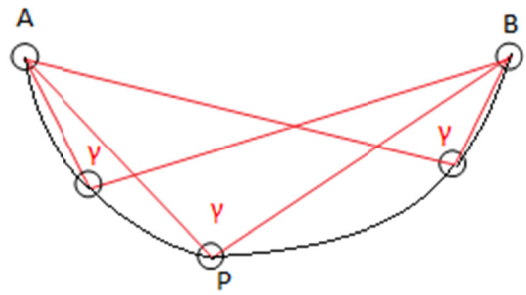


Figure 4-2. Two-dimensional resection with unknown heading.

In order to calculate the coordinates of point P, a second angle must be known. This can be provided by a third point with known coordinates. An example is shown in Figure 4-3.

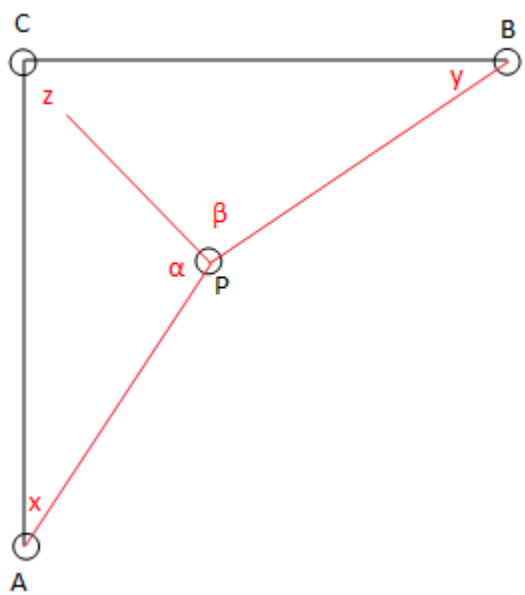


Figure 4-3. Two-dimensional resection with unknown heading and three known points.

In this example, the coordinates of points A, B, and C and the angles α and β are known. Angles x, y, and z and the coordinates of point P are unknown.

This is a well-known problem in surveying called the Snellius-Pothenot problem, first solved by Willebrod Snellius in 1615. The solution is as follows [33]:

Points A, B, C, and P form a quadrilateral, so the sum of all five angles must equal 2π radians (4.1).

$$x + y + z + \alpha + \beta = 2\pi \tag{4.1}$$

Applying the law of sines to triangles PAC and PBC yields (4.2).

$$\frac{AC \sin x}{\sin \alpha} = PC = \frac{BC \sin y}{\sin \beta} \tag{4.2}$$

An auxiliary angle is defined (4.3) and used to turn (4.2) into (4.4).

$$\tan \emptyset = \frac{\sin x}{\sin y} \tag{4.3}$$

$$\tan \emptyset = \frac{BC \sin \alpha}{AC \sin \beta} \tag{4.4}$$

The trigonometric identities (4.5) and (4.6) are used to turn (4.4) into (4.7).

$$\tan\left(\frac{\pi}{4} - \emptyset\right) = \frac{1 - \tan \emptyset}{\tan \emptyset + 1} \tag{4.5}$$

$$\frac{\tan[(x-y)/2]}{\tan[(x+y)/2]} = \frac{\sin x - \sin y}{\sin x + \sin y} \tag{4.6}$$

$$\tan\frac{1}{2}(x-y) = \tan\frac{1}{2}(\alpha + \beta + z)\tan\left(\frac{\pi}{4} - \emptyset\right) \tag{4.7}$$

Equations (4.1) and (4.7) contain two unknowns (x and y). The entire solution can be quickly found in an eight step algorithm [34], as shown in Equations (4.8) through (4.16):

$$\emptyset = atan2(BC \sin \alpha, AC \sin \beta) \tag{4.8}$$

$$K = 2\pi - \alpha - \beta - z \tag{4.9}$$

$$W = 2 * \text{atan}\left[\tan\left(\frac{\pi}{4} - \emptyset\right)\tan\left(\frac{1}{2}(\alpha + \beta + z)\right)\right] \tag{4.10}$$

$$x = \frac{K + W}{2} \tag{4.11}$$

$$y = \frac{K - W}{2} \tag{4.12}$$

$$if\ |\beta| > |\alpha|: PC = \frac{BC \sin y}{\sin \beta} \tag{4.13}$$

$$else, PC = \frac{AC \sin x}{\sin \alpha} \tag{4.14}$$

$$PA = \sqrt{AC^2 + PC^2 - 2 * AC * PC * \cos(\pi - \alpha - x)} \tag{4.15}$$

$$PB = \sqrt{BC^2 + PC^2 - 2 * BC * PC * \cos(\pi - \beta - y)} \tag{4.16}$$

With the lengths PA and PB known, the coordinates of P can easily be found. And with P known, the heading can be calculated based on the absolute angle to any one of the original three known points. For example, in reference to Figure 4-3 (with the north direction pointing straight up), the observer at point P knows (s)he is

travelling northwest if (s)he is directly facing point C. While actual measurements of angles α and β will have some error associated with them, the position calculation procedure itself is exact.

4.1.2 THREE-DIMENSIONAL SOLUTION

The three-dimensional resection problem is much more complicated. Exact, closed form solutions do exist, but they employ elaborate methods in solving eighth order polynomials and may take multiple seconds to compute [35].

Numerical solutions may be faster, depending on the precision required. However, many of the approaches in the literature require certain conditions in order to guarantee convergence. For example, as noted by [36], many numerical solutions in the photogrammetry community require an initial estimate of the position to within 10% and of the orientation to within 15°. Unique conditions are also encountered in the computer vision community. For example, three-dimensional resection is often used to estimate the position and orientation of objects relative to the camera. Possibly the most popular solution for this particular problem is the POSIT algorithm developed by [37]. While the algorithm is very fast, it is only accurate when the known points are relatively close to one another.

A suitable algorithm for this research could not be found, so a solution was developed. The following subsection describes the numerical algorithm developed in this research.

4.1.2.1 POSITION ALGORITHM

The approach begins with the observation that while the azimuth and zenith angles of and between two beacons will change based on roll and pitch, the direct two-dimensional angle will not (the two-dimensional angle is the angle on the plane formed by the two three-dimensional vectors). Thus, in order to allow for a position calculation independent of orientation, the algorithm was developed to use the two-dimensional angles. The four triangles formed by these angles, in combination with the two triangles formed by the four beacons, produce a hexahedron, as shown in Figure 4-4.

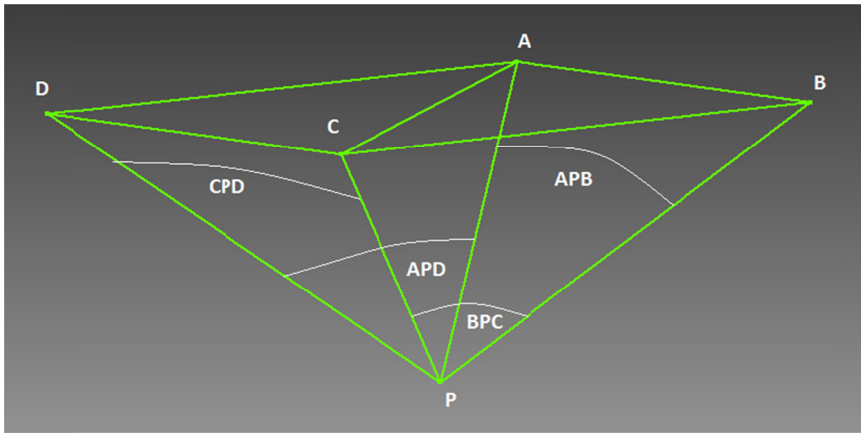


Figure 4-4. Known beacon points A, B, C, D and unknown point P form a hexahedron. Two-dimensional angles between beacons are shown.

While four known points are required to ensure a single solution, three points will work in most cases. A fourth point is required when three points produce two solutions. This algorithm will attempt a solution using three points and only use a fourth point to validate the solution. This both simplifies the calculation and allows for the possibility of obtaining positions when only three beacons are found. If four points are found, the algorithm will attempt to calculate a position using the three closest points and validate the position with the fourth point.

Three known points and the unknown point form a tetrahedron. The interior angle APC is then made use of, and the angles CPD and APD are only used to validate solutions when they are available.

There are two main steps in the algorithm:

1. Numerically solve for distances to beacons
2. Trilaterate distances and obtain position

The Newton-Raphson method was chosen as the numeric solution for its speed and precision.

4.1.2.1.1 NUMERICALLY SOLVE FOR DISTANCES TO BEACONS

As stated in section 4.1.2.1, a solution will first be attempted using the tetrahedron formed by three known points and the unknown point, and only validated with a fourth point when one is available and needed. If four points are detected, the algorithm will use the three closest points based on the position estimate in the previous section. The tetrahedron is shown for reference in Figure 4-5.

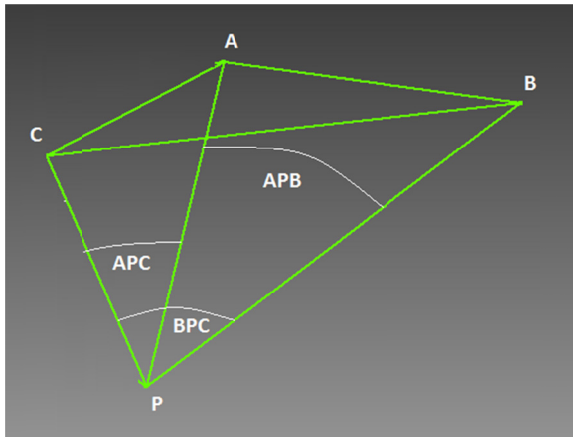


Figure 4-5. Tetrahedron formed by known points A, B, and C and unknown point P.

Three equations will be solved numerically: the law of cosines for the three triangles having unknown point P as a vertex, as shown in Equations (4.17) through (4.19).

$$\cos APB = \frac{ap^2 + bp^2 - AB^2}{2 * ap * bp} \tag{4.17}$$

$$\cos APC = \frac{ap^2 + cp^2 - AC^2}{2 * ap * cp} \tag{4.18}$$

$$\cos BPC = \frac{bp^2 + cp^2 - BC^2}{2 * bp * cp} \tag{4.19}$$

These three equations have three unknowns (lengths ap, bp, and cp), but attempting a simultaneous solution of these equations results in a cumbersome equation:

$$\cos APB = \frac{ap^2 + bp^2 - AB^2}{2 * ap * bp}$$

$$\begin{aligned} ap &= \sqrt{AC^2 + cp^2 \cos APC^2 - cp^2} + cp \cos APC \\ cp &= \sqrt{BC^2 + bp^2 \cos BPC^2 - bp^2} + bp \cos BPC \end{aligned} \tag{4.20}$$

Equation (4.20) cannot be directly solved in terms of bp, so the Newton-Raphson (NR) method will be used. The NR method is a numerical solution for finding successively better approximations of the roots of a function by taking advantage of derivatives.

For a function with a single variable, f(x) = 0, the algorithm is as follows:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{4.21}$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{4.22}$$

In equation (4.21), x_0 is an initial estimate. Equation (4.22) is repeated either a set number of times or until a sufficiently precise value of x is reached.

For a set of three equations (f_1, f_2, f_3) and three variables (x, y, z), the algorithm is somewhat more complicated:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} - \begin{bmatrix} \frac{\partial f_1(x_0,y_0,z_0)}{\partial x} & \frac{\partial f_1(x_0,y_0,z_0)}{\partial y} & \frac{\partial f_1(x_0,y_0,z_0)}{\partial z} \\ \frac{\partial f_2(x_0,y_0,z_0)}{\partial x} & \frac{\partial f_2(x_0,y_0,z_0)}{\partial y} & \frac{\partial f_2(x_0,y_0,z_0)}{\partial z} \\ \frac{\partial f_3(x_0,y_0,z_0)}{\partial x} & \frac{\partial f_3(x_0,y_0,z_0)}{\partial y} & \frac{\partial f_3(x_0,y_0,z_0)}{\partial z} \end{bmatrix}^{-1} \begin{bmatrix} f_1(x_0,y_0,z_0) \\ f_2(x_0,y_0,z_0) \\ f_3(x_0,y_0,z_0) \end{bmatrix} \quad (4.23)$$

Using the law of cosines of equations (4.17)-(4.19), the variables x, y, and z are swapped for ap, bp, and cp; and the equations are as follows:

$$f_1(ap, bp, cp) = ap^2 + bp^2 - AB^2 - 2 * \cos APB * ap * bp \quad (4.24)$$

$$f_2(ap, bp, cp) = bp^2 + cp^2 - BC^2 - 2 * \cos BPC * bp * cp \quad (4.25)$$

$$f_3(ap, bp, cp) = ap^2 + cp^2 - AC^2 - 2 * \cos APC * ap * cp \quad (4.26)$$

$$\frac{\partial f_1(ap, bp, cp)}{\partial ap} = 2ap - 2 * \cos APB * bp \quad (4.27)$$

$$\frac{\partial f_1(ap, bp, cp)}{\partial bp} = 2bp - 2 * \cos APB * ap \quad (4.28)$$

$$\frac{\partial f_1(ap, bp, cp)}{\partial cp} = 0 \quad (4.29)$$

$$\frac{\partial f_2(ap, bp, cp)}{\partial ap} = 0 \quad (4.30)$$

$$\frac{\partial f_2(ap, bp, cp)}{\partial bp} = 2bp - 2 * \cos BPC * cp \quad (4.31)$$

$$\frac{\partial f_2(ap, bp, cp)}{\partial cp} = 2cp - 2 * \cos BPC * bp \quad (4.32)$$

$$\frac{\partial f_3(ap, bp, cp)}{\partial ap} = ap - 2 * \cos APC * cp \quad (4.33)$$

$$\frac{\partial f_3(ap, bp, cp)}{\partial bp} = 0 \quad (4.34)$$

$$\frac{\partial f_3(ap, bp, cp)}{\partial cp} = 2cp - 2 * \cos APC * ap \quad (4.35)$$

With just a few iterations, the lengths ap, bp, and cp are solved to a precision within four decimal places.

4.1.2.1.2 TRILATERATE DISTANCES AND OBTAIN POSITION

With the known distances ap, bp, and cp to known points A, B, and C, the position can be solved using trilateration. The algorithm used can be found in [38] and in the actual code in the appendix.

Three parameters are checked in an attempt to ensure that the calculated position is correct:

1. The Newton-Raphson process converges
2. No lengths in the Newton-Raphson process are negative
3. The angles that would be apparent from the calculated position match the measured angles

4.1.2.2 ALGORITHM VERIFICATION

The algorithm was verified using a simulation. Four known beacon points were said to form a 100m x 100m square. Point P was placed at a location within the square, and the angles between beacons that would be seen from that location were calculated and input into the algorithm. The algorithm would then output a location, which was compared to the actual location. A total of 196,020 test locations were evaluated. These were spaced evenly at one meter increments in a volume formed by the beacon square (excluding the border) and up to 20 meters high. The distribution of test points is shown in Figure 4-6.

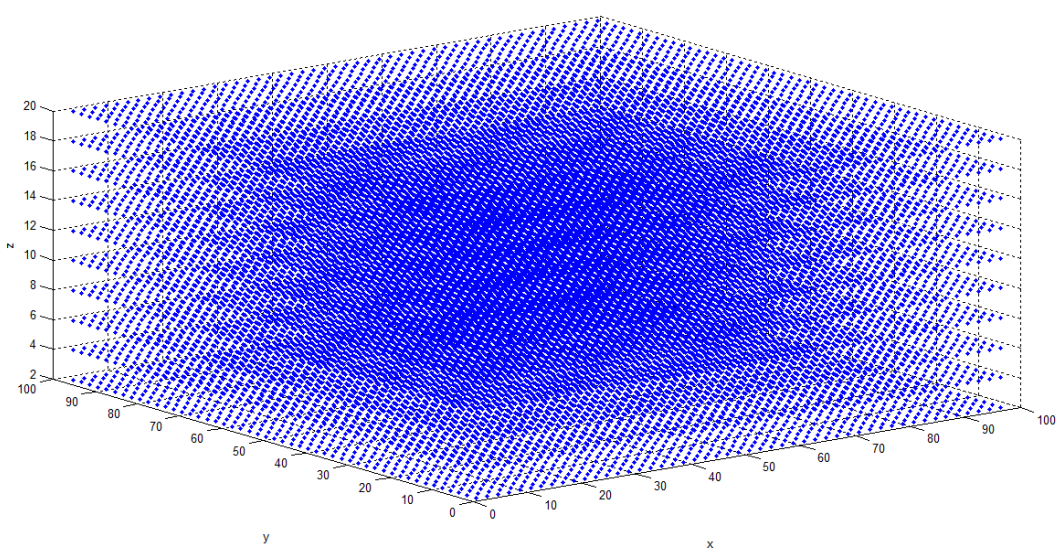


Figure 4-6. Test point locations. Only every other location along the x, y, and z directions are shown, for a total of 24,010 points, or 12% of the total test points.

The algorithm was tested based on the exact angles (using double precision numbers) that would be apparent at each test point. The purpose of this is to determine how much error, if any, the calculation procedure itself adds to the position error.

4.1.2.2.1 THREE BEACONS

When constrained to beacons A, B, and C, the spherical estimation combined with the Newton-Raphson numerical scheme resulted in 195,970 cases (99.97% of all cases) where all three lengths (ap, bp, and cp) were within 0.001m of the correct lengths. The average error in these lengths was 2.8e-11m. Of the 50 cases with an error greater than 0.001m, the average error in was 36.26m. Of these 50 cases, 21 had a length less than zero and 2 did not converge. So when the calculation is incorrect, it tends to be off by a large amount.

The trilateration calculation using these lengths resulted in 195,970 cases where the calculated position was within a distance of 0.001m of the actual position. These were of course the same 195,970 cases with accurate lengths. The average distance between the actual and calculated positions in these cases was 1.74e-10m. As mentioned in section 4.1.2.1.2, the third check is to see whether the angles at the calculated position match the initial 'observed' angles. Of the 50 cases with incorrect solutions, 28 matched the observed angles to within 0.057° (0.001 rad). Of these 28 cases, 2 either had a length less than zero or did not converge. So 26 incorrect position calculations would pass the three parameter error detection.

Avoiding the use of angles to beacons that are very nearby may also be an effective check. Of the 50 incorrect position calculations, 48 occurred when the actual location was in a corner near a beacon. The two that did not occur in a corner did not converge during their calculations. The positions of the 50 incorrect solutions are plotted in Figure 4-7.

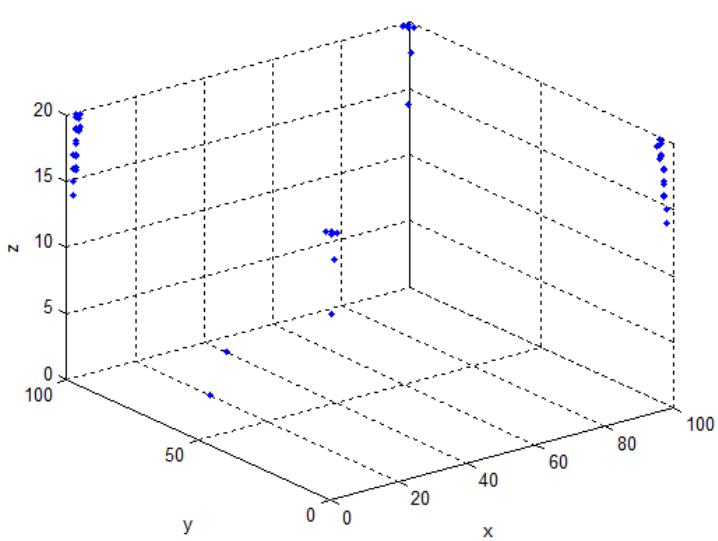


Figure 4-7. Positions with incorrect solutions.

4.1.2.2.2 FOUR BEACONS

Using all four beacons resulted in 195,986 cases where the calculated position was within a distance of 0.001m of the actual position. The average error in distance for these cases was 5.7e-14m. Of the 34 cases with a position error greater than 0.001m, all 34 were rejected by the error checking parameters.

The average calculation time was 0.79 milliseconds on a single thread using an Intel® Core™ i7-2620 2.70GHz processor, for both the three beacon and four beacon tests.

4.1.2.3 ORIENTATION

Obtaining the orientation is relatively simple once the position has been solved. The general strategy is to observe how the vectors from point P to three beacons are described in the vehicle's coordinate system compared to some reference coordinate system.

First, the observed vectors to three beacons (A, B, and C) are found with respect to the vehicle's coordinate system. These vectors are shown in (4.36).

$$A_o = \begin{bmatrix} A_{o,x} \\ A_{o,y} \\ A_{o,z} \end{bmatrix}, B_o = \begin{bmatrix} B_{o,x} \\ B_{o,y} \\ B_{o,z} \end{bmatrix}, C_o = \begin{bmatrix} C_{o,x} \\ C_{o,y} \\ C_{o,z} \end{bmatrix} \tag{4.36}$$

Then the reference vectors to the three beacons are found with respect to some reference coordinate system. The reference coordinate system can be in any orientation, but a practical one may be such that the y-axis is pointed north, the x-axis is pointed east, and the z-axis is in the opposite direction of gravity. The components of these vectors can only be calculated once the coordinates of P are known. The vectors are shown in (4.37).

$$A_r = \begin{bmatrix} A_{r,x} \\ A_{r,y} \\ A_{r,z} \end{bmatrix}, B_r = \begin{bmatrix} B_{r,x} \\ B_{r,y} \\ B_{r,z} \end{bmatrix}, C_r = \begin{bmatrix} C_{r,x} \\ C_{r,y} \\ C_{r,z} \end{bmatrix} \tag{4.37}$$

With these six vectors, the rotation matrix that maps vectors described in the vehicle's coordinate system to the reference coordinate system can be solved.

$$\begin{bmatrix} A_{r,x} \\ A_{r,y} \\ A_{r,z} \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} A_{o,x} \\ A_{o,y} \\ A_{o,z} \end{bmatrix} \tag{4.38}$$

Equation (4.38) shows how the vector from point P to beacon A in the vehicle's coordinate system can be mapped into the reference coordinate system using a rotation matrix. The rotation matrix describes how the vehicle's coordinate system is oriented with respect to the reference coordinate system. Equation (4.38) is a set of three equations and beacons B and C will similarly add three equations each. The nine unknown components of the rotation matrix can be solved for with these nine equations.

Once the rotation matrix is known, any vector in the vehicle's coordinate system can be mapped into the reference coordinate system. For example, the heading can be described as a constant vector in the vehicle's coordinate system and the rotation matrix can be used to calculate the respective direction in the reference coordinate system.

4.2 OPTICAL ANGLE MEASUREMENTS

Angles will be measured using digital cameras. Since the heading of the vehicle will vary, angles need to be observable from any azimuth direction. Two approaches are considered in this research: the first is a circular array of cameras and the second is a camera combined with a spherical mirror.

The camera model is required for both approaches and will be presented first, followed by the camera-mirror model.

4.2.1 CAMERA MODEL

A digital camera records the light intensity and color of a field of view across a two-dimensional array of pixels. Each pixel corresponds to a certain pair of zenith and azimuth angles within the field of view. Since the purpose of using images in this research is to make angular measurements between beacons, the angles associated with each pixel need to be known precisely, and thus a precise camera model needs to be developed.

The simplest camera model is the pinhole camera model. The typical pinhole camera is an opaque box with a small hole on one side. Light rays from a variety of directions pass through the hole and impact the opposite side of the box, where photographic film or a pixel array captures the intensity of the light rays. Figure 4-8 [39] shows the geometry of the pinhole camera model.

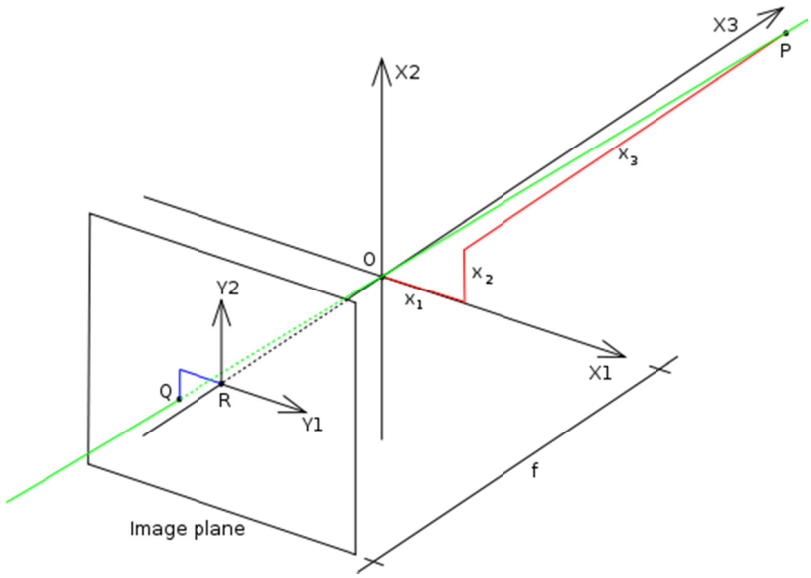


Figure 4-8. 3D pinhole camera geometry.

The X1-X2 plane is the side of the box with the pinhole, which is located at the origin. An object at point P (x_1, x_1, x_1) relative to the pinhole emits or reflects light. Light rays from many directions may be emitted from point P, but only one ray passes through the pinhole. In this case, the light ray passes through the pinhole and impacts the film on the Y1-Y2 image plane, at point Q. No other light ray will impact this point. In fact, every point on the image plane will be associated with a distinct light ray. The zenith (θ) and azimuth (φ) angles of the light ray associated with a point can be calculated based on the location of the point and the focal length (f) of the camera:

$$\theta = -\tan\frac{Y1}{f} \tag{4.39}$$

$$\varphi = -\tan\frac{Y2}{f} \tag{4.40}$$

The pinhole camera has a major drawback in that so little light is able to pass through such a small hole. Impractically long exposure times are required to get a bright, low-noise image. This is usually dealt with by increasing the size of the hole (aperture). However, a larger aperture allows light rays from the same point in space to impact multiple points on the image plane, creating a blurry image. A lens within the aperture causes these light rays to recombine at the same point on the image plane, thus focusing the image. Figure 4-9 [40] shows the effect of a simple lens within the aperture.

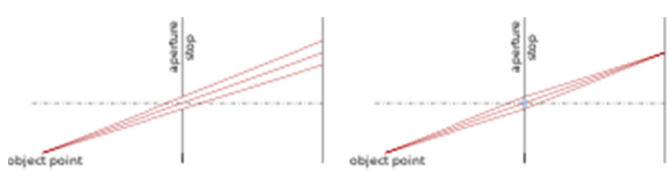


Figure 4-9. Effect of camera lens within large aperture.

While lenses correct the blurriness, they distort the image to some degree. The most common lens distortion is radial distortion, the two main varieties of which are shown in Figure 4-10 [41].

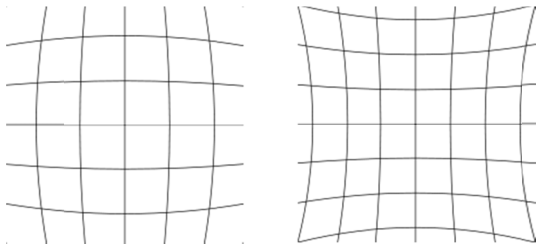


Figure 4-10. Radial lens distortion. The 'barrel' type radial distortion is shown on the left and the 'pincushion' type radial distortion is shown on the right.

This distortion means that the zenith and azimuth angles are no longer linearly dependent on the horizontal and vertical pixel location, and thus the simple pinhole camera model is not accurate. However, many camera models are able to account for radial distortion. One of the easiest to implement was an algorithm developed by [42]. The lens distortion is automatically calculated by taking at least two pictures from two different angles of a known rectilinear object (specifically a checkerboard) and measuring how the object distorts. The output of this algorithm is a set of parameters and equations that precisely relate pixel coordinates to zenith and azimuth angles. These are the intrinsic parameters of the camera.

For a multiple-camera setup, each camera has to be calibrated and assigned a unique set of intrinsic parameters. In addition, the relative position and rotation of each camera has to be determined so that the angle between beacons detected in different cameras can be accurately calculated. The position and rotation are defined by a translation vector and a rotation matrix, respectively. These are the extrinsic parameters of a camera. The extrinsic parameters can be found by taking pictures of an object at a known position from multiple cameras, in which case the fields of view of the cameras must overlap. If the fields of view do not overlap, then the camera array must be translated or rotated by a measurable amount in between camera shots.

4.2.2 CAMERA-MIRROR MODEL

An alternative approach of building an omnidirectional camera is to combine a single camera with a convex mirror. The convex mirror reflects light from all azimuth directions into the camera's more limited field of view. The mirror can have a number of shapes, three of which are shown in Figure 4-11.

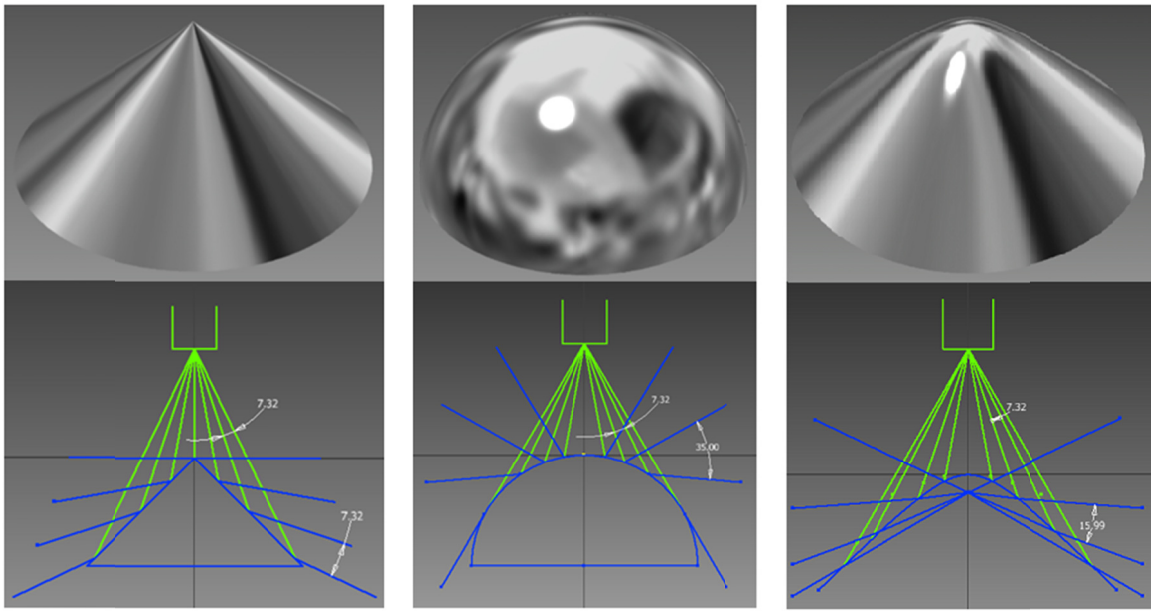


Figure 4-11. Conical, spherical, and hyperbolic mirrors (from left to right). Cross sections with camera ray lines shown below. Camera lines shown in green and their reflections shown in blue.

A conical mirror causes each camera ray line to rotate at an angle equivalent to the angle of the cone's vertex. Therefore (and conveniently), the angles between the reflected rays are the same. The drawback of the conical mirror is that it provides a very narrow zenith field of view. This would make it very difficult to detect beacons in any environment that was not flat.

A spherical mirror provides a much greater zenith field of view, which allows for the detection of beacons at a much greater variety of heights and orientations. The cost of this is a lower, and varying, angular resolution. The resolution at the lower zenith angles is roughly a quarter of the resolution at the higher zenith angles, depending on the camera and mirror parameters. This is the opposite of what would be desired; the high resolution areas would go unused unless a beacon is almost directly above the camera (or below, if the camera and mirror are flipped).

A hyperbolic mirror can provide about the same zenith field of view as the spherical mirror, but its resolution is much more consistent. Its reflected rays also intersect in a virtual focal point, allowing for an undistorted image and simpler calibration. For these reasons, the parabolic mirror is likely the best choice for this application.

Developing a model that maps image coordinates to azimuth and zenith angles is more complicated for these camera-mirror setups, but methods already exist in the literature. The approach that will be used extends the checkerboard distortion measurement method of [42] into camera-mirror models. This will be discussed in more detail in Chapter 5.

All of these camera-mirror models provide an increase in the field of view at a cost of a decreased angular resolution, but having the light from all directions pass through a single aperture does have some benefits. Generally, the pixels on a single camera can be synchronized more easily than with multiple cameras. Additionally, if expensive filters are used, only one is required.

4.3 LOCATION PRECISION

Location precision is dependent on the precision of the angular measurements between beacons and the distances between beacons. Both of these factors will be discussed in detail in the following subsections.

4.3.1 CAMERA-BASED ANGULAR PRECISION

In general, the angular precision of a camera is its horizontal and vertical field of view divided by the number of horizontal and vertical pixels, respectively (although it may vary slightly due to distortion). The resulting azimuth and zenith angular resolutions are usually about equal.

If a camera is perfectly calibrated and if the pixel that best corresponds to the center of a beacon can be determined, then the azimuth and zenith angles to that beacon are known to one half of the horizontal and vertical angular resolution of the camera. The reasoning is illustrated in Figure 4-12.

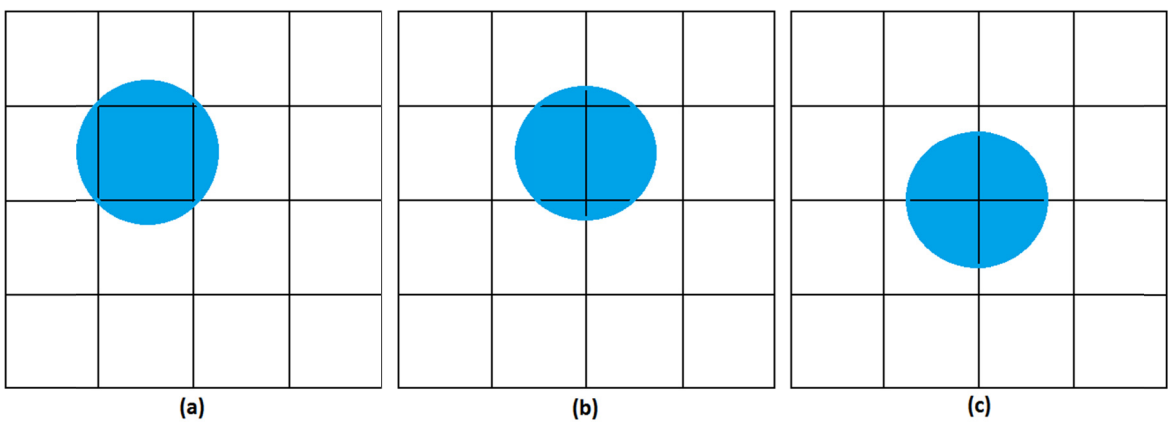


Figure 4-12. Various center beacon locations on a pixel grid.

In Figure 4-12a, the center of the beacon exactly coincides with center of one pixel. In this rare case, there would be zero error in both the azimuth and zenith angles. In Figure 4-12b, the beacon coincides between two horizontal pixels. Neither pixel best corresponds to the center of the beacon, so whichever pixel is chosen will contain an azimuth error equal to half of the angle from one pixel to the other. In Figure 4-12c, the beacon is in the middle of four pixels. This is the worst case; both the azimuth and zenith angles to this beacon will contain errors equal to half their respective angular resolutions. Since there is no preferred position of the beacon between pixels, the error in the azimuth and zenith angles should be a uniform random distribution between negative one half of the angular resolution and positive one half of the angular resolution.

As described in section 4.1.2, the actual parameter used in the position calculation is the angle between beacons. This is calculated by finding the absolute azimuth and zenith angles to two beacons, then using that information to describe their directions as unit vectors, and then taking the dot product of those two unit vectors. The full procedure is detailed below. First, the azimuth (φ) and zenith (θ) angles and their associated errors (ε) are found for two beacons (b_1 and b_2), and described as unit vectors in a Cartesian coordinate system:

$$\begin{bmatrix} x_1 = \sin(\varphi_1 + \varepsilon_{\varphi.1}) \\ y_1 = \sin(\theta_1 + \varepsilon_{\theta.1}) \\ z_1 = \cos(\varphi_1 + \varepsilon_{\varphi.1})\cos(\theta_1 + \varepsilon_{\theta.1}) \end{bmatrix} \tag{4.41}$$

$$\begin{bmatrix} x_2 = \sin(\varphi_2 + \varepsilon_{\varphi.2}) \\ y_2 = \sin(\theta_2 + \varepsilon_{\theta.2}) \\ z_2 = \cos(\varphi_2 + \varepsilon_{\varphi.2})\cos(\theta_2 + \varepsilon_{\theta.2}) \end{bmatrix} \tag{4.42}$$

The angle between these two unit vectors (α) is found using the dot product:

$$\cos(\alpha + \varepsilon_T) = \widehat{b_1} \cdot \widehat{b_2} \tag{4.43}$$

$$\begin{aligned} \cos(\alpha + \varepsilon_T) = {} & [\sin(\varphi_1 + \varepsilon_{\varphi.1}) * \sin(\varphi_2 + \varepsilon_{\varphi.2})] \\ & + [\sin(\theta_1 + \varepsilon_{\theta.1}) * \sin(\theta_2 + \varepsilon_{\theta.2})] \\ & + [\cos(\varphi_1 + \varepsilon_{\varphi.1})\cos(\theta_1 + \varepsilon_{\theta.1}) \\ & * \cos(\varphi_2 + \varepsilon_{\varphi.2})\cos(\theta_2 + \varepsilon_{\theta.2})] \end{aligned} \tag{4.44}$$

The total angle error (ε_T) should only depend on the individual angle errors and not on the actual angles themselves. To simplify things, the case where the two beacons are both found at azimuth and zenith angles of zero is considered:

$$\begin{aligned} \cos(\varepsilon_T) = {} & \sin(\varepsilon_{\varphi.1}) * \sin(\varepsilon_{\varphi.2}) + \sin(\varepsilon_{\theta.1}) * \sin(\varepsilon_{\theta.2}) + \cos(\varepsilon_{\varphi.1})\cos(\varepsilon_{\theta.1}) \\ & * \cos(\varepsilon_{\varphi.2})\cos(\varepsilon_{\theta.2}) \end{aligned} \tag{4.45}$$

As a combination of uniformly distributed random numbers, ε_T itself is more or less normally distributed. A test was carried out in order to get an idea of the distribution of ε_T. It was assumed that the azimuth and zenith angular resolutions of a camera were both 0.1° and that the errors were uniformly distributed between -0.05° and 0.05°. A million tests of equation (4.45) were carried out, with each of the four errors being assigned an independent and uniformly distributed random number. A histogram of the results is shown in Figure 4-13.

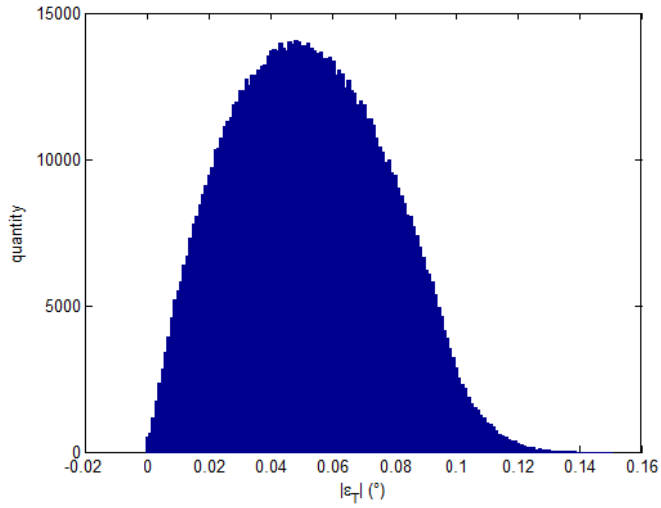


Figure 4-13. Histogram of total angular error.

As can be seen, the expected absolute value error is about 0.05° with a maximum error of ±0.14°. Since the absolute value of the error was calculated, a histogram of the error would also have a reflection of this graph about $\varepsilon_T = 0$.

This is assuming that a monochrome pixel array is used. If the common Bayer filter is used, the error in the azimuth and zenith angles may be doubled.

4.3.1.1 SUB-PIXEL ACCURACY

Since each pixel records the intensity of the light it receives, it may be possible to achieve even greater angular precision. Consider again Figure 4-12, reshown here for convenience as Figure 4-14.

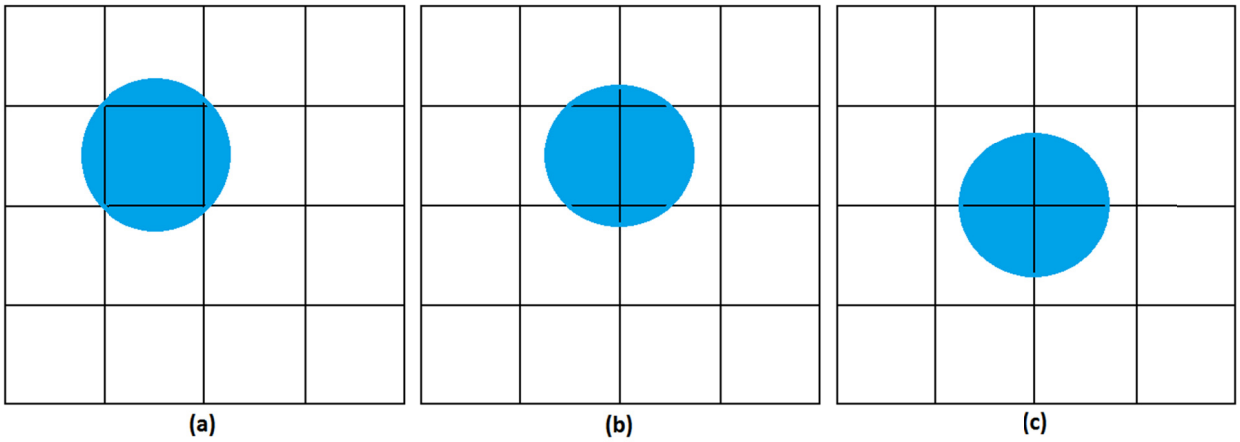


Figure 4-14. Various center beacon locations on a pixel grid.

Since the pixel at the center of the beacon in (a) is completely filled with light from the beacon, it will measure a significantly higher intensity than the surrounding four pixels. And since the surrounding pixels have equivalent

areas filled with light from the beacon, it may be possible to say with high confidence that the beacon is at the center of the pixel circumscribed by the beacon. Likewise for (b), the two pixels on either side of the beacon should receive approximately the same intensity, indicating that the beacon lies between them.

A typical 8-bit camera could theoretically increase the angular precision by a factor of 256. Internal and external noise would likely severely reduce that, but the principle may be very beneficial for static positioning.

The attempt at achieving sub-pixel accuracy is as follows:

1. Detect beacons by using color thresholds and/or blinking recognition and obtain estimate of their image coordinates.
2. Go back to the original image and find the 9 pixels with the highest intensities that match the beacon color, within a radius of 10 pixels from the beacon coordinate estimate.
3. Find the weighted average of the x and y coordinates of these 9 pixels, where the weight is the blue intensity of each pixel.

4.3.2 EFFECT OF ANGLE PRECISION ON POSITION PRECISION

In order to predict how the precision of the angular measurements affect the position precision, a number of simulations were carried out. The algorithm verification process presented in section 4.1.2.2 was repeated, but with error added in order to model its effect. A variety of angular resolutions (δ) were applied, with azimuth and zenith resolutions always being equivalent. It was assumed that there was no error in the position of the beacons and that the pixels that most represent the center of each beacon would always be found. As discussed in the previous section, the error in the azimuth and zenith angles to each beacon was modeled to be a uniformly distributed random number between –δ/2 and +δ/2.

Six cases were run limited to three beacons and six cases were run with all four beacons. Statistics for each of the tests are presented in Table 4-1.

Table 4-1. Position statistics for various angular resolutions.

δ (°)	0.02	0.04	0.10	0.20	0.30	0.50
3 beacons, angle check = 3δ						
# rejected	27	24	21	20	25	25
Mean error (cm)	3.26	5.56	13.37	23.32	36.08	60.74
CEP (cm)	1.36	2.82	6.85	13.92	18.77	32.23
R95 (cm)	7.07	13.79	37.06	66.08	108.74	180.75
4 beacons, angle check = 3δ						
# rejected	16	31	18	16	32	43
Mean error (cm)	0.87	1.78	4.34	8.80	12.91	21.63
CEP (cm)	0.77	1.62	3.85	7.86	11.67	19.19
R95 (cm)	1.72	3.52	8.45	16.86	25.79	42.42

The algorithm verified a solution by making sure that the angles with respect to calculated positions matched measured angles to within 3ε (in addition to checking for convergence and negative lengths). The three beacon cases have such higher errors because positions occurred in the corner near beacon D when only beacons A, B, and C were used. Thus, being able to recognize the nearest three beacons is a significant advantage. As would be expected, doubling the angular resolution consistently leads to a doubling in the position precision.

For reference, the common 1280x720 webcam typically has an angular resolution of around 0.05 degrees and can be obtained for less than $25. If these results can be achieved in practice, an array of such webcams with a 100m x 100m beacon grid would approach the RTK accuracies discussed in section 2.1.2.

5 DESIGN

This chapter presents the design of the system that was used for testing. The purpose of the design was to validate the theories detailed in chapter 4 and is not necessarily the best approach for a commercial product.

5.1 OMNIDIRECTIONAL CAMERA

This section will detail the design of the simple omnidirectional camera, followed by the calibration process.

5.1.1 CAMERA DESIGN

The omnidirectional camera was created by combining a spherical mirror and a webcam. While a hyperbolic mirror would have been preferred, a high-quality spherical mirror could be obtained for $35, while the few hyperbolic mirrors that could be found were hundreds of dollars.

The spherical mirror is 12mm wide and has a focal length of -7.75mm. The full specifications can be found in Appendix A. An image of the mirror as mounted in the system is shown in Figure 5-1.

Figure 5-1. Spherical mirror.

The camera is a Microsoft HD-5000 webcam, which retails for $49.99. It can capture 1280x720 resolution images with 3-channel, 24 bit color depth at 30 frames per second. Full specifications are in Appendix A. The camera housing was removed to both make mounting easier and decrease its size in the reflection of the spherical mirror. An image of the camera with housing removed is shown in Figure 5-2.

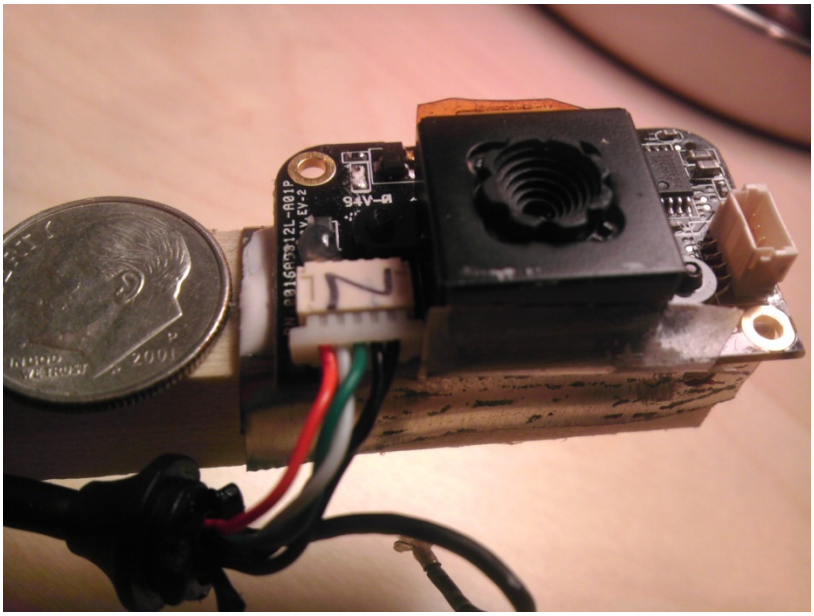


Figure 5-2. Webcam image sensor on sensor board.

The mirror and camera were simply glued to small wood beams and fixed onto a 0.25” bolt, using nuts and lock washers. The distance separating the camera and mirror was increased until the mirror filled the vertical field of view of the camera. An image of the entire assembly and an image taken by the assembly are shown in Figure 5-3 and Figure 5-4, respectively.

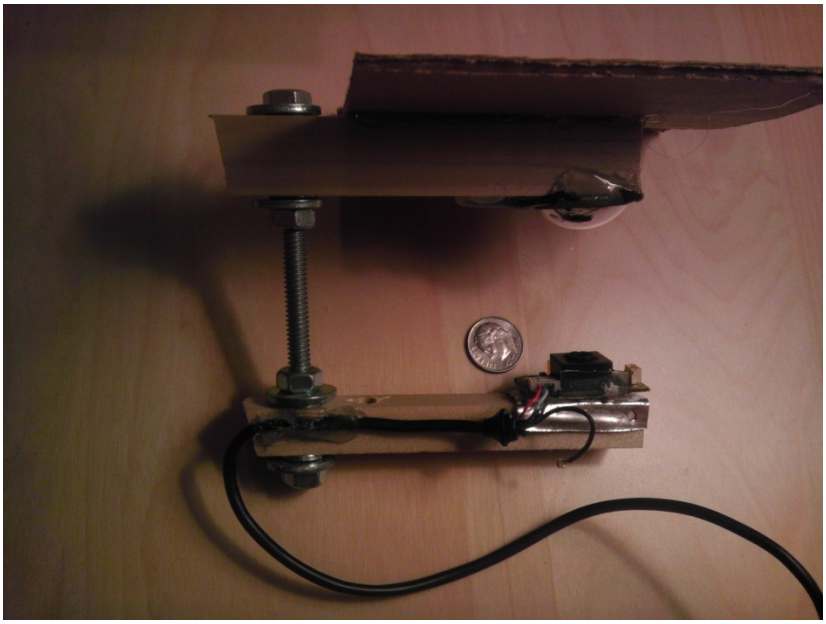


Figure 5-3. Camera and mirror assembly.

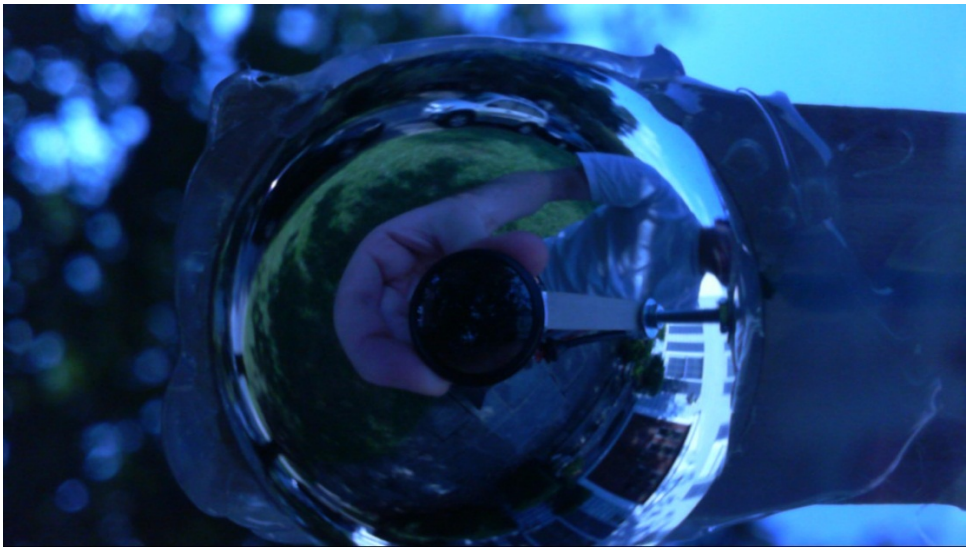


Figure 5-4. Photo from omnidirectional camera.

The field of view of the camera and the focal length of the mirror combine into a field of view with a full 360° azimuth and a zenith of ±127.5°. The 0.056° angular resolution of the camera is stretched to 0.26° at the center of the mirror and almost an entire degree at the edge of the mirror.

5.1.2 CAMERA CALIBRATION

The calibration process comes from [43], [44], and [45], which extends Zhang's method of camera calibration [42] to omnidirectional cameras. The system is specifically designed for a *central* omnidirectional camera (i.e. an omnidirectional camera with a single focal point), such as a camera combined with a hyperbolic mirror or a camera combined with a fisheye lens. The author notes that the model provides "very good results" with spherical mirrors, but does not compare the accuracy to central omnidirectional cameras.

Like in Zhang's method, the process works by taking pictures of a flat checkerboard at a variety of positions. The program extracts the corners of the checkerboard squares (automatically, but also allows precise user input) and attempts to model the intrinsic properties of the camera-mirror system based on how the grid of corners are skewed in the images. Specifically, the program finds three sets of information: the center of the image (vector T), slight misalignments between the camera and mirror axes (matrix A), and a polynomial that relates a pixel's distance from the center of the image to its zenith angle (function f(ρ), where f is the zenith and ρ is the distance in pixels from the center). The azimuth angle at any given pixel is assumed to be the two-dimensional angle formed by the pixel and the center of the image. The results for the omnidirectional camera used in this report are summarized in Equations (5.1) to (5.3).

$$T = \begin{bmatrix} row \\ col \end{bmatrix} = \begin{bmatrix} 362.32 \\ 639.74 \end{bmatrix} \tag{5.1}$$

$$A = \begin{bmatrix} 0.999780 & -0.000091 \\ -0.000002 & 1 \end{bmatrix} \tag{5.2}$$

$$f(\rho) = -239.72 + 5.85x10^{-5}\rho^2 - 2.64x10^{-7}\rho^3 + 5.45x10^{-10}\rho^4 \tag{5.3}$$

A vector describing the camera ray of any given pixel is calculated as follows:

1. Multiply the correction matrix (A) by the pixel coordinates. This outputs slightly adjusted pixel coordinates that account for the misalignment of the camera and mirror axes.
2. Calculate the distance in pixels (ρ) of the corrected pixel coordinates to the center of the image (T).
3. Calculate f(ρ) using Equation (5.3).
4. The value of f(ρ) is the z-component (aligned with the mirror and camera axes) of the camera ray. The x and y components are respectively the row and column distances from the center of the image.

The program attempts to quantify the amount of error in the model by using the camera model to re-project the checkerboard corners and then compare the re-projection coordinates with the original coordinates. The difference for each corner from all the calibration images used in this report is shown in Figure 5-5.

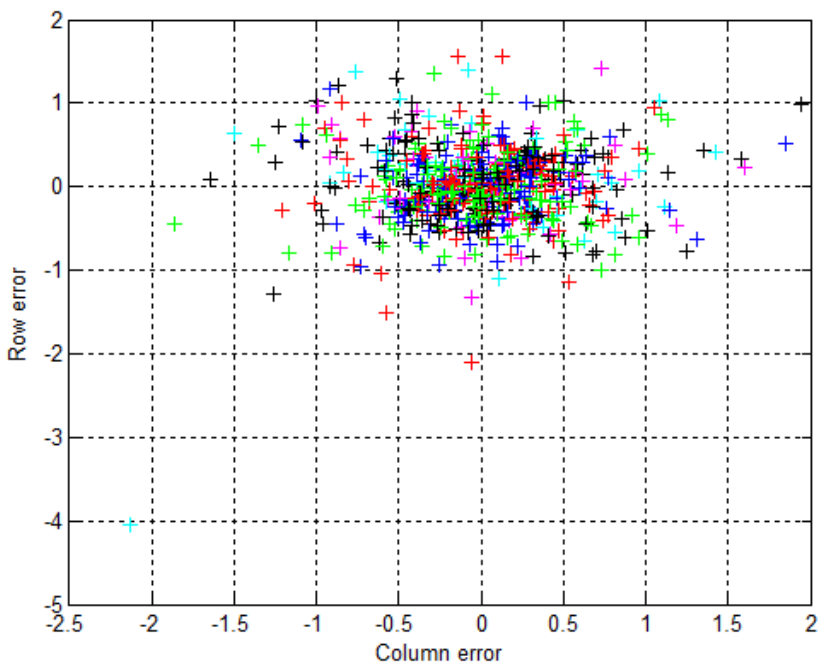


Figure 5-5. Re-projection error in calibration images.

The average error was 0.55 pixels. This indicates that achieving sup-pixel accuracy in the angle measurements is unlikely.

5.2 BEACONS

Brightly colored, passive beacons were tested early on (similar to the beacons used in [28] and [29]), but it was found that consistently detecting them was difficult due to changes in lighting conditions outdoors. Additionally, the ability of the system to function at night was desired. So the decision was made to use active beacons instead.

High-power light emitting diodes (HPLEDs) were chosen as the light source for four reasons:

1. They emit light in very narrow and consistent spectrums, making them easy to detect and allowing for the possibility of using very narrow bandpass optical filters.
2. They can be precisely modulated, which further distinguishes them from the environment and allows for the possibility of using optical communication (e.g. broadcasting their position information).
3. They are the most efficient lighting source available.
4. Some HPLEDs produce as much luminous flux as ordinary 40W light bulbs.

It is challenging to produce a light source that can be seen from tens of meters away in the high lighting conditions found outdoors. Originally, the approach in making the beacons easily visible was to combine the narrow spectrum of the HPLEDs with a narrow bandpass optical filter on the camera. That way, the majority of environmental light could be filtered out while all the light from the HPLEDs passed through. A blue HPLED with a 455nm wavelength was chosen for four reasons:

1. Blue contrasts well with the strong green and brown colors found on farms.
2. Extremely high brightness for an LED at 0.55W of blue light per LED.
3. A 10nm bandpass filter centered at 455nm could be obtained.
4. Perhaps the highest efficiency LED available of up to 53% wall plug efficiency.

As efficient as the LEDs are, they do produce about 0.5W of heat in $12mm^2$ footprints and can be damaged when temperatures exceed about 85°C, so some form of thermal management is required. The standard method of soldering the LED to a metal-core printed circuit board (MCPCB) and using thermal glue to adhere the MCPCB to a heat sink was used. An image of the assembly is shown in Figure 5-6.

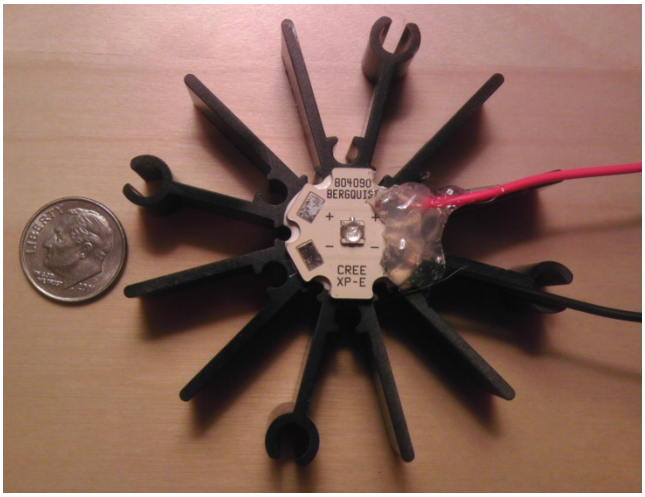


Figure 5-6. HPLED assembled on MCPCB and heat sink.

Individual HPLEDs were purchased for $2.36 each. The specifications for the HPLED can be found in Appendix A.

6 EXPERIMENTAL RESULTS

The system is currently unreliable in the bright lighting conditions of daylight, so all testing was done at night. Steps that may be taken to allow the system to work during the day will be described in section 7.2.1. The purpose of testing was to determine the static and dynamic location precision that could be obtained when beacons are easily detected.

The accuracy of the orientation calculations was not tested because the means to accurately measure the true orientation were not available at the time of testing. The calculations at least appear to be accurate though.

Because proper surveying tools were not in hand, the test area was confined to a relatively flat area so that the heights of the beacons and the height of the camera could all be assumed to be constant. Three HPLEDs (beacons A, B, and C) were placed in an L shape, with dimensions as shown in Figure 6-1.

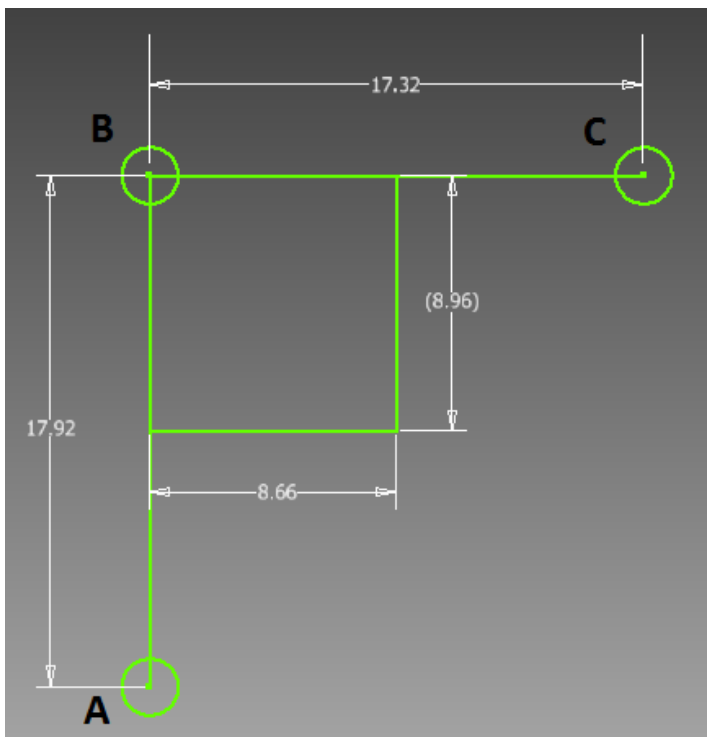


Figure 6-1. Experimental beacon grid. Dimensions are in meters.

As discussed in section 4.3.2, the algorithm calculates a position using the three nearest beacons (as long as all four are detected). Thus the positions tested in this experiment were confined to the top left quadrant shown in Figure 6-1.

A typical location in this geometry will see zenith angles to the beacons of roughly 85° on average (the axis of the omnidirectional camera was kept roughly perpendicular to the ground throughout testing). At 85°, the omnidirectional camera has a zenith resolution of 0.42° and an azimuth resolution of 0.23°. Repeating the simulation described in section 4.3.1 (where it was assumed that the pixel that most represents the center of a

beacon would be correctly chosen), a histogram of the angular error between two beacons is created and shown in Figure 6-2.

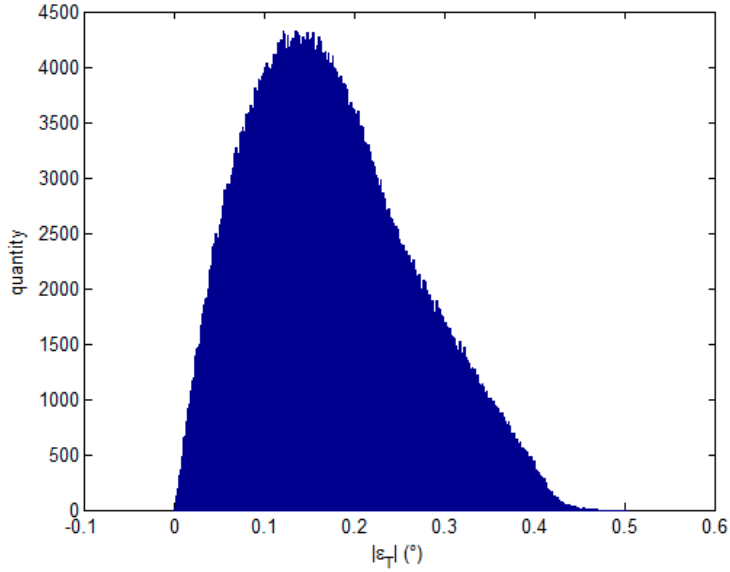


Figure 6-2. Histogram of total angular error expected in experiment.

The expected value of $|\varepsilon_T|$ is 0.163°.

6.1 STATIC POSITIONING

Ten positions within the quadrant were tested and compared to measurements taken by hand. The locations of the tests are shown in Figure 6-3.

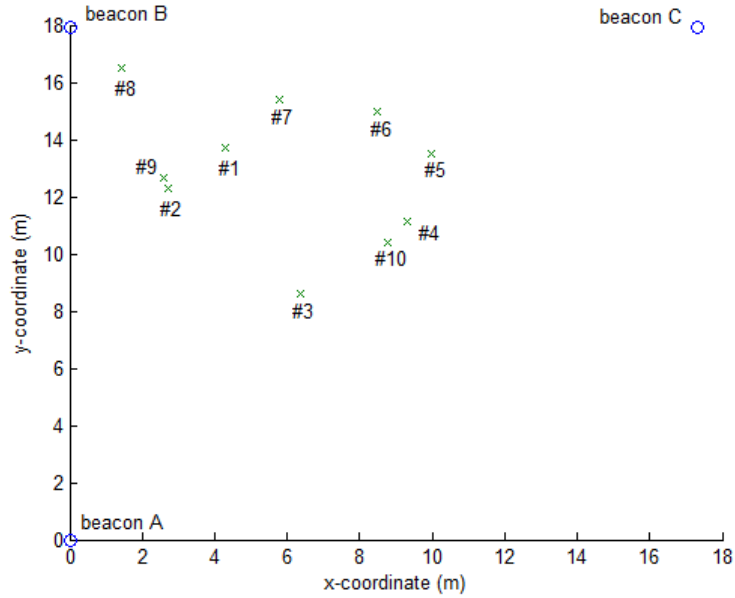


Figure 6-3. Test locations.

At each position, the system captured and calculated positions for 300 frames over 10 second intervals. The results for the first position are in Appendix C. The mean error for the ten positions is shown in Table 6-1.

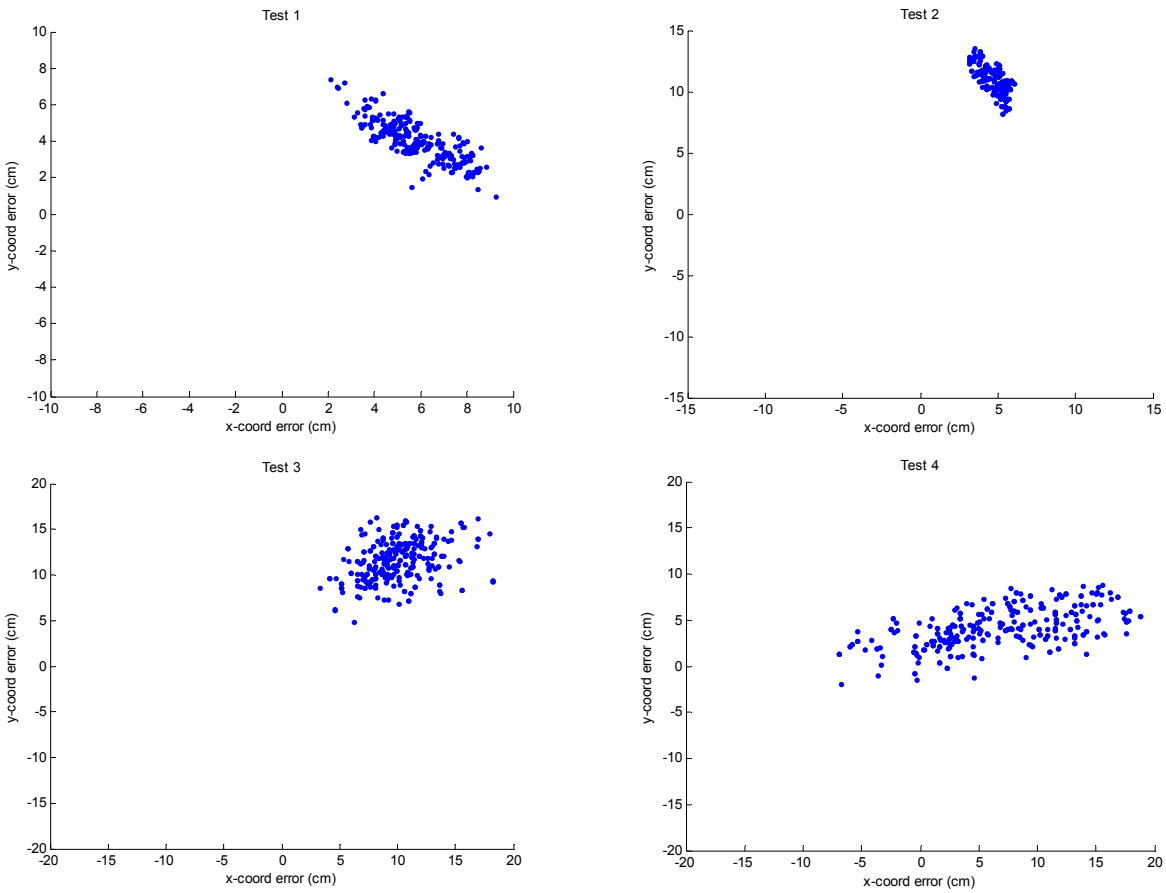
Table 6-1. Mean coordinate and angle errors.

Test #	Mean abs x-coord error (cm)	Mean abs y-coord error (cm)	Mean abs z-coord error (cm)	Mean abs APB angle error (°)	Mean abs BPC angle error (°)	Mean abs APC angle error (°)
1	5.78	4.02	27.41	0.09	1.50	0.45
2	4.75	10.88	30.12	0.59	1.55	0.18
3	9.99	11.62	33.02	0.31	1.08	1.93
4	7.49	4.27	32.00	0.34	0.73	2.32
5	10.36	4.75	33.07	0.44	1.20	1.73
6	10.10	7.59	31.90	0.60	2.21	0.63
7	9.56	7.15	29.18	0.73	2.67	0.37
8	75.57	20.66	39.34	4.85	3.33	3.17
9	10.26	2.58	28.80	0.31	1.31	0.86
10	6.60	3.70	33.38	0.19	0.64	2.52

Test #8 occurred very close to beacon B. As found in section 4.1.2.2.1, positions in a corner near beacons can result in two solutions. The calculated solution fluctuated between an approximately correct solution and a solution occurring in the opposite corner, where beacon D would have been placed. Without beacon D actually in the experiment, the incorrect solutions could not be rejected.

The error in the z-coordinate stands out with magnitudes roughly three or four times the error in the x and y coordinates. Interestingly, the error is heavily biased; the calculated z-coordinates are consistently about 30cm below their measurements. This was caused by the three angles having generally higher calculated values than actual values. One possibility is that the azimuth to each beacon is consistently overestimated. Error in the calibration model might explain this.

The x and y coordinates are also heavily biased in the positive x and y directions, apparently caused by significantly more bias in angles BPC and APC compared to APB. A measurement error in the position of beacon C could be a cause. Figure 6-4 shows the distribution of the error in the x and y coordinates for all ten positions.



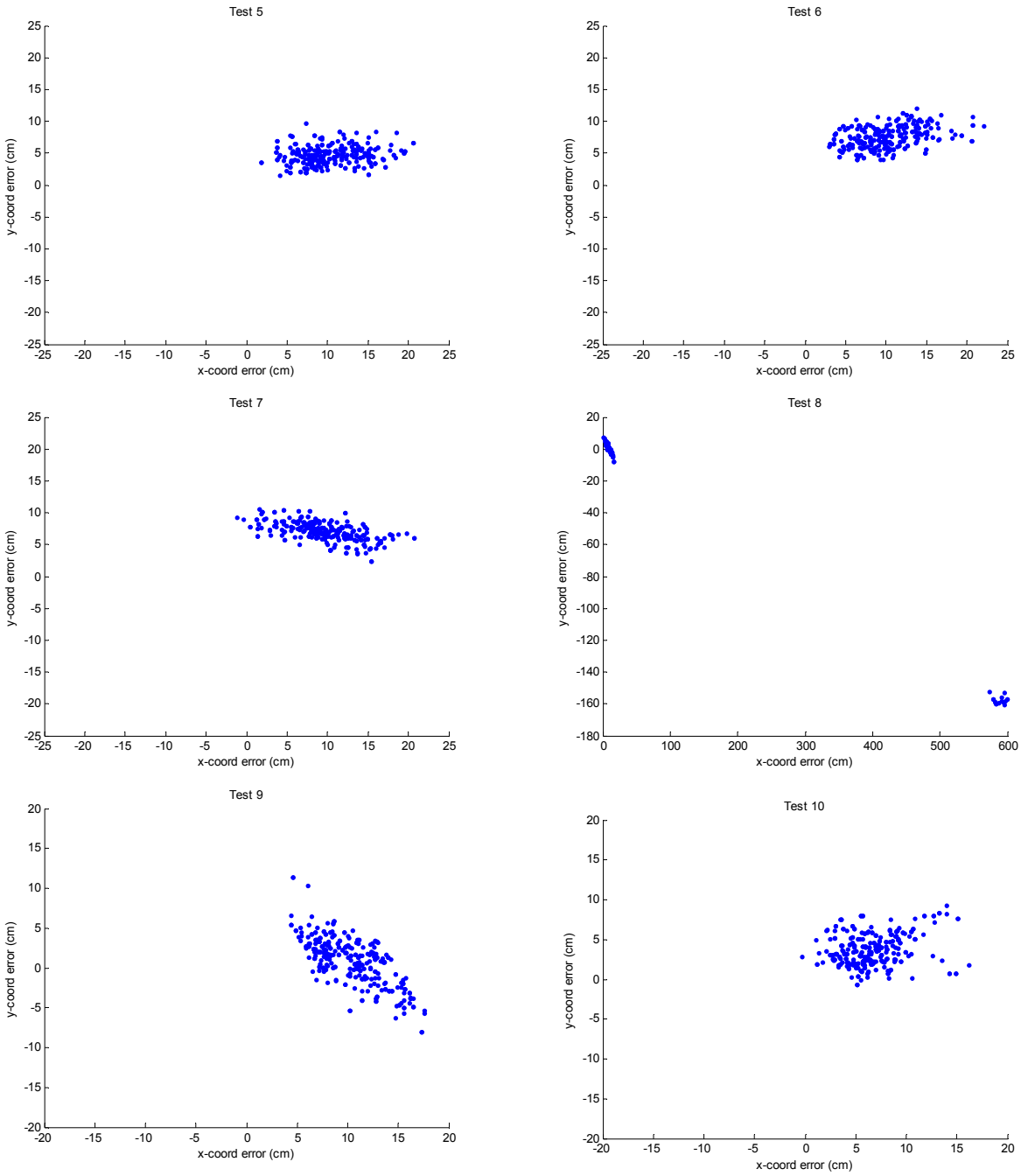


Figure 6-4. Error in x and y coordinates over all ten tests.

Significant bias in all calculations is also apparent when the absolute error is compared to the standard deviation of the calculations, which is shown in Table 6-2.

Table 6-2. Standard deviation of coordinate and angle calculations.

Test #	σ^2 x-coord (cm)	σ^2 y-coord (cm)	σ^2 z-coord (cm)	σ^2 APB angle (°)	σ^2 BPC angle (°)	σ^2 APC angle (°)
1	1.45	1.07	0.48	0.10	0.07	0.12
2	0.70	1.00	0.45	0.07	0.05	0.07
3	2.71	2.21	2.30	0.23	0.15	0.19
4	5.95	2.14	1.62	0.36	0.16	0.30
5	3.78	1.43	1.75	0.19	0.15	0.25
6	3.85	1.70	1.37	0.22	0.21	0.18
7	4.15	1.49	1.05	0.23	0.16	0.23
8	187.43	51.31	76.88	12.36	0.18	7.82
9	3.09	2.99	1.54	0.27	0.17	0.28
10	2.81	1.98	1.79	0.18	0.14	0.23

Interestingly, the standard deviations in the angles are only slightly higher than the expected value of 0.163° for $|\varepsilon_T|$. This indicates that the calculated centers of the beacons generally did not fluctuate much more than half a pixel.

With calibration errors appearing to be a possible cause of the bias in the calculations, the omnidirectional camera was recalibrated and tested again. A similar beacon layout was set up (the distance between the A and B beacons was 58cm shorter and the distance between the B and C beacons was 187cm longer) and three positions were tested. Eight different orientations were tested in the third position. As in the last experiment, positions were calculated across 300 frames over a 10 second period. The results are shown in Table 6-3.

Table 6-3. Mean coordinate error after recalibration.

Test #	Location coordinates (m, m, m)	Pitch and yaw	Heading	Mean abs x-coord error (cm)	Mean abs y-coord error (cm)	Mean abs z-coord error (cm)
1	10.46, 8.47, 1.35	Level with ground	Facing beacon B	9.60	10.61	4.84
2	5.06, 12.68, 1.35	Level with ground	Facing beacon B	1.53	8.10	1.67
3	9.06, 13.46, 1.35	Level with ground	20° CCW yaw from beacon B	5.02	5.78	0.71
4	9.06, 13.46, 1.35	Level with ground	10° CW yaw from test #3	4.60	1.51	3.84
5	9.06, 13.46, 1.35	Level with ground	10° CW yaw from test #4	11.85	5.19	2.83
6	9.06, 13.46, 1.35	Level with ground	10° CW yaw from test #5	6.05	2.63	2.06
7	9.06, 13.46, 1.35	Level with ground	10° CW yaw from test #6	13.73	1.97	1.49
8	9.04, 13.48, 1.34	Pitched forward 20°	Facing beacon B	12.33	3.43	1.87
9	9.08, 13.44, 1.34	Pitched backward 20°	Facing beacon B	4.05	1.80	9.14
10	9.07, 13.48, 1.35	Rolled right 20°	Facing beacon B	3.98	3.82	9.60

The bias in the z-coordinate has been greatly reduced, but there is still bias in the positive direction of all three axes. The orientation tests indicate that this bias is likely being caused by a misalignment between the camera and mirror axes. Scatterplots of the yaw tests (# 3-7) are shown in Figure 6-5.

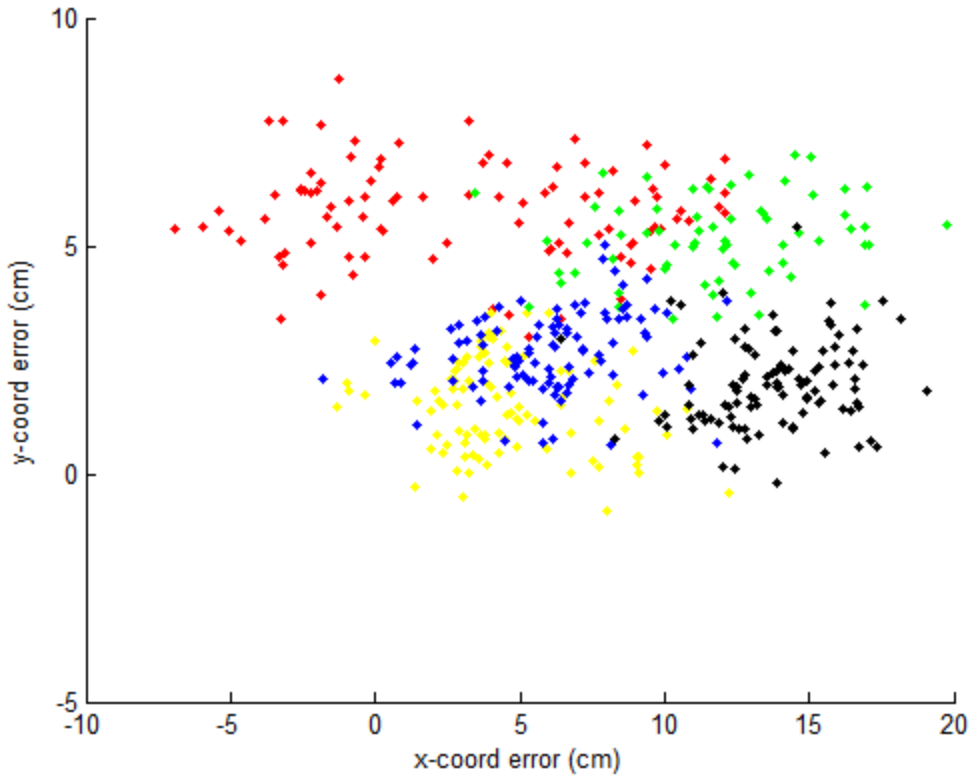


Figure 6-5. Scatter plots of x and y coordinate error in yaw tests. All tests occurred at the same location. The red series had a heading 20° to the left of beacon B. The other tests were each rotated 10° to the right (CW) relative to the previous heading, in order of yellow, green, blue, and black.

Each of the orientation tests clearly has a unique bias. This rules out measurement error as the main error source because each test occurred in the same location (rotations were made about the axes of the camera and mirror). By visual inspection, the centers of the beacons in the images appeared to be consistently calculated within two pixels at most, and they would have to be off by 5-10 pixels to explain the magnitude or error, which is larger than the beacons actually appear in the images.

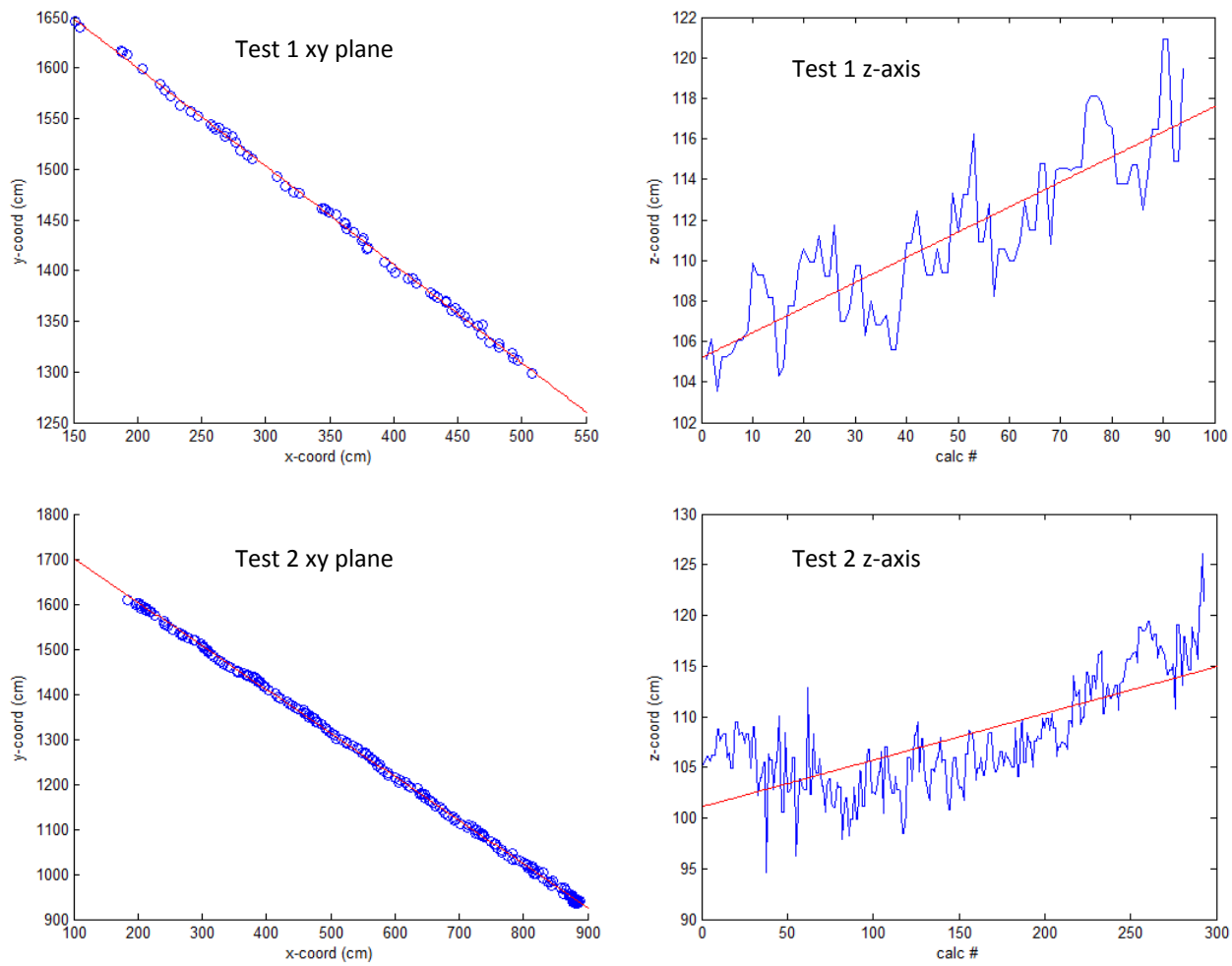
The error must be in the calibration that maps pixel points to directions. A misshapen mirror could cause such an error, but seems unlikely based on the quality of the mirror. The alignment of the camera and mirror axes does appear to be somewhat off. While [45] mentions that the calibration procedure attempts to account for misalignment, the degree of misalignment must be "small."

6.2 DYNAMIC POSITIONING

The camera used in the omnidirectional camera has a CMOS image sensor. CMOS image sensors typically employ a rolling shutter rather than a global shutter. As such, the pixels are not synchronized; the values of pixels toward

the top of a frame are determined at slightly earlier times than the pixels toward the bottom of the frame. If a beacon is captured toward the top of the frame and the camera moves before another beacon is captured toward the bottom of the frame, then the angle between the two beacons will appear to be skewed and the position calculation will have additional error. The purpose of the dynamic positioning experiments is to quantify the level of error that could be expected while the camera is in motion.

Three dynamic tests were carried out. In each, a start position on one of the edges of the testing quadrant was chosen, with the end point being the corner by beacon B. The omnidirectional camera was held in hand as a tester walked in a straight line in a variety of paces. Plots of the calculated transits in the xy plane are shown in Figure 6-6.



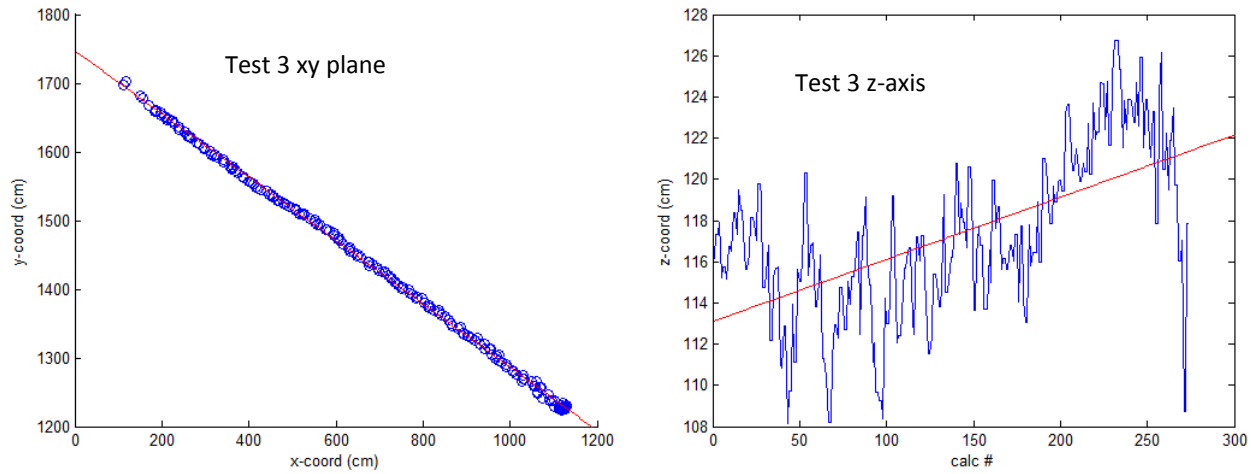


Figure 6-6. Plots of transits with least squares linear fit lines for three dynamic tests.

There is a general upward trend in the z-axis data because the error in the calculated z-coordinates decreases as the position approaches beacon B. It is assumed that this decrease in error is linear.

Table 6-4 shows relevant statistics of the data. The average speed was calculated by dividing total distance traveled by total travel time. The deviations were calculated based on the distances from each point to the least squares linear regression models.

Table 6-4. Relevant statistics for dynamic tests.

Test #	Average speed (m/s)	Average xy deviation (cm)	Standard deviation xy deviation (cm)	Maximum xy deviation (cm)	Average z deviation (cm)	Standard deviation z deviation (cm)	Maximum z deviation (cm)
1	1.58	1.74	1.17	5.37	1.70	1.16	4.54
2	0.99	2.94	1.84	8.59	3.08	2.06	11.53
3	1.21	3.34	2.14	11.65	2.72	2.02	12.52

Some level of deviation is of course expected because the actual transits were not perfectly straight. Without even accounting for that, however, these deviations do not appear to be much different than the deviations in the static tests shown in Table 6-2.

It should be noted, though, that these experiments are significantly smaller in scale and with significantly lower resolutions than would be seen in practice. Higher angular resolutions and larger distances would be more sensitive to motion when pixels are not synchronized.

7 CONCLUSIONS AND FUTURE WORK

7.1 CONCLUSIONS

The goal of this research was to investigate the potential of optical resection as a high-precision and low-cost positioning technique. The principle of using optical resection to calculate position does indeed work. While the theoretical precision described in Chapter 4 has not quite been reached in experiments, there is reason to think that it can be with a more detailed construction and calibration process.

It is too early to predict the design and cost of a commercial product, but considering the low-cost of the main components, it is hard to imagine the individual units being uncompetitive with other systems, especially RTK GPS receivers. The most significant cost uncertainty is with the beacons. The amount of power required to make active beacons reliable is not known, so the power distribution costs cannot be accurately estimated. However, high beacon costs would likely decrease the area that can be covered cost-effectively, rather than making the approach too expensive in general. With 230,000 farms less than 10 acres in the United States, the costs of beacons would have to be extremely high in order to be uncompetitive with an RTK GPS base station, for example.

7.2 FUTURE WORK

The most pressing area of future work is to find a way to make beacons consistently detectable during the day. Other areas of consideration include: multi-camera configurations, increasing dynamic precision, and integration with other systems such as GPS and IMUs. These areas will be discussed in more detail in the following subsections.

7.2.1 BEACON DETECTION IN HIGH AMBIENT LIGHT ENVIRONMENTS

The idea of matching narrow-spectrum LEDs with narrow bandpass filters was mentioned in section 5.2. While this approach is not reliable at the time of this writing, it appears to be worth pursuing. Photographs using this technique are shown in Figure 7-1 to demonstrate the potential.

Figure 7-1. Photographs of one LED from a distance of 22.9m. The first photograph is with the LED powered (circled in red) and the bandpass filter in front of the lens. The second photograph is with the LED unpowered to show the difference. The third photograph shows the LED on and no bandpass filter in front of the lens. All photographs were taken under clear skies at approximately 1 PM on September 18th at latitude 38.85°N.

As can be seen in the images above, 455nm light is highly absorbed by the vegetation. Similar blue wavelengths and red wavelengths would generally work well on farmland, with slight variations depending on the dominant form of chlorophyll.

While the LED is easily visible with the bandpass filter, it is not distinguishable from other sources of blue light. Even in an agricultural setting, beacons would likely have to modulate in defined sequences in order to be distinguished.

Future research should also further investigate the use of passive beacons. Measuring ambient light and modeling how passive beacons change appearance in different lighting conditions is one possible route to improve detection. Another is to use patterns of colors on each beacon so that color thresholds can be loosened while still distinguishing beacons from the environment.

7.2.2 MULTIPLE CAMERA CONFIGURATIONS

The main disadvantage of a single camera combined with a convex mirror is that the zenith field of view is limited. It may be sufficient for relatively flat areas, but hills that produce significant roll and pitch rotations may cause beacons to be lost in blind spots. Using multiple cameras is the most practical way to further increase the field of

view. Multiple cameras can also produce a constant angular resolution across the field of view and they may be a more cost effective method of obtaining very high resolution.

7.2.3 INCREASING DYNAMIC PRECISION

Unsynchronized pixels will cause a decrease in position precision while in motion, especially with high resolution cameras and large distances between beacons. A global shutter and hardware triggers for multiple cameras will solve the problem, but they may add a lot of cost. Global shutters are typical of CCD image sensors, which are significantly more expensive than the CMOS sensors that generally lack global shutters. However, the random pixel access with CMOS sensors may help to alleviate synchronization issues and provide other benefits. A typical CMOS sensor may be limited to 30 fps, which generally means that pixels cannot be guaranteed to be synchronized to less than $1/30^{th}$ of a second, especially if multiple cameras are used without triggers. However, random pixel access allows CMOS sensors to specify a window of pixels and sample them at much higher frame rates, often 300 fps or more. Once beacons are located in the image(s), windowing could increase the synchronization by an order of magnitude.

7.2.4 INTEGRATION WITH OTHER SYSTEMS

There are many possible benefits of integration, particularly with a GPS unit and an IMU. A GPS unit and an IMU could provide an initial estimate of the position and orientation, which would allow beacons to be distinguished based on the directions to them alone. Otherwise they would have to be distinguished by using multiple colors, different modulation patterns, or some other means. It could also provide an estimate of the pixel coordinates of beacons, which would make it much easier to avoid false positives in the environment and greatly reduce the computational load.

The sensor fusion of GPS and dead reckoning was shown to be very effective in section 2.1.3. The fact that optical resection can readily provide 1 or 2 orders of magnitude greater sample rates than GPS could make sensor fusion even more valuable.

A GPS and IMU could also provide a check in the resection calculation, especially valuable when only three beacons are identified.

WORKS CITED

[1] National Agricultural Statistics Service, "2007 Census of Agriculture," U.S. Department of Agriculture, Washington D.C., 2007.

[2] T. Hague, J. Marchant and N. Tillet, "Ground based sensing systems for autonomous agricultural vehicles," *Computers and Electronics in Agriculture,* pp. 11-28, 2000.

[3] M. Lawrence, "Slate," 1 June 2012. [Online]. Available: http://www.slate.com/articles/technology/future_tense/2012/06/automated_farm_equipment_and_small_scale_farmers_.html. [Accessed 26 June 2012].

[4] L. Bedord, "Agriculture.com," Meredith Corporation, 10 August 2011. [Online]. Available: http://www.agriculture.com/machinery/farm-implements/planters/autonomous-plting-system_231-ar18253. [Accessed 27 June 2012].

[5] M. Chivers, "Differential GPS Explained," Esri, [Online]. Available: http://www.esri.com/news/arcuser/0103/differential1of2.html. [Accessed 8 October 2012].

[6] National Geodetic Survey, *Differential GPS,* National Oceanic and Atmosphere Administration.

[7] NSTB/WAAS T&E Team, "Wide-Area Augmentation System Performance Analysis Report," National Traffic Safety Board, Atlantic City, 2006.

[8] Trimble Navigation, Ltd., OmniSTAR, [Online]. Available: http://www.omnistar.com/SubscriptionServices/OmniSTARXP.aspx. [Accessed 10 July 2012].

[9] European Space Agency, "Real Time Kinematics," 2011. [Online]. Available: http://www.navipedia.net/index.php/Real_Time_Kinematics. [Accessed 8 October 2012].

[10] A. Stoll and H. Kutzbach, "Guidance of a Forage Harvester with GPS," *Precision Agriculture,* no. 2, pp. 281-291, 2000.

[11] M. Kise, N. Noguchi, K. Ishii and H. Terai, "Development of the agricultural autonomous tractor with an RTK-GPS and a FOG," in *4th IFAC Symposium on Intelligent Autonomous Vehicles*, 2001.

[12] M. Perez-Ruiz and D. Slaughter, "Automatic GPS-based intra-row weed knife control system for transplanted row crops," *Computers and Electronics in Agriculture,* vol. 80, no. January, pp. 41-49, 2012.

[13] M. Rodriguez and J. Gomez, "Analysis of three different Kalman filter implementations for agriculture vehicle

positioning," *The Open Agriculture Journal,* vol. 3, pp. 13-19, 2009.

[14] P. King, "A Low Cost Localization Solution Using a Kalman Filter for Data Fusion," Virginia Polytechnic Institute and State University, Blacksburg, 2008.

[15] G. Linsong, H. Yong, Z. Qin and H. Shufeng, "Real-Time Tractor Position Estimation System Using a Kalman Filter," *Computer Science and Automation Engineering,* vol. 18, no. 5, pp. 96-101, 2002.

[16] F. Caron, E. Duflos, D. Pomorski and P. Vanheeghe, "GPS/IMU data fusion using multisensor Kalman filtering: introduction of contextual aspects," *Information Fusion,* vol. 7, no. 2, pp. 221-230, 2006.

[17] Y. Li, J. Wang, C. Rizos, P. Mumford and W. Ding, "Low-cost Tightly Coupled GPS/INS Integration Based on a Nonlinear Kalman Filtering Design," in *Proceedings of the 2006 National Technical Meeting of The Institute of Navigation*, Monterey, 2006.

[18] J. Reid, Q. Zhang, N. Noguchi and M. Dickson, "Agriculture automatic guidance research in North America," *Computers and Electronics in Agriculture,* no. 25, pp. 155-167, 2000.

[19] T. Hague and N. Tillett, "A bandpass filter-based approach to crop row location and tracking," *Mechatronics,* vol. 11, no. 1, pp. 1-12, 2001.

[20] M. Kise, Q. Zhang and F. Mas, "A Stereovision-based Crop Row Detection Method for Tractor-automated Guidance," *Biosystems Engineering,* vol. 90, no. 4, pp. 357-367, 2005.

[21] M. Li, K. Imou, K. Wakabayashi and S. Yokoyama, "Review of research on agricultural vehicle autonomous guidance," *Int J Agric & Biol Eng,,* vol. 2, no. 3, pp. 1-16, 2009.

[22] The Economist, "Visible light communication - tripping the light fantastic," The Economist Newspaper, 28 January 2012. [Online]. Available: http://www.economist.com/node/21543470. [Accessed 8 October 2012].

[23] Y. H. Choi, I. H. Park, Y. H. Kim and J. Y. Kim, "Novel LBS Technique Based on Visible Light Communications," in *2012 IEEE International Conference on Consumer Electronics*, Las Vegas, 2012.

[24] B. Bai, G. Chen, Z. Xu and Y. Fan, "Visible Light Positioning based on LED Traffic Light and Photodiode," in *Vehicular Technology Conference*, San Francisco, 2011.

[25] Y. Kim, J. Hwang, J. Lee and M. Yoo, "Position Estimation Algorithm Based on Tracking of Received Light Intensity for Indoor Visible Light Communication Systems," in *The Third International Conference on Ubiquitous and Future Networks*, Dalian, China, 2011.

[26] IEEE, "Slideshow: How Kiva's Robotic Warehouse Works," July 2008. [Online]. Available:

http://spectrum.ieee.org/robotics/robotics-software/slideshow-how-kivas-robotic-warehouse-works/1. [Accessed 14 August 2012].

[27] R. Siegwart and D. Scaramuzza, "Autonomous Mobile Robots," [Online]. Available: http://www.asl.ethz.ch/education/master/mobile_robotics/Lecture7b.pdf. [Accessed 8 October 2012].

[28] M. Li, K. Imou and K. Wakabayashi, "3D Positioning for Mobile Robot Using Omnidirectional Vision," in *2010 International Conference on Intelligent Computation Technology and Automation*, Changsha, China, 2010.

[29] M. Li, Z. Liu, J. Huang, S. H. Dai, K. Wakabayashi and K. Imou, "Artificial Landmark Positioning System Using Omnidirectional Vision for Agricultural," in *2012 International Conference on Intelligent System Design and Engineering Application*, Sanya, Hainan, China, 2012.

[30] IC Insights, Inc., "Research Bulletin," 10 May 2012. [Online]. Available: http://www.icinsights.com/data/articles/documents/428.pdf. [Accessed 10 July 2012].

[31] M. Loose, A. Hoffman and V. Suntharalingam, "CMOS Detector Technology," Scientific Detector Workshop, Sicily, Italy, 2005.

[32] S. Moore and J. Simon, "The Greatest Century That Ever Was: 25 Miraculous Trends of the Past 100 Years," cato.org, 1999.

[33] Wikipedia, "Snellius–Pothenot problem," Wikimedia Foundation, [Online]. Available: http://en.wikipedia.org/wiki/Snellius%E2%80%93Pothenot_problem. [Accessed 3 October 2012].

[34] Wikimedia Foundation, Inc., "Snellius–Pothenot problem," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Snellius%E2%80%93Pothenot_problem#Solution_algorithm. [Accessed 8 August 2012].

[35] J. Awange, E. Grafarend, B. Palancz and P. Zaletnyik, "Geodetic Resection," in *Algebraic Geodesy and Geoinformatics, 2nd ed.*, Springer, 2010, p. 235.

[36] R. Haralick, C.-N. Lee, K. Ottenberg and M. Nolle, "Review and Analysis of Solutions of the Three Point Perspective Pose Estimation Problem," *International Journal of Comptuer Vision,* vol. 3, no. 13, pp. 331-356, 1994.

[37] D. Dementhon and L. Davis, "Model-based Object Pose in 25 Lines of Code," *International Journal of Computer Vision,* vol. 15, no. 1-2, pp. 123-141, 1995.

[38] Wikimedia Foundation, Inc., "Trilateration," Wikipedia, [Online]. Available:

http://en.wikipedia.org/wiki/Trilateration. [Accessed 27 August 2012].

[39] Wikimedia Foundation, Inc., "Pinhole camera model," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Pinhole_camera_model. [Accessed 22 August 2012].

[40] BenFrantzDale. [Online]. Available: http://commons.wikimedia.org/wiki/File:Big_pinhole_with_lens.svg. [Accessed 4 October 2012].

[41] WolfWings. [Online]. Available: http://en.wikipedia.org/wiki/File:Barrel_distortion.svg. [Accessed 4 October 2012].

[42] Z. Zhang, "A flexible new technique for camera calibration," *EEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 11, no. 22, pp. 1330-1334, 2000.

[43] D. Scaramuzza, A. Martinelli and R. Siegwart, "A Flexible Technique for Accurate Omnidirectional Camera Calibration and Structure from Motion," in *IEEE International Conference of Vision Systems*, New York, 2006.

[44] D. Scaramuzza and A. Martinelli, "A Toolbox for Easy Calibrating Omnidirectional Cameras," in *IEEE International Conference on Intelligent Robots and Systems*, Beijing, 2006.

[45] M. S. D. S. R. Rufli, "Automatic Detection of Checkerboards on Blurred and Distorted Images," in *International Conference on Intelligent Robots and Systems*, France, 2008.

APPENDIX A: DEVICES AND COMPONENTS

A.1 SPHERICAL MIRROR

Table A-1. Spherical mirror specifications.

Diameter (mm)	12.00
Focal Length (mm)	-7.75
Radius (mm)	15.50
Edge Thickness (mm)	1.79
Center Thickness (mm)	3.00
Diameter Tolerance (mm)	+0.0/-0.10
Focal Length Tolerance (%)	±2
Surface quality	60-40
Clear Aperture (%)	90
Typical Energy Density Limit	0.2 J/cm^2 @ 532nm, 10ns

A.2 CAMERA

Table A-2. Camera specifications.

Product Name	Microsoft® LifeCam HD-5000
Sensor	CMOS
Resolution	1280x720
Imaging rate (fps)	30
Field of view (°)	66 diagonal
Color depth	24 bit
Length (mm)	37.8
Width (mm)	40.8
Depth (mm)	109
Weight (g)	96.5
Cable Length (mm)	1,829
Interface	USB 2.0

A.3 HIGH POWER LED

Table A-3. HPLED specifications.

Product Name	Cree® XLamp™ XT-E
Wavelength (nm)	455
Viewing angle, FWHM (°)	140
Forward voltage, typ (V)	2.85
Forward voltage, max (V)	3.4
Junction temperature, typ (°C)	85

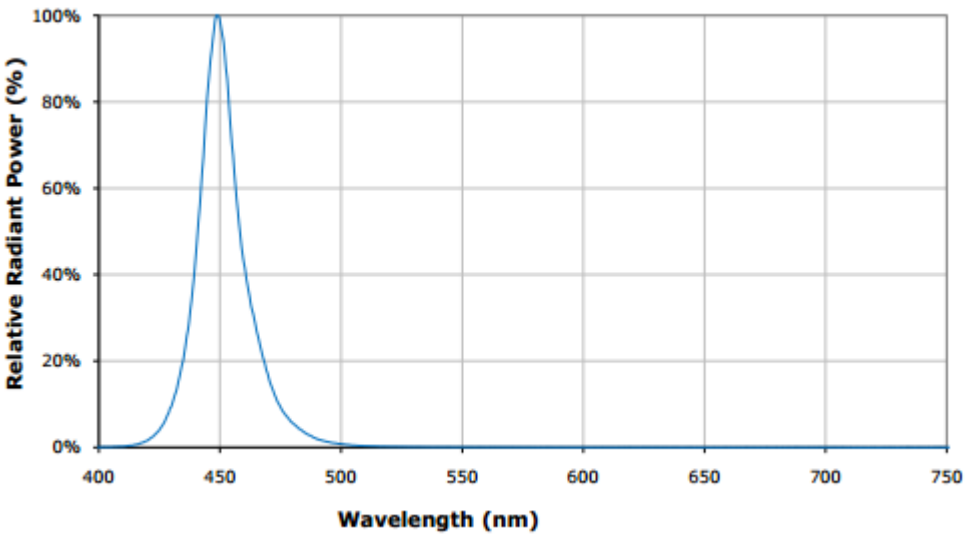


Figure A-1. Relative spectral power distribution.

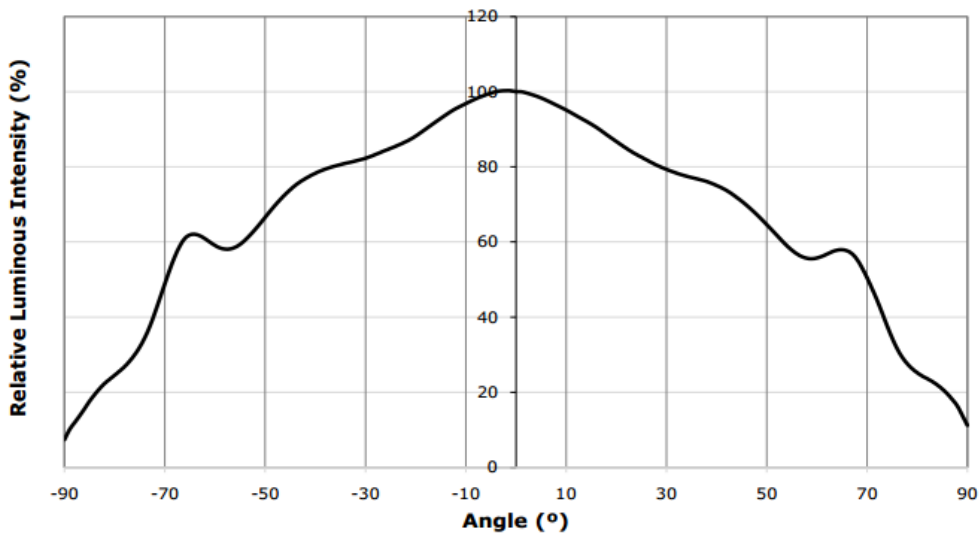


Figure A-2. Relative spatial intensity.

APPENDIX B: C++ CODE

B.1 MAIN.H

```
#include <string>
#include <atlstr.h>
#include <streams.h>
#include <fstream>
#include <dshow.h>
#include <qedit.h>
#include <opencv/cv.h>
#include <opencv/highgui.h>
#include <BlobResult.h>
#include <Eigen\Dense>
#include <math.h>

using namespace std;
using namespace Eigen;

#define PI 3.14159265
```

B.2 BEACONANGLES.H

```
#include "main.h"

bool getAngles(IplImage* frame1, IplImage* frame2, const Matrix2d& A, const Vector2d& T,
double coefficients[5], double angles[6]);
Vector3d get3dVector(const Vector2d& beacon, const Matrix2d& A, const Vector2d& T, double
coefficients[5]);
```

B.3 DSHOW.H

```
#include "main.h"

//      Set camera properties, start capturing
void DShowStartCapture();

//      Grab frame, convert to openCV format
IplImage *GetFrameDShow2OpenCV();

//      Release capture
void DshowRelease();
```

B.4 RESECTION.H

```
#include "main.h"

void getPosition(double 67eacon[3], double 67eacon[3], double 67eacon[3], double
angles[6], double position[4]);
```

```
void trilateration(double sphere1[4], double sphere2[4], double sphere3[4], double
position[4]);
void getSphere(double beacon1[3], double beacon2[3], double angle, double sphere[4]);
double getLength(double point1[3], double point2[3]);
double dotProduct(double point1[3], double point2[3]);
void crossProduct(double vector1[3], double vector2[3], double product[3]);
void getVector(double point1[3], double point2[3], double vector[3]);
int getFurthestBeacon(double distances[4]);

void Nrmethod(double estimate[3], double APB, double APC, double BPC, double AB, double
AC, double BC, double lengths[4]);
void NRmethodABC(double estimate[3], double APB, double APC, double BPC, double AB,
double AC, double BC, double lengths[4]);
void NRmethodBCD(double estimate[4], double BPC, double CPD, double BPD, double BC,
double CD, double BD, double lengths[4]);
void NRmethodCDA(double estimate[4], double CPD, double APD, double APC, double CD,
double AD, double AC, double lengths[4]);
void NRmethodDAB(double estimate[4], double APD, double APB, double BPD, double AB,
double AD, double BD, double lengths[4]);
void Nrmethod4(double estimate[4], double APB, double BPC, double CPD, double APD, double
AB, double BC, double CD, double AD, double lengths[4]);
```

B.5 MAIN.CPP

```
#include “beaconAngles.h”
#include “Dshow.h”
#include “main.h”
#include “resection.h”

void uSleep(int waitTime) {
    LARGE_INTEGER start, current, frequency;
       QueryPerformanceCounter(&start);
       QueryPerformanceFrequency(&frequency);

    do {
        QueryPerformanceCounter(&current);
    } while((current.QuadPart – start.QuadPart) * 1000000.0 / frequency.QuadPart <
waitTime);
}
double diffclock(clock_t clock1,clock_t clock2)
{
       double diffticks=clock1-clock2;
       double diffms=(diffticks*1000)/CLOCKS_PER_SEC;
       return diffms;
}
std::string get_extension(std::string const& x, char ext_sep = ‘.’) {
     return x.substr(x.find_last_of(ext_sep) + 1); // no error checking
}
std::string strip_extension(std::string const& x, char ext_sep = ‘.’) {
     return x.substr(0, x.find_last_of(ext_sep));
}
std::string append_number(std::string const& x, unsigned int num, char sep1 = ‘3’, char
sep2 = ‘3’, char sep3 = ‘_’) {
        std::stringstream s;
        s << strip_extension(x) << sep1 << sep2 << sep3 << num << ‘.’ <<
get_extension(x);
```

```
        return s.str();
}

int main()
{
        //---INPUTS REQUIRED---//
        //      1.      Filename
        //      2.   Lengths AB, BC, AC
        //      3.      69eacon and 69eacon positions (if required)
        //      4.      Measured distances from 69eacon and 69eacon


        //      data recording
        ofstream myfile;
        myfile.open ("test3d.txt");

        double AB = 705.5;
        double BC = 682;
        double AC = 982.5;
        double angleB = acos((pow(AB,2)+pow(BC,2)-pow(AC,2))/(2*AB*BC));

        //      beacons
        double 69eacon[3] = {0,0,0.6};                          //      must be {0,0,0}

        double 69eacon[3] = {0,AB,0.6};                         //      must be {0,y,0}
        double 69eacon[3];
        if (angleB == PI/2)
        {
                69eacon[0] = BC;
                69eacon[1] = 69eacon[1];
                69eacon[2] = 0.6;
        }else if (angleB > PI/2)
        {
                69eacon[0] = BC*cos(angleB-PI/2);
                69eacon[1] = 69eacon[1] + BC*sin(angleB-PI/2);
                69eacon[2] = 0.6;
        }else if (angleB < PI/2)
        {
                69eacon[0] = BC*cos(PI/2-angleB);
                69eacon[1] = 69eacon[1] – BC*sin(PI/2-angleB);
                69eacon[2] = 0.6;
        }

        //      measured position
        double distA = 44*12+6.5;
        double distB = 37*12+11;
        double yActual = (pow(distA,2) – pow(distB,2) + pow(69eacon[1],2))/(2*69eacon[1]);
        double xActual = pow(pow(distA,2) – pow(yActual,2) ,0.5);
        myfile << xActual << " " << yActual << endl;

        //      camera calibration
        Matrix2d A;          A(0,0) = 0.999780;   A(0,1) = -0.000091;
                                     A(1,0) = -0.000002;  A(1,1) = 1;

        Vector2d T;          T(0) = 361.319461;
                                     T(1) = 638.741876;
```

```
	double coefficients[5] = {-239.7201, 0, 0.005848336, -0.00002643976,
0.00000005452470};

	double angles[6];		//	0=APB, 1=BPC, 2=CPD, 3=APD, 4=APC, 5=BPD
	double position[4];		//	0=x, 1=y, 2=z, 3=residual

	DshowStartCapture();
	uSleep(500000);

	LARGE_INTEGER start, last, frequency;

	IplImage* img = 0;
	IplImage* frame1 = cvCreateImage(cvSize(800,720), IPL_DEPTH_8U, 3);
	IplImage* frame2 = cvCreateImage(cvSize(800,720), IPL_DEPTH_8U, 3);
	QueryPerformanceCounter(&start);
	QueryPerformanceCounter(&last);
	QueryPerformanceFrequency(&frequency);
	bool anglesFound = false;
	int c=0;
	int count =0;

	while(count < 900)
	{
		count++;
		for (int i=0; i<1; i++)
		{
			uSleep(33000 - (last.QuadPart - start.QuadPart)* 1000000.0 /
frequency.QuadPart);
			QueryPerformanceCounter(&start);
			img = GetFrameDShow2OpenCV();

			cvSetImageROI(img, cvRect(250, 0, 800, 720));

			if(i==0)
			{
				cvCopy(img, frame1);
			}else
			{
				cvCopy(img, frame2);
			}
		}
		anglesFound = getAngles(frame1, frame2, A, T, coefficients, angles);
		if (anglesFound == true)
		{
			getPosition(70eacon, 70eacon, 70eacon, angles, position);
			cout << position[0] << " " << position[1] << " " << position[2] <<
endl;
			myfile << position[0] << " " << position[1] << " " << position[2] <<
" " << position[3] << " " << angles[0] << " " << angles[1] <<  " "  << " " << angles[4]
<< endl;
		}

		cvWaitKey(1);
		QueryPerformanceCounter(&last);
	}

	return 0;
}
```

B.6 BEACONANGLES.CPP

```
#include "beaconAngles.h"

bool getAngles(IplImage* frame1, IplImage* frame2, const Matrix2d& A, const Vector2d& T,
double coefficients[5], double angles[6])
{
	bool anglesFound = false;

	//	find beacons in image
	Vector2d 71eacon;	71eacon(0) = 500;	71eacon(1) = 641;
	Vector2d 71eacon;	71eacon(0) = 350;	71eacon(1) = 699;
	Vector2d 71eacon;	71eacon(0) = 650;	71eacon(1) = 299;

	IplImage* frame1Thresh = cvCreateImage(cvSize(800,720), IPL_DEPTH_8U, 1);
	IplImage* frame1ThreshDilate = cvCreateImage(cvSize(800,720), IPL_DEPTH_8U, 1);

	cvInRangeS(frame1, cvScalar(160,0,0), cvScalar(255,50,30), frame1Thresh);
	cvDilate(frame1Thresh, frame1ThreshDilate, 0, 5);

	double point[3][2];

	CblobResult blobs = CblobResult(frame1ThreshDilate, NULL, 0);
	CvRect blobBox;
	int num = blobs.GetNumBlobs();
	if (num == 3)
	{
		anglesFound = true;
		for (int I = 0; I < 3; i++)
		{
			Cblob *currentBlob = blobs.GetBlob(i);
			blobBox = currentBlob->GetBoundingBox();
			int xCenter = blobBox.x + blobBox.width/2;
			int yCenter = blobBox.y + blobBox.height/2;

			int blue = 0;
			int bright1 = 0;
			int bright1coord[2] = {0,0};
			int bright2 = 0;
			int bright2coord[2] = {0,0};
			int bright3 = 0;
			int bright3coord[2] = {0,0};
			int bright4 = 0;
			int bright4coord[2] = {0,0};
			int bright5 = 0;
			int bright5coord[2] = {0,0};
			int bright6 = 0;
			int bright6coord[2] = {0,0};
			int bright7 = 0;
			int bright7coord[2] = {0,0};
			int bright8 = 0;
			int bright8coord[2] = {0,0};
			int bright9 = 0;
			int bright9coord[2] = {0,0};

			for (int x = xCenter-5; x < xCenter+6; x++)
			{
```

```
                        for (int y = yCenter-5; y < yCenter+6; y++)
                        {
                                blue = ((uchar*)(frame1->imageData + frame1-
>widthStep*y))[x*3] + ((uchar*)(frame2->imageData + frame1->widthStep*y))[x*3];
                                if (blue > bright1)
                                {
                                        bright9 = bright8;
                                        bright9coord[0] = bright8coord[0];
                                        bright9coord[1] = bright8coord[1];

                                        bright8 = bright7;
                                        bright8coord[0] = bright7coord[0];
                                        bright8coord[1] = bright7coord[1];

                                        bright7 = bright6;
                                        bright7coord[0] = bright6coord[0];
                                        bright7coord[1] = bright6coord[1];

                                        bright6 = bright5;
                                        bright6coord[0] = bright5coord[0];
                                        bright6coord[1] = bright5coord[1];

                                        bright5 = bright4;
                                        bright5coord[0] = bright4coord[0];
                                        bright5coord[1] = bright4coord[1];

                                        bright4 = bright3;
                                        bright4coord[0] = bright3coord[0];
                                        bright4coord[1] = bright3coord[1];

                                        bright3 = bright2;
                                        bright3coord[0] = bright2coord[0];
                                        bright3coord[1] = bright2coord[1];

                                        bright2 = bright1;
                                        bright2coord[0] = bright1coord[0];
                                        bright2coord[1] = bright1coord[1];

                                        bright1 = blue;
                                        bright1coord[0] = x;
                                        bright1coord[1] = y;
                                }else if (blue > bright2)
                                {
                                        bright9 = bright8;
                                        bright9coord[0] = bright8coord[0];
                                        bright9coord[1] = bright8coord[1];

                                        bright8 = bright7;
                                        bright8coord[0] = bright7coord[0];
                                        bright8coord[1] = bright7coord[1];

                                        bright7 = bright6;
                                        bright7coord[0] = bright6coord[0];
                                        bright7coord[1] = bright6coord[1];

                                        bright6 = bright5;
                                        bright6coord[0] = bright5coord[0];
                                        bright6coord[1] = bright5coord[1];
```

```
        bright5 = bright4;
        bright5coord[0] = bright4coord[0];
        bright5coord[1] = bright4coord[1];

        bright4 = bright3;
        bright4coord[0] = bright3coord[0];
        bright4coord[1] = bright3coord[1];

        bright3 = bright2;
        bright3coord[0] = bright2coord[0];
        bright3coord[1] = bright2coord[1];

        bright2 = blue;
        bright2coord[0] = x;
        bright2coord[1] = y;
}else if (blue > bright3)
{
        bright9 = bright8;
        bright9coord[0] = bright8coord[0];
        bright9coord[1] = bright8coord[1];

        bright8 = bright7;
        bright8coord[0] = bright7coord[0];
        bright8coord[1] = bright7coord[1];

        bright7 = bright6;
        bright7coord[0] = bright6coord[0];
        bright7coord[1] = bright6coord[1];

        bright6 = bright5;
        bright6coord[0] = bright5coord[0];
        bright6coord[1] = bright5coord[1];

        bright5 = bright4;
        bright5coord[0] = bright4coord[0];
        bright5coord[1] = bright4coord[1];

        bright4 = bright3;
        bright4coord[0] = bright3coord[0];
        bright4coord[1] = bright3coord[1];

        bright3 = blue;
        bright3coord[0] = x;
        bright3coord[1] = y;
}else if (blue > bright4)
{
        bright9 = bright8;
        bright9coord[0] = bright8coord[0];
        bright9coord[1] = bright8coord[1];

        bright8 = bright7;
        bright8coord[0] = bright7coord[0];
        bright8coord[1] = bright7coord[1];

        bright7 = bright6;
        bright7coord[0] = bright6coord[0];
        bright7coord[1] = bright6coord[1];
```

```
        bright6 = bright5;
        bright6coord[0] = bright5coord[0];
        bright6coord[1] = bright5coord[1];

        bright5 = bright4;
        bright5coord[0] = bright4coord[0];
        bright5coord[1] = bright4coord[1];

        bright4 = blue;
        bright4coord[0] = x;
        bright4coord[1] = y;
}else if (blue > bright5)
{
        bright9 = bright8;
        bright9coord[0] = bright8coord[0];
        bright9coord[1] = bright8coord[1];

        bright8 = bright7;
        bright8coord[0] = bright7coord[0];
        bright8coord[1] = bright7coord[1];

        bright7 = bright6;
        bright7coord[0] = bright6coord[0];
        bright7coord[1] = bright6coord[1];

        bright6 = bright5;
        bright6coord[0] = bright5coord[0];
        bright6coord[1] = bright5coord[1];

        bright5 = blue;
        bright5coord[0] = x;
        bright5coord[1] = y;
}else if (blue > bright6)
{
        bright9 = bright8;
        bright9coord[0] = bright8coord[0];
        bright9coord[1] = bright8coord[1];

        bright8 = bright7;
        bright8coord[0] = bright7coord[0];
        bright8coord[1] = bright7coord[1];

        bright7 = bright6;
        bright7coord[0] = bright6coord[0];
        bright7coord[1] = bright6coord[1];

        bright6 = blue;
        bright6coord[0] = x;
        bright6coord[1] = y;
}else if (blue > bright7)
{
        bright9 = bright8;
        bright9coord[0] = bright8coord[0];
        bright9coord[1] = bright8coord[1];

        bright8 = bright7;
        bright8coord[0] = bright7coord[0];
```

```
                                        bright8coord[1] = bright7coord[1];

                                        bright7 = blue;
                                        bright7coord[0] = x;
                                        bright7coord[1] = y;
                                    }else if (blue > bright8)
                                    {
                                        bright9 = bright8;
                                        bright9coord[0] = bright8coord[0];
                                        bright9coord[1] = bright8coord[1];

                                        bright8 = blue;
                                        bright8coord[0] = x;
                                        bright8coord[1] = y;
                                    }else if (blue > bright9)
                                    {
                                        bright9 = blue;
                                        bright9coord[0] = x;
                                        bright9coord[1] = y;
                                    }

                                }
                            }

                            double xEst = (double(bright1)*double(bright1coord[0]) +
double(bright2)*double(bright2coord[0]) + double(bright3)*double(bright3coord[0]) +
double(bright4)*double(bright4coord[0]) + double(bright5)*double(bright5coord[0]) +
double(bright6)*double(bright6coord[0]) + double(bright7)*double(bright7coord[0]) +
double(bright8)*double(bright8coord[0]) +
double(bright9)*double(bright9coord[0]))/(double(bright1+bright2+bright3+bright4+bright5+
bright6+bright7+bright8+bright9));
                            double yEst = (double(bright1)*double(bright1coord[1]) +
double(bright2)*double(bright2coord[1]) + double(bright3)*double(bright3coord[1]) +
double(bright4)*double(bright4coord[1]) + double(bright5)*double(bright5coord[1]) +
double(bright6)*double(bright6coord[1]) + double(bright7)*double(bright7coord[1]) +
double(bright8)*double(bright8coord[1]) +
double(bright9)*double(bright9coord[1]))/(double(bright1+bright2+bright3+bright4+bright5+
bright6+bright7+bright8+bright9));
                            int xEstInt = floor(xEst + 0.5);
                            int yEstInt = floor(yEst + 0.5);
                            ((uchar*)(frame1->imageData + frame1->widthStep*yEstInt))[xEstInt*3]
= 0;
                            ((uchar*)(frame1->imageData + frame1-
>widthStep*yEstInt))[xEstInt*3+1] = 0;
                            ((uchar*)(frame1->imageData + frame1-
>widthStep*yEstInt))[xEstInt*3+2] = 255;

                            point[i][0] = xEst;  point[i][1] = yEst;
                        }
                    }

                    cvNamedWindow("projWindow", CV_WINDOW_AUTOSIZE);
                    cvShowImage("projWindow", frame1);

                    //     determine beacon points by assuming we'll always be aimed at B, with A on
the left and C on the right
                    double yAbsMin = 800;
                    double yMin = 800;
```

```
	double yMax = 0;
	for (int i=0; i<3; i++)
	{
		if (abs(point[i][1]-360) < yAbsMin)
		{
			yAbsMin = abs(point[i][1]-360);
			76eacon(0) = point[i][1];
			76eacon(1) = point[i][0] + 250;
		}
		if (point[i][1] < yMin)
		{
			yMin = point[i][1];
			76eacon(0) = point[i][1];
			76eacon(1) = point[i][0] + 250;
		}
		if (point[i][1] > yMax)
		{
			yMax = point[i][1];
			76eacon(0) = point[i][1];
			76eacon(1) = point[i][0] + 250;
		}
	}

	//	get angles from beacon image points
	Vector3d pA = get3dVector(76eacon, A, T, coefficients);
	Vector3d pB = get3dVector(76eacon, A, T, coefficients);
	Vector3d pC = get3dVector(76eacon, A, T, coefficients);

	angles[0] = acos(pA.dot(pB));
	angles[1] = acos(pB.dot(pC));
	angles[4] = acos(pA.dot(pC));

	cvReleaseImage(&frame1Thresh);
	cvReleaseImage(&frame1ThreshDilate);

	return anglesFound;
}


Vector3d get3dVector(const Vector2d& beacon, const Matrix2d& A, const Vector2d& T, double
coefficients[5])
{
	Vector2d beaconAdj = A.inverse()*(beacon-T);
	double p = pow(pow(beaconAdj(0),2) + pow(beaconAdj(1),2), 0.5);
	Vector3d beacon3d;
	beacon3d(0) = beaconAdj(0);
	beacon3d(1) = beaconAdj(1);
	beacon3d(2) = coefficients[0] + coefficients[1]*p + coefficients[2]*pow(p,2) +
coefficients[3]*pow(p,3) + coefficients[4]*pow(p,4);

	return beacon3d.normalized();
}
```

B.7 DSHOW.CPP

```
#include “Dshow.h”


//     for playing
IGraphBuilder *pGraphBuilder;
IcaptureGraphBuilder2 *pCaptureGraphBuilder2;
ImediaControl *pMediaControl = NULL;


IbaseFilter *pDeviceFilter_0 = NULL;
IbaseFilter *m_pGrabber_0 = NULL;

//     select camera
IcreateDevEnum *pCreateDevEnum = NULL;
IenumMoniker *pEnumMoniker = NULL;
Imoniker *pMoniker = NULL;

HRESULT hr;
IsampleGrabber *m_pGrabberSettings_0 = NULL;
long pBufferSize = 2764800;
unsigned char* pBuffer_0 = new unsigned char[pBufferSize];  //used to = 0
IplImage* img_0 = cvCreateImage(cvSize(1280,720),IPL_DEPTH_8U,3);


void setCameraMode(IcaptureGraphBuilder2 *pCaptureGraphBuilder2, IAMStreamConfig
*pConfig, IbaseFilter *pDeviceFilter, HRESULT hr)
{
              // Set res, frame rate, and color mode
              hr = CoInitialize(0);
              hr = pCaptureGraphBuilder2->FindInterface(&PIN_CATEGORY_CAPTURE, 0,
pDeviceFilter, IID_IAMStreamConfig, (void**)&pConfig);


              int iCount = 0, iSize = 0;
              hr = pConfig->GetNumberOfCapabilities(&iCount, &iSize);

              // Check the size to make sure we pass in the correct structure.
              If (iSize == sizeof(VIDEO_STREAM_CONFIG_CAPS))
              {
                     // Use the video capabilities structure.

                     For (int iFormat = 0; iFormat < iCount; iFormat++)
                     {
                            VIDEO_STREAM_CONFIG_CAPS scc;
                            AM_MEDIA_TYPE *pmtConfig;
                            hr = pConfig->GetStreamCaps(iFormat, &pmtConfig, (BYTE*)&scc);
                            if (SUCCEEDED(hr))
                            {
                                   /* Examine the format, and possibly use it. */
                                   if ((pmtConfig->majortype == MEDIATYPE_Video) &&
                                          (pmtConfig->subtype == MEDIASUBTYPE_RGB24) &&
                                          (pmtConfig->formattype == FORMAT_VideoInfo) &&
                                          (pmtConfig->cbFormat >= sizeof
(VIDEOINFOHEADER)) &&
                                          (pmtConfig->pbFormat != NULL))
                                   {
```

```
                                        VIDEOINFOHEADER *pVih =
(VIDEOINFOHEADER*)pmtConfig->pbFormat;
                                        // pVih contains the detailed format
information.
                                        LONG lWidth = pVih->bmiHeader.biWidth;
                                        LONG lHeight = pVih->bmiHeader.biHeight;
                                }
                                if (iFormat == 26) { //2 = '1280x720YUV' YUV, 22 =
'1280x800YUV', 26 = '1280x720RGB'
                                        hr = pConfig->SetFormat(pmtConfig);
                                }
                                // Delete the media type when you are done.
                                DeleteMediaType(pmtConfig);
                        }
                }
            }
}
void setCameraControl(IbaseFilter *pDeviceFilter, HRESULT hr, int exposure, int focus)
{
        // Query the capture filter for the IAMCameraControl interface.
        IAMCameraControl *pCameraControl = 0;
        hr = pDeviceFilter->QueryInterface(IID_IAMCameraControl, (void**)&pCameraControl);
        if (FAILED(hr))
        {
                // The device does not support IAMCameraControl
        }
        else
        {
                long Min, Max, Step, Default, Flags, Val;

                // Get the range and default values
                hr = pCameraControl->GetRange(CameraControl_Exposure, &Min, &Max, &Step,
&Default, &Flags);
                hr = pCameraControl->GetRange(CameraControl_Focus, &Min, &Max, &Step,
&Default, &Flags);
                if (SUCCEEDED(hr))
                {
                        hr = pCameraControl->Set(CameraControl_Exposure, exposure,
CameraControl_Flags_Manual ); // Min = -11, Max = 1, Step = 1
                        hr = pCameraControl->Set(CameraControl_Focus, focus,
CameraControl_Flags_Manual );
                }
        }
}
void setCameraProperties(IbaseFilter *pDeviceFilter, HRESULT hr, int brightness, int
backLightCompensation, int contrast, int saturation, int sharpness, int whiteBalance)
{
        // Query the capture filter for the IAMVideoProcAmp interface.
        IAMVideoProcAmp *pProcAmp = 0;
        hr = pDeviceFilter->QueryInterface(IID_IAMVideoProcAmp, (void**)&pProcAmp);
        if (FAILED(hr))
        {
                // The device does not support IAMVideoProcAmp
        }
        else
        {
                long Min, Max, Step, Default, Flags, Val;
```

```
            // Get the range and default values
            hr = pProcAmp->GetRange(VideoProcAmp_Brightness, &Min, &Max, &Step,
&Default, &Flags);
            hr = pProcAmp->GetRange(VideoProcAmp_BacklightCompensation, &Min, &Max,
&Step, &Default, &Flags);
            hr = pProcAmp->GetRange(VideoProcAmp_Contrast, &Min, &Max, &Step, &Default,
&Flags);
            hr = pProcAmp->GetRange(VideoProcAmp_Saturation, &Min, &Max, &Step,
&Default, &Flags);
            hr = pProcAmp->GetRange(VideoProcAmp_Sharpness, &Min, &Max, &Step,
&Default, &Flags);
            hr = pProcAmp->GetRange(VideoProcAmp_WhiteBalance, &Min, &Max, &Step,
&Default, &Flags);
            if (SUCCEEDED(hr))
            {
                    hr = pProcAmp->Set(VideoProcAmp_Brightness, 130,
VideoProcAmp_Flags_Manual);
                    hr = pProcAmp->Set(VideoProcAmp_BacklightCompensation, 0,
VideoProcAmp_Flags_Manual);
                    hr = pProcAmp->Set(VideoProcAmp_Contrast, 4,
VideoProcAmp_Flags_Manual);
                    hr = pProcAmp->Set(VideoProcAmp_Saturation, 100,
VideoProcAmp_Flags_Manual);
                    hr = pProcAmp->Set(VideoProcAmp_Sharpness, 0,
VideoProcAmp_Flags_Manual);
                    hr = pProcAmp->Set(VideoProcAmp_WhiteBalance, 2800,
VideoProcAmp_Flags_Manual);
            }
        }
}
Ipin *GetPin(IbaseFilter *pFilter, PIN_DIRECTION PinDir)
{
    BOOL       bFound = FALSE;
    IenumPins  *pEnum;
    Ipin       *pPin;

    pFilter->EnumPins(&pEnum);
    while(pEnum->Next(1, &pPin, 0) == S_OK)
    {
        PIN_DIRECTION PinDirThis;
        pPin->QueryDirection(&PinDirThis);
        if (bFound = (PinDir == PinDirThis))
            break;
        pPin->Release();
    }
    pEnum->Release();
    return (bFound ? pPin : 0);
}



void DshowStartCapture()
{

        ULONG nFetched = 0;
        // initialize COM
        CoInitialize(NULL);
```

```
	// selecting a device
	// Create CreateDevEnum to list device
	std::string USB1 =
"\\\\?\\usb#vid_045e&pid_076d&mi_00#7&1ba27d43&0&0000#{65e8773d-8f56-11d0-a3b9-
00a0c9223196}\\global";  //left port
	//std::string USB2 =
"\\\\?\\usb#vid_045e&pid_076d&mi_00#7&8e8cc9e&0&0000#{65e8773d-8f56-11d0-a3b9-
00a0c9223196}\\global";  //left port


	CoCreateInstance(CLSID_SystemDeviceEnum, NULL, CLSCTX_INPROC_SERVER,
IID_IcreateDevEnum, (PVOID *)&pCreateDevEnum);

	// Create EnumMoniker to list VideoInputDevice
	pCreateDevEnum->CreateClassEnumerator(CLSID_VideoInputDeviceCategory,
&pEnumMoniker, 0);
	if (pEnumMoniker == NULL) {
	// this will be shown if there is no capture device
	printf("no device\n");

	}

	// reset EnumMoniker
	pEnumMoniker->Reset();

	// get each Moniker
	while (pEnumMoniker->Next(1, &pMoniker, &nFetched) == S_OK) {
		IpropertyBag *pPropertyBag;
		TCHAR devname[256];
		TCHAR devpath[256];

		// bind to IpropertyBag
		pMoniker->BindToStorage(0, 0, IID_IpropertyBag, (void **)&pPropertyBag);

		VARIANT var;

		// get FriendlyName
		var.vt = VT_BSTR;
		pPropertyBag->Read(L"FriendlyName", &var, 0);
		WideCharToMultiByte(CP_ACP, 0, var.bstrVal, -1, devname, sizeof(devname),
0, 0);
		VariantClear(&var);

		// get DevicePath
		// DevicePath : A unique string
		var.vt = VT_BSTR;

		pPropertyBag->Read(L"DevicePath", &var, 0);

		WideCharToMultiByte(CP_ACP, 0, var.bstrVal, -1, devpath, sizeof(devpath),
0, 0);
		std::string devpathString = devpath;

		if ((devname[0] == 'M') && (devname[21] == '5') && (devpathString == USB1
))
		{
			pMoniker->BindToObject(0, 0, IID_IbaseFilter,
(void**)&pDeviceFilter_0 );
```

```
            }


            pMoniker->Release();
            pPropertyBag->Release();

            if (pDeviceFilter_0 == NULL)
            {
                    MessageBox(NULL, “No MS HD-5000 cameras found”, “No cameras”,
MB_OK);

            }

        }

        // create FilterGraph and CaptureGraphBuilder2
        CoCreateInstance(CLSID_FilterGraph, NULL, CLSCTX_INPROC, IID_IgraphBuilder,
(LPVOID *)&pGraphBuilder);
        CoCreateInstance(CLSID_CaptureGraphBuilder2, NULL, CLSCTX_INPROC,
IID_IcaptureGraphBuilder2, (LPVOID *)&pCaptureGraphBuilder2);

        hr = CoInitialize(0);
        IAMStreamConfig *pConfig = NULL;
        setCameraMode(pCaptureGraphBuilder2,  pConfig,  pDeviceFilter_0,  hr);
                        //     FPS, Res, color mode
        setCameraControl(pDeviceFilter_0, hr, -4, 40);
                                        //     Exposure, focus
        setCameraProperties(pDeviceFilter_0, hr, 130, 0, 4, 100, 0, 2800);
                        //     Brightness, saturation, etc


        //     set grabber properties
        AM_MEDIA_TYPE mt;
        hr = CoCreateInstance(CLSID_SampleGrabber, NULL, CLSCTX_INPROC_SERVER,
IID_IbaseFilter, (void**)&m_pGrabber_0); // create IsampleGrabber
        pCaptureGraphBuilder2->SetFiltergraph(pGraphBuilder);
                                                                        // set FilterGraph
        pGraphBuilder->QueryInterface(IID_ImediaControl, (LPVOID *)&pMediaControl);
                                                            // get MediaControl interface

        m_pGrabber_0->QueryInterface(IID_IsampleGrabber, (void**)&m_pGrabberSettings_0);
    ZeroMemory(&mt, sizeof(AM_MEDIA_TYPE));
    mt.majortype = MEDIATYPE_Video;
    mt.subtype = MEDIASUBTYPE_RGB24;
    hr = m_pGrabberSettings_0->SetMediaType(&mt);
    if (FAILED(hr))
    {

    }
    hr = m_pGrabberSettings_0->SetOneShot(FALSE);
    hr = m_pGrabberSettings_0->SetBufferSamples(TRUE);

        //     build filter graph
        pGraphBuilder->AddFilter(pDeviceFilter_0, L”Device Filter”);
        pGraphBuilder->AddFilter(m_pGrabber_0, L”Sample Grabber”);
        Ipin* pSourceOut_0 = GetPin(pDeviceFilter_0, PINDIR_OUTPUT);
        Ipin* pGrabberIn_0 = GetPin(m_pGrabber_0, PINDIR_INPUT);
        pGraphBuilder->Connect(pSourceOut_0, pGrabberIn_0);
```

```
    // start playing
       pMediaControl->Run();

}
IplImage *GetFrameDShow2OpenCV()
{
       //pBuffer_0 = new unsigned char[pBufferSize]; //memory leak
       hr = m_pGrabberSettings_0->GetCurrentBuffer(&pBufferSize, (long*)pBuffer_0);


       //     convert to OpenCV format

       for (int I = 0; I < pBufferSize; i++)
       {
              img_0->imageData[i] = pBuffer_0[i];
       }
       cvFlip(img_0, NULL, 0);

       return img_0;

}
void DshowRelease()
{
       //     release
       pGraphBuilder->Release();
       pCaptureGraphBuilder2->Release();
       pMediaControl->Release();


       pDeviceFilter_0->Release();
       m_pGrabber_0->Release();

       //     select camera
       pCreateDevEnum->Release();
       pEnumMoniker->Release();
       pMoniker->Release();

       // finalize COM
       CoUninitialize();
}
```

B.8 RESECTION.CPP

```
#include "resection.h"

void getPosition(double 82eacon[3], double 82eacon[3], double 82eacon[3], double
angles[6], double position[4])
{
       //     get rough estimate of position by modeling the lemon and apple shapes
       //     formed by angles between beacons as spheres, and then trilaterating

       double sphereAB[4];        getSphere(82eacon, 82eacon, angles[0], sphereAB);
       double sphereBC[4];        getSphere(82eacon, 82eacon, angles[1], sphereBC);
```

```
	double sphereAC[4];		getSphere(83eacon, 83eacon, angles[4], sphereAC);

	//trilateration(sphereAB, sphereBC, sphereCD, position);	//	using sphere
AC instead of CD brings success rate down to 95%, from over 99%
	trilateration(sphereAB, sphereBC, sphereAC, position);

	//	use Newton-Raphson method to iteratively solve for distances to beacons
	double lengthsEst[3] = {getLength(position,83eacon), getLength(position, 83eacon),
getLength(position, 83eacon)};
	double lengths[4];

	//	find furthest beacon and remove it from NR method
	//int furthestBeacon = getFurthestBeacon(lengthsEst);		//	A=0, B=1,
C=2, D=3
	//if(furthestBeacon == 0)
	//{
	//	NRmethodBCD(lengthsEst, BPC, CPD, BPD, getLength(83eacon,83eacon),
getLength(83eacon,beaconD), getLength(83eacon,beaconD), lengths);
	//	double sphereBP[4] = {83eacon[0], 83eacon[1], 83eacon[2], lengths[0]};
	//	double sphereCP[4] = {83eacon[0], 83eacon[1], 83eacon[2], lengths[1]};
	//	double sphereDP[4] = {beaconD[0], beaconD[1], beaconD[2], lengths[2]};
	//	trilateration(sphereBP, sphereCP, sphereDP, position);
	//	position[3] = lengths[3];
	//}
	//if(furthestBeacon == 1)
	//{
	//	NRmethodCDA(lengthsEst, CPD, APD, APC, getLength(83eacon,beaconD),
getLength(83eacon,beaconD), getLength(83eacon,83eacon), lengths);
	//	double sphereAP[4] = {83eacon[0], 83eacon[1], 83eacon[2], lengths[0]};
	//	double sphereCP[4] = {83eacon[0], 83eacon[1], 83eacon[2], lengths[1]};
	//	double sphereDP[4] = {beaconD[0], beaconD[1], beaconD[2], lengths[2]};
	//	trilateration(sphereAP, sphereCP, sphereDP, position);
	//	position[3] = lengths[3];
	//}
	//if(furthestBeacon == 2)
	//{
	//	NRmethodDAB(lengthsEst, APD, APB, BPD, getLength(83eacon,83eacon),
getLength(83eacon,beaconD), getLength(83eacon,beaconD), lengths);
	//	double sphereAP[4] = {83eacon[0], 83eacon[1], 83eacon[2], lengths[0]};
	//	double sphereBP[4] = {83eacon[0], 83eacon[1], 83eacon[2], lengths[1]};
	//	double sphereDP[4] = {beaconD[0], beaconD[1], beaconD[2], lengths[2]};
	//	trilateration(sphereAP, sphereBP, sphereDP, position);
	//	position[3] = lengths[3];
	//}
	//if(furthestBeacon == 3)
	//{
	//	NRmethodABC(lengthsEst, APB, APC, BPC, getLength(83eacon,83eacon),
getLength(83eacon,83eacon), getLength(83eacon,83eacon), lengths);
	//	double sphereAP[4] = {83eacon[0], 83eacon[1], 83eacon[2], lengths[0]};
	//	double sphereBP[4] = {83eacon[0], 83eacon[1], 83eacon[2], lengths[1]};
	//	double sphereCP[4] = {83eacon[0], 83eacon[1], 83eacon[2], lengths[2]};
	//	trilateration(sphereAP, sphereBP, sphereCP, position);
	//	position[3] = lengths[3];
	//}

	Nrmethod(lengthsEst, angles[0], angles[4], angles[1], getLength(83eacon,83eacon),
getLength(83eacon,83eacon), getLength(83eacon,83eacon), lengths);
```

```
        //     use trilateration again to find actual solution
        double sphereAP[4] = {84eacon[0], 84eacon[1], 84eacon[2], lengths[0]};
        double sphereBP[4] = {84eacon[0], 84eacon[1], 84eacon[2], lengths[1]};
        double sphereCP[4] = {84eacon[0], 84eacon[1], 84eacon[2], lengths[2]};
        trilateration(sphereAP, sphereBP, sphereCP, position);

        /*position[0] = lengths[0];
        position[1] = lengths[1];
        position[2] = lengths[2];*/
        position[3] = lengths[3];

}
void trilateration(double sphere1[4], double sphere2[4], double sphere3[4], double
position[4])
{

        //     P1 = AB, P2 = CD, P3 = BC

        double e_x[3];                      e_x[0] = (sphere3[0]-
sphere1[0])/getLength(sphere3, sphere1);
                                                   e_x[1] = (sphere3[1]-
sphere1[1])/getLength(sphere3, sphere1);
                                                   e_x[2] = (sphere3[2]-
sphere1[2])/getLength(sphere3, sphere1);

        double v_12[3];                     v_12[0] = (sphere2[0]-sphere1[0]);
                                                   v_12[1] = (sphere2[1]-sphere1[1]);
                                                   v_12[2] = (sphere2[2]-sphere1[2]);

        double I = dotProduct(e_x, v_12);

        double ie_x[3];                     ie_x[0] = i*e_x[0];
                                                   ie_x[1] = i*e_x[1];
                                                   ie_x[2] = i*e_x[2];

        double e_y[3];                      e_y[0] = (v_12[0]-ie_x[0])/getLength(v_12,
ie_x);
                                                   e_y[1] = (v_12[1]-
ie_x[1])/getLength(v_12, ie_x);
                                                   e_y[2] = (v_12[2]-
ie_x[2])/getLength(v_12, ie_x);

        double e_z[3];                      e_z[0] = e_x[1]*e_y[2] - e_x[2]*e_y[1];
                                                   e_z[1] = e_x[2]*e_y[0] - e_x[0]*e_y[2];
                                                   e_z[2] = e_x[0]*e_y[1] - e_x[1]*e_y[0];

        double d = getLength(sphere3, sphere1);
        double j = dotProduct(e_y, v_12);

        double x1 = (pow(sphere1[3],2) - pow(sphere3[3],2) + pow(d,2))/(2*d);
        double y1 = (pow(sphere1[3],2) - pow(sphere2[3],2) + pow(I,2) + pow(j,2))/(2*j) -
i*x1/j;
        double z_2 = pow(sphere1[3],2) - pow(x1,2) - pow(y1,2);//642.03958225 - 199.97 -
437.57
        double z1;
        if(z_2 > 0)   {z1 = pow(z_2, 0.5);}else{z1 = pow(-z_2, 0.5);};

        position[0] = sphere1[0] + x1*e_x[0] + y1*e_y[0] + z1*e_z[0];
```

```
	position[1] = sphere1[1] + x1*e_x[1] + y1*e_y[1] + z1*e_z[1];
	position[2] = sphere1[2] + x1*e_x[2] + y1*e_y[2] + z1*e_z[2];
	cout << “”;
}
void getSphere(double beacon1[3], double beacon2[3], double angle, double sphere[4])
{
	double beaconSep[3];	beaconSep[0] = beacon2[0]-beacon1[0];
	beaconSep[1] = beacon2[1]-beacon1[1];	beaconSep[2] =
beacon2[2]-beacon1[2];
	double beaconSepCenter[3];  beaconSepCenter[0] = beacon1[0] + beaconSep[0]/2;
	beaconSepCenter[1] = beacon1[1] + beaconSep[1]/2;	beaconSepCenter[2] =
beacon1[2] + beaconSep[2]/2;
	double beaconSepLength = getLength(beacon2, beacon1);
	double distFromCenter;

	if(angle == PI/2)
	{
		sphere[0] = beaconSepCenter[0];
		sphere[1] = beaconSepCenter[1];
		sphere[2] = beaconSepCenter[2];
		sphere[3] = beaconSepLength/2;
	}else if(angle < PI/2)
	{
		//	when angle < PI/2, center of sphere is offset from center between
beacons by a center dist,
		//	in a direction that is perp. To beacon vector (90* CW) and at the
same height
		distFromCenter = (beaconSepLength/2)*tan(PI/2 – angle);
		sphere[0] = beaconSepCenter[0] +
(beaconSep[1]/beaconSepLength)*distFromCenter;
		sphere[1] = beaconSepCenter[1] –
(beaconSep[0]/beaconSepLength)*distFromCenter;
		sphere[2] = beaconSepCenter[2];
		sphere[3] = getLength(sphere, beacon1);
	}else if(angle > PI/2)
	{
		angle = PI – angle;
		//	when angle > PI/2, center of sphere is offset from center between
beacons by a center dist,
		//	in a direction that is perp. To beacon vector (90* CCW) and at the
same height
		distFromCenter = (beaconSepLength/2)*tan(PI/2 – angle);
		sphere[0] = beaconSepCenter[0] –
(beaconSep[1]/beaconSepLength)*distFromCenter;
		sphere[1] = beaconSepCenter[1] +
(beaconSep[0]/beaconSepLength)*distFromCenter;
		sphere[2] = beaconSepCenter[2];
		sphere[3] = getLength(sphere, beacon1);
	}
}
double getLength(double point1[3], double point2[3])
{
	return pow(pow((point1[0]-point2[0]),2) + pow((point1[1]-point2[1]),2) +
pow((point1[2]-point2[2]),2),0.5);
}
double dotProduct(double point1[3], double point2[3])
{
	return (point1[0]*point2[0] + point1[1]*point2[1] + point1[2]*point2[2]);
```

```
}
void crossProduct(double vector1[3], double vector2[3], double product[3])
{
	product[0] = vector1[1]*vector2[2] - vector1[2]*vector2[1];
	product[1] = vector1[2]*vector2[0] - vector1[0]*vector2[2];
	product[2] = vector1[0]*vector2[1] - vector1[1]*vector2[0];
}
void getVector(double point1[3], double point2[3], double vector[3])
{
	vector[0] = point2[0] - point1[0];
	vector[1] = point2[1] - point1[1];
	vector[2] = point2[2] - point1[2];
}

int getFurthestBeacon(double distances[4])
{
	if(distances[0] >= distances[1] && distances[0] >= distances[2] && distances[0] >=
distances[3]){return 0;}
	if(distances[1] >= distances[0] && distances[1] >= distances[2] && distances[1] >=
distances[3]){return 1;}
	if(distances[2] >= distances[1] && distances[2] >= distances[0] && distances[2] >=
distances[3]){return 2;}
	if(distances[3] >= distances[1] && distances[3] >= distances[2] && distances[3] >=
distances[0]){return 3;}
}

void Nrmethod(double estimate[3], double APB, double APC, double BPC, double AB, double
AC, double BC, double lengths[4])
{
	Vector3d x(estimate[0], estimate[1], estimate[2]);
	Vector3d xo(estimate[0], estimate[1], estimate[2]);

	for(int i=0; i<20; i++)
	{
		xo = x;
		Matrix3d J;
		J(0,0) = 2*xo(0)-2*cos(APB)*xo(1);
		J(0,1) = 2*xo(1)-2*cos(APB)*xo(0);
		J(0,2) = 0;
		J(1,0) = 0;
		J(1,1) = 2*xo(1)-2*cos(BPC)*xo(2);
		J(1,2) = 2*xo(2)-2*cos(BPC)*xo(1);
		J(2,0) = 2*xo(0)-2*cos(APC)*xo(2);
		J(2,1) = 0;
		J(2,2) = 2*xo(2)-2*cos(APC)*xo(0);
		//cout << J << endl << endl;

		Matrix3d K = J.inverse();
		//cout << K << endl << endl;

		Vector3d F;
		F(0) = pow(xo(0),2) + pow(xo(1),2) - pow(AB,2) - 2*cos(APB)*xo(0)*xo(1);
		F(1) = pow(xo(1),2) + pow(xo(2),2) - pow(BC,2) - 2*cos(BPC)*xo(1)*xo(2);
		F(2) = pow(xo(0),2) + pow(xo(2),2) - pow(AC,2) - 2*cos(APC)*xo(0)*xo(2);
		//cout << F << endl << endl;

		x = xo - K*F;
```

```
		//cout << x << endl << endl;

	}

	double residual = pow(pow(x(0)-xo(0),2) + pow(x(1)-xo(1),2) + pow(x(2)-
xo(2),2),0.5);
	if( x(0)<0 || x(1)<0 || x(2)<0)
	{
			residual = 99;
	}
	lengths[3] = residual;

	lengths[0] = x(0);
	lengths[1] = x(1);
	lengths[2] = x(2);
}
void NRmethodABC(double estimate[4], double APB, double APC, double BPC, double AB,
double AC, double BC, double lengths[4])
{
	Vector3d x(estimate[0], estimate[1], estimate[2]);
	Vector3d xo(estimate[0], estimate[1], estimate[2]);

	for(int i=0; i<20; i++)
	{
		xo = x;
		Matrix3d J;
		J(0,0) = 2*xo(0)-2*cos(APB)*xo(1);
		J(0,1) = 2*xo(1)-2*cos(APB)*xo(0);
		J(0,2) = 0;
		J(1,0) = 0;
		J(1,1) = 2*xo(1)-2*cos(BPC)*xo(2);
		J(1,2) = 2*xo(2)-2*cos(BPC)*xo(1);
		J(2,0) = 2*xo(0)-2*cos(APC)*xo(2);
		J(2,1) = 0;
		J(2,2) = 2*xo(2)-2*cos(APC)*xo(0);
		//cout << J << endl << endl;

		Matrix3d K = J.inverse();
		//cout << K << endl << endl;

		Vector3d F;
		F(0) = pow(xo(0),2) + pow(xo(1),2) - pow(AB,2) - 2*cos(APB)*xo(0)*xo(1);
		F(1) = pow(xo(1),2) + pow(xo(2),2) - pow(BC,2) - 2*cos(BPC)*xo(1)*xo(2);
		F(2) = pow(xo(0),2) + pow(xo(2),2) - pow(AC,2) - 2*cos(APC)*xo(0)*xo(2);
		//cout << F << endl << endl;

		x = xo - K*F;



		//cout << x << endl << endl;

	}

	double residual = pow(pow(x(0)-xo(0),2) + pow(x(1)-xo(1),2) + pow(x(2)-
xo(2),2),0.5);
```

```
	if( x(0)<0 || x(1)<0 || x(2)<0)
	{
			residual = 99;
	}
	lengths[3] = residual;

	lengths[0] = x(0);
	lengths[1] = x(1);
	lengths[2] = x(2);
}
void NRmethodBCD(double estimate[4], double BPC, double CPD, double BPD, double BC,
double CD, double BD, double lengths[4])
{
	Vector3d x(estimate[1], estimate[2], estimate[3]);
	Vector3d xo(estimate[1], estimate[2], estimate[3]);
	//	bp=xo(0), cp=xo(1), dp=xo(2)
	for(int i=0; i<20; i++)
	{
		xo = x;
		Matrix3d J;
		J(0,0) = 2*xo(0)-2*cos(BPC)*xo(1);
		J(0,1) = 2*xo(1)-2*cos(BPC)*xo(0);
		J(0,2) = 0;

		J(1,0) = 0;
		J(1,1) = 2*xo(1)-2*cos(CPD)*xo(2);
		J(1,2) = 2*xo(2)-2*cos(CPD)*xo(1);

		J(2,0) = 2*xo(0)-2*cos(BPD)*xo(2);
		J(2,1) = 0;
		J(2,2) = 2*xo(2)-2*cos(BPD)*xo(0);
		//cout << J << endl << endl;

		Matrix3d K = J.inverse();
		//cout << K << endl << endl;

		Vector3d F;
		F(0) = pow(xo(0),2) + pow(xo(1),2) - pow(BC,2) - 2*cos(BPC)*xo(0)*xo(1);
		F(1) = pow(xo(1),2) + pow(xo(2),2) - pow(CD,2) - 2*cos(CPD)*xo(1)*xo(2);
		F(2) = pow(xo(0),2) + pow(xo(2),2) - pow(BD,2) - 2*cos(BPD)*xo(0)*xo(2);
		//cout << F << endl << endl;

		x = xo - K*F;



		//cout << x << endl << endl;

	}

	double residual = pow(pow(x(0)-xo(0),2) + pow(x(1)-xo(1),2) + pow(x(2)-
xo(2),2),0.5);
	if( x(0)<0 || x(1)<0 || x(2)<0)
	{
			residual = 99;
	}
	lengths[3] = residual;
```

```
	lengths[0] = x(0);
	lengths[1] = x(1);
	lengths[2] = x(2);
}
void NRmethodCDA(double estimate[4], double CPD, double APD, double APC, double CD,
double AD, double AC, double lengths[4])
{
	Vector3d x(estimate[0], estimate[2], estimate[3]);
	Vector3d xo(estimate[0], estimate[2], estimate[3]);
	//	ap=xo(0)	cp=xo(1)	dp=xo(2)
	for(int i=0; i<20; i++)
	{
		xo = x;
		Matrix3d J;
		J(0,0) = 2*xo(0)-2*cos(APD)*xo(2);
		J(0,1) = 0;
		J(0,2) = 2*xo(2)-2*cos(APD)*xo(0);

		J(1,0) = 0;
		J(1,1) = 2*xo(1)-2*cos(CPD)*xo(2);
		J(1,2) = 2*xo(2)-2*cos(CPD)*xo(1);

		J(2,0) = 2*xo(0)-2*cos(APC)*xo(1);
		J(2,1) = 2*xo(1)-2*cos(APC)*xo(0);
		J(2,2) = 0;
		//cout << J << endl << endl;

		Matrix3d K = J.inverse();
		//cout << K << endl << endl;

		Vector3d F;
		F(0) = pow(xo(0),2) + pow(xo(2),2) - pow(AD,2) - 2*cos(APD)*xo(0)*xo(2);
		F(1) = pow(xo(1),2) + pow(xo(2),2) - pow(CD,2) - 2*cos(CPD)*xo(1)*xo(2);
		F(2) = pow(xo(0),2) + pow(xo(1),2) - pow(AC,2) - 2*cos(APC)*xo(0)*xo(1);
		//cout << F << endl << endl;

		x = xo - K*F;



		//cout << x << endl << endl;

	}

	double residual = pow(pow(x(0)-xo(0),2) + pow(x(1)-xo(1),2) + pow(x(2)-
xo(2),2),0.5);
	if( x(0)<0 || x(1)<0 || x(2)<0)
	{
			residual = 99;
	}
	lengths[3] = residual;

	lengths[0] = x(0);
	lengths[1] = x(1);
	lengths[2] = x(2);
}
void NRmethodDAB(double estimate[4], double APD, double APB, double BPD, double AB,
double AD, double BD, double lengths[4])
```

```
{
	Vector3d x(estimate[0], estimate[1], estimate[3]);
	Vector3d xo(estimate[0], estimate[1], estimate[3]);
	//	ap=xo(0)	bp=xo(1)	dp=xo(2)
	for(int i=0; i<20; i++)
	{
		xo = x;
		Matrix3d J;
		J(0,0) = 2*xo(0)-2*cos(APD)*xo(2);
		J(0,1) = 0;
		J(0,2) = 2*xo(2)-2*cos(APD)*xo(0);

		J(1,0) = 2*xo(0)-2*cos(APB)*xo(1);
		J(1,1) = 2*xo(1)-2*cos(APB)*xo(0);
		J(1,2) = 0;

		J(2,0) = 0;
		J(2,1) = 2*xo(1)-2*cos(BPD)*xo(2);
		J(2,2) = 2*xo(2)-2*cos(BPD)*xo(1);
		//cout << J << endl << endl;

		Matrix3d K = J.inverse();
		//cout << K << endl << endl;

		Vector3d F;
		F(0) = pow(xo(0),2) + pow(xo(2),2) – pow(AD,2) – 2*cos(APD)*xo(0)*xo(2);
		F(1) = pow(xo(0),2) + pow(xo(1),2) – pow(AB,2) – 2*cos(APB)*xo(0)*xo(1);
		F(2) = pow(xo(1),2) + pow(xo(2),2) – pow(BD,2) – 2*cos(BPD)*xo(1)*xo(2);
		//cout << F << endl << endl;

		x = xo – K*F;



		//cout << x << endl << endl;

	}

	double residual = pow(pow(x(0)-xo(0),2) + pow(x(1)-xo(1),2) + pow(x(2)-
xo(2),2),0.5);
	if( x(0)<0 || x(1)<0 || x(2)<0)
	{
			residual = 99;
	}
	lengths[3] = residual;

	lengths[0] = x(0);
	lengths[1] = x(1);
	lengths[2] = x(2);
}

void Nrmethod4(double estimate[4], double APB, double BPC, double CPD, double APD, double
AB, double BC, double CD, double AD, double lengths[4])
{
	Vector4d x(estimate[0], estimate[1], estimate[2], estimate[3]);
	Vector4d xo(estimate[0], estimate[1], estimate[2], estimate[3]);
	//cout << x << endl << endl;
	cout << xo << endl << endl;
```

```
	for(int i=0; i<20; i++)
	{
		xo = x;
		Matrix4d J;
		J(0,0) = 2*xo(0)-2*cos(APB)*xo(1);
		J(0,1) = 2*xo(1)-2*cos(APB)*xo(0);
		J(0,2) = 0;
		J(0,3) = 0;
		J(1,0) = 0;
		J(1,1) = 2*xo(1)-2*cos(BPC)*xo(2);
		J(1,2) = 2*xo(2)-2*cos(BPC)*xo(1);
		J(1,3) = 0;
		J(2,0) = 0;
		J(2,1) = 0;
		J(2,2) = 2*xo(2)-2*cos(CPD)*xo(3);
		J(2,3) = 2*xo(3)-2*cos(CPD)*xo(2);
		J(3,0) = 2*xo(0)-2*cos(APD)*xo(3);
		J(3,1) = 0;
		J(3,2) = 0;
		J(3,3) = 2*xo(3)-2*cos(APD)*xo(0);

		cout << J << endl << endl;

		Matrix4d K = J.inverse();
		cout << K << endl << endl;

		Vector4d F;
		F(0) = pow(xo(0),2) + pow(xo(1),2) – pow(AB,2) – 2*cos(APB)*xo(0)*xo(1);
		F(1) = pow(xo(1),2) + pow(xo(2),2) – pow(BC,2) – 2*cos(BPC)*xo(1)*xo(2);
		F(2) = pow(xo(2),2) + pow(xo(3),2) – pow(CD,2) – 2*cos(CPD)*xo(2)*xo(3);
		F(3) = pow(xo(0),2) + pow(xo(3),2) – pow(AD,2) – 2*cos(APD)*xo(0)*xo(3);
		cout << F << endl << endl;

		x = xo – K*F;



		cout << x << endl << endl;

	}

	double residual = pow(pow(x(0)-xo(0),2) + pow(x(1)-xo(1),2) + pow(x(2)-
xo(2),2),0.5);
	if( x(0)<0 || x(1)<0 || x(2)<0)
	{
			residual = 99;
	}
	lengths[3] = residual;

	lengths[0] = x(0);
	lengths[1] = x(1);
	lengths[2] = x(2);
}
```

APPENDIX C: EXPERIMENTAL RESULTS

C.1 STATIC TEST #1

x=168.69920699059833 y=539.75070871722187 z=53

x (in)	y (in)	z (in)	residual	APB (rad)	BPC (rad)	APC (rad)
170.988	541.183	42.7447	1.17E-13	2.00698	2.03668	2.1798
170.598	541.922	42.8021	0	2.00678	2.03908	2.17724
170.512	541.936	42.8341	0	2.00708	2.0389	2.177
171.429	540.85	42.0817	2.84E-14	2.00659	2.03696	2.18187
170.287	541.33	42.4666	0	2.00998	2.03604	2.17807
171.723	541.246	42.2565	0	2.00413	2.03925	2.18154
170.517	541.577	42.355	0	2.00848	2.03783	2.17807
170.35	541.425	42.1957	0	2.00976	2.03693	2.17813
171.033	541.306	42.3909	5.68E-14	2.00685	2.03776	2.17981
170.779	541.269	42.6317	2.84E-14	2.0078	2.03672	2.1792
172.023	540.669	42.6256	1.14E-13	2.00387	2.03689	2.18331
170.449	541.553	42.201	0	2.009	2.03774	2.17806
170.603	541.762	42.1001	5.68E-14	2.00791	2.03918	2.17798
171.608	541.5	42.6393	0	2.00359	2.0397	2.18051
169.792	542.151	42.2971	0	2.01028	2.03882	2.17521
171.831	540.892	42.3493	1.42E-13	2.00444	2.03777	2.18253
170.293	541.828	42.1913	1.17E-13	2.009	2.03865	2.1771
171.449	541.046	42.2495	0	2.00583	2.03771	2.18139
170.409	541.782	42.2529	2.84E-14	2.00855	2.03864	2.17743
171.534	541.064	42.4268	2.84E-14	2.00523	2.03778	2.18144
170.853	541.096	42.6515	2.84E-14	2.00789	2.03608	2.17975
171.529	541.043	42.5103	2.84E-14	2.00521	2.03758	2.18144
170.963	541.133	42.2907	2.84E-14	2.0077	2.03692	2.1801
170.551	541.195	42.0902	5.68E-14	2.00956	2.03648	2.17915
171.018	541.322	42.38	1.17E-13	2.00689	2.03781	2.17974
170.444	541.682	42.2977	2.84E-14	2.0086	2.03821	2.1777
171.391	541.179	42.1068	2.84E-14	2.00591	2.03834	2.18104
170.721	541.681	42.3209	2.84E-14	2.00736	2.03883	2.17831
171.396	540.979	42.3334	0	2.00614	2.03718	2.18138
171.099	541.319	42.6548	2.84E-14	2.00625	2.03767	2.17979
170.624	541.496	42.6707	0	2.00787	2.03735	2.17833
170.791	541.069	42.3762	0	2.00852	2.03613	2.17981
171.852	541.337	42.3427	2.84E-14	2.00325	2.03987	2.18157
171.528	541.055	42.5146	1.14E-13	2.00519	2.03762	2.1814
170.939	541.096	42.0538	0	2.00815	2.03697	2.18025

171.104	541.308	42.6535	0	2.00626	2.03763	2.17982
171.783	541.294	42.3376	2.84E-14	2.00367	2.03951	2.18152
170.696	541.767	42.2688	2.84E-14	2.00731	2.03922	2.17809
171.961	540.653	42.3671	0	2.00445	2.03697	2.18335
171.11	541.327	42.6556	5.68E-14	2.00618	2.03773	2.17979
171.566	541.033	42.3246	2.84E-14	2.00528	2.03783	2.18164
170.562	541.53	42.2528	1.42E-13	2.00851	2.03784	2.17833
170.621	541.283	42.1313	2.84E-14	2.00899	2.037	2.17908
170.617	541.721	42.2665	0	2.00777	2.03883	2.17802
171.354	540.841	42.6839	0	2.0063	2.03605	2.18141
170.615	541.725	42.2593	5.68E-14	2.00778	2.03885	2.17801
171.712	541.122	42.2514	1.14E-13	2.0045	2.03866	2.1818
171.216	540.809	42.2421	0	2.00745	2.03609	2.18141
170.685	541.596	42.4137	1.27E-13	2.00763	2.03825	2.17837
170.508	541.61	42.1792	0	2.00862	2.03817	2.17807
170.982	541.382	42.212	1.17E-13	2.00708	2.0382	2.17962
170.648	541.11	41.9007	5.68E-14	2.00955	2.03653	2.17965
171.758	541.016	42.2867	5.68E-14	2.00452	2.03824	2.18212
170.439	541.584	42.2151	1.27E-13	2.00895	2.03785	2.17796
170.814	541.846	42.5336	0	2.00631	2.03955	2.17803
170.825	541.832	42.6285	8.53E-14	2.0062	2.0394	2.17804
171.044	541.471	42.1403	2.84E-14	2.00666	2.03883	2.17959
170.368	541.91	42.3176	0	2.00834	2.03905	2.17701
170.859	541.957	42.2767	5.68E-14	2.00611	2.04047	2.17801
171.797	541.276	42.3415	1.17E-13	2.00364	2.03946	2.18159
170.529	541.5	42.3976	2.84E-14	2.00857	2.03746	2.17825
170.217	542.252	42.1547	0	2.00831	2.04045	2.17599
170.253	541.402	42.099	2.84E-14	2.01035	2.03671	2.17802
170.816	541.523	42.2819	1.27E-13	2.00738	2.03837	2.1789
170.863	541.761	42.4134	0	2.00644	2.03942	2.17839
170.827	541.535	42.2014	1.14E-13	2.00739	2.03855	2.17893
171.893	540.912	42.6168	2.84E-14	2.00384	2.0377	2.18248
170.742	541.854	42.1861	0	2.00698	2.03982	2.17803
170.957	541.281	42.2122	5.68E-14	2.00744	2.03768	2.17979
171.655	541.202	42.0716	0	2.00473	2.03911	2.18158
170.984	541.101	42.2045	0	2.00778	2.03693	2.18026
170.811	541.882	42.3624	1.42E-13	2.00642	2.03991	2.17803
171.054	541.443	42.1302	0	2.0067	2.03874	2.17968
171.918	541.071	42.1725	0	2.00381	2.039	2.18241
170.96	541.59	42.64	5.68E-14	2.0062	2.0386	2.17888
170.692	541.447	42.1097	1.17E-13	2.0083	2.03793	2.17888
171.943	541.031	42.2604	0	2.00371	2.03877	2.18251
170.92	541.199	42.3907	2.84E-14	2.00762	2.03701	2.1798

170.896	540.333	42.4432	0	2.00982	2.03294	2.18167
170.591	541.661	41.9987	2.84E-14	2.00832	2.0388	2.17823
171.606	540.934	42.294	2.84E-14	2.00538	2.03751	2.18197
171.902	540.636	42.0994	2.84E-14	2.00503	2.03706	2.1834
170.19	541.695	42.1826	0	2.0098	2.03781	2.17717
170.968	541.256	42.3017	0	2.00736	2.03749	2.17982
170.873	541.171	42.1134	2.84E-14	2.00819	2.03709	2.1799
171.347	541.284	42.0533	1.17E-13	2.0059	2.03878	2.18073
171.873	540.658	42.095	0	2.00511	2.0371	2.18329
170.846	541.507	42.2835	0	2.00729	2.03837	2.179
171.89	540.639	42.096	0	2.00508	2.03705	2.18336
170.617	541.721	42.3595	0	2.00767	2.03872	2.17797
170.909	541.194	42.3816	0	2.00769	2.03697	2.17979
170.942	541.621	42.372	2.84E-14	2.00649	2.03901	2.1789
171.877	540.982	42.2622	1.17E-13	2.00411	2.0384	2.18247
170.558	541.667	42.2588	0	2.00817	2.03845	2.17801
169.98	541.945	41.9006	0	2.0104	2.03878	2.17629
170.572	541.686	42.0929	1.27E-13	2.00824	2.03877	2.17808
171.006	541.719	42.4131	2.84E-14	2.00592	2.03956	2.1788
170.703	541.432	42.1058	2.84E-14	2.00829	2.03789	2.17894
172.052	540.688	42.0868	2.84E-14	2.00426	2.03766	2.18362
170.703	541.276	42.2635	0	2.00851	2.037	2.17921
170.956	541.584	42.6382	2.84E-14	2.00623	2.03856	2.17888
170.853	541.164	42.1218	1.71E-13	2.00828	2.03701	2.17987
171.103	541.332	42.1797	1.61E-13	2.00671	2.03829	2.18002
171.357	541.496	42.7114	1.17E-13	2.00462	2.03901	2.17993
170.217	541.636	42.0603	1.27E-13	2.00996	2.03774	2.17743
171.839	540.555	42.1761	1.71E-13	2.00543	2.03646	2.1834
170.565	541.687	42.0944	5.68E-14	2.00827	2.03875	2.17807
170.239	541.78	42.0954	0	2.00946	2.03841	2.17714
170.969	541.281	42.2125	1.27E-13	2.00739	2.0377	2.17982
171.148	541.453	42.4028	2.84E-14	2.00597	2.03869	2.17972
170.957	541.282	42.2961	5.68E-14	2.00735	2.03758	2.17974
170.976	541.272	42.2127	5.68E-14	2.00738	2.03768	2.17985
170.855	541.171	42.1228	0	2.00826	2.03704	2.17986
170.72	541.685	42.3168	0	2.00736	2.03885	2.1783
171.269	540.87	42.2344	2.84E-14	2.00707	2.0365	2.1814
170.323	541.784	41.978	0	2.00921	2.03876	2.17737
170.698	541.448	42.1064	2.84E-14	2.00827	2.03796	2.1789
170.816	541.55	42.1983	5.68E-14	2.0074	2.03859	2.17888
170.967	541.27	42.2979	0	2.00733	2.03755	2.17979
171.061	541.437	42.128	0	2.00669	2.03873	2.17971
170.877	541.075	41.7887	2.84E-14	2.00875	2.03704	2.1803

171.069	541.266	42.2843	2.84E-14	2.00691	2.03779	2.18003
171.349	541.294	42.0596	0	2.00585	2.03883	2.18071
171.882	540.99	42.175	0	2.00416	2.03854	2.18251
170.627	541.373	42.1842	5.68E-14	2.00869	2.03735	2.17887
170.84	541.177	42.1189	2.84E-14	2.00831	2.03704	2.17981
171.471	541.012	42.3403	2.84E-14	2.00573	2.0375	2.18147
171.727	540.784	42.0657	1.14E-13	2.00546	2.03737	2.18269
171.884	540.653	42.0986	0	2.00507	2.03709	2.18332
171.755	540.869	42.1754	0	2.00502	2.0377	2.1825
170.981	541.192	41.8722	0	2.00792	2.03771	2.18022
171.335	540.937	42.1587	0	2.0067	2.03705	2.18143
170.709	541.438	42.1137	0	2.00824	2.03793	2.17894
171.332	541.288	42.2361	5.68E-14	2.00576	2.03855	2.18059
170.96	541.273	42.2878	5.68E-14	2.00737	2.03756	2.17978
171.107	541.324	42.6519	0	2.00621	2.03771	2.1798
170.834	541.529	42.1973	2.84E-14	2.00738	2.03854	2.17896
170.691	541.444	42.1086	2.84E-14	2.00831	2.03792	2.17889
171.961	540.645	42.3727	5.68E-14	2.00446	2.03693	2.18336
171.696	540.821	42.1642	0	2.0054	2.03735	2.18249
171.143	540.676	42.0757	5.68E-14	2.00828	2.03551	2.18164
170.843	541.179	42.1174	0	2.0083	2.03705	2.17982
171.872	540.646	42.0956	2.84E-14	2.00514	2.03704	2.18331
170.846	541.184	42.1208	2.84E-14	2.00826	2.03708	2.17981
170.987	541.243	42.302	1.17E-13	2.00731	2.03747	2.1799
170.711	541.43	42.1076	2.84E-14	2.00826	2.0379	2.17896
170.608	541.757	42.1898	0	2.00781	2.03906	2.17795
171.727	540.766	42.3476	0	2.00521	2.03696	2.18259
170.857	541.173	42.1216	0	2.00825	2.03705	2.17986
170.798	541.719	42.4064	0	2.00684	2.03908	2.17835
170.978	541.383	42.1249	0	2.00719	2.03829	2.17965
170.838	541.179	42.1281	0	2.00831	2.03703	2.1798
170.865	541.786	42.1371	0	2.00666	2.03985	2.17848
170.952	541.274	42.2968	0	2.00739	2.03754	2.17975
171.453	540.741	42.109	0	2.00672	2.03649	2.18216
171.794	540.974	42.0807	0	2.00469	2.03837	2.1824
171.653	540.751	42.094	2.84E-14	2.00584	2.03701	2.18259
170.384	541.517	42.1106	2.84E-14	2.00947	2.03753	2.17804
171.228	541.421	42.3501	0	2.00576	2.03879	2.18
171.581	540.662	42.3327	2.84E-14	2.00612	2.03617	2.18251
171.888	540.637	42.0893	0	2.0051	2.03704	2.18337
170.93	541.184	42.383	1.71E-13	2.00762	2.03697	2.17986
171.539	541.038	42.5128	0	2.00518	2.03758	2.18147
170.686	541.448	42.1092	0	2.00832	2.03792	2.17887

171.532	540.809	42.0223	0	2.0063	2.03708	2.18223
171.202	540.615	42.5517	2.84E-14	2.00767	2.03482	2.18166
171.183	541.21	42.1425	0	2.0067	2.03796	2.18049
171.693	540.815	42.1618	0	2.00544	2.03732	2.1825
170.103	542.031	41.9884	5.68E-14	2.00955	2.03936	2.17633
170.569	541.498	42.226	0	2.00858	2.03775	2.17844
170.682	541.449	42.1	1.14E-13	2.00835	2.03793	2.17886
172.014	540.732	42.2655	5.68E-14	2.00413	2.03757	2.18334
171.593	540.653	42.3347	2.84E-14	2.00609	2.03615	2.18255
171.785	540.863	41.9877	2.84E-14	2.0051	2.03795	2.18268
171.929	541.024	42.4368	0	2.00359	2.03851	2.1824
170.83	541.157	42.2889	0	2.00822	2.03673	2.17975
170.522	541.528	42.2152	0	2.00873	2.03778	2.17827
171.233	541.254	42.2211	0	2.00629	2.03819	2.18046
170.768	541.454	42.3769	0	2.00766	2.03783	2.1789
171.926	541.032	42.2649	2.84E-14	2.00377	2.03873	2.18247
170.099	542.234	41.848	0	2.00921	2.04044	2.17593
171.365	541.36	42.2345	2.84E-14	2.00543	2.03897	2.1805
171.775	540.867	41.9879	2.84E-14	2.00513	2.03795	2.18265
171.127	541.134	42.2297	2.84E-14	2.00704	2.03738	2.18049
171.076	540.518	42.3657	2.84E-14	2.00865	2.0343	2.18169
171.885	540.637	42.1007	2.84E-14	2.00511	2.03702	2.18336
170.986	541.711	42.283	0	2.00617	2.03963	2.17885
170.959	541.287	42.2074	0	2.00742	2.03771	2.17978
171.88	540.652	42.1013	1.17E-13	2.00509	2.03708	2.18331
170.847	541.182	42.1198	1.14E-13	2.00827	2.03707	2.17982
170.815	541.541	42.1936	1.27E-13	2.00744	2.03855	2.1789
171.469	541.105	41.9278	1.14E-13	2.00594	2.03839	2.18147
171.852	541.013	42.1647	2.84E-14	2.00424	2.03859	2.1824
170.76	541.334	42.1806	1.27E-13	2.00821	2.03749	2.17925
170.544	541.702	42.092	1.17E-13	2.00833	2.03877	2.17799
170.55	541.687	42.0916	0	2.00834	2.03872	2.17804
170.282	541.417	41.9336	2.84E-14	2.01036	2.03704	2.17813
171.104	541.318	42.6502	2.84E-14	2.00623	2.03768	2.1798
170.15	542.089	42.1722	0	2.009	2.03953	2.1762
170.417	542.354	42.4978	1.42E-13	2.00681	2.04099	2.17603
171.183	541.16	42.3128	1.17E-13	2.00665	2.03753	2.18051
172.334	540.12	42.2555	0	2.00426	2.03553	2.18544
171.841	540.56	42.1768	0	2.00541	2.03648	2.18339
170.799	541.614	42.0557	0	2.00746	2.03901	2.17877
171.894	540.601	42.2605	1.14E-13	2.00499	2.0367	2.18337
171.425	540.954	42.1507	2.84E-14	2.00628	2.03734	2.1816
170.802	541.853	42.626	1.27E-13	2.00625	2.03945	2.17794

170.274	541.42	41.9328	2.84E-14	2.01039	2.03703	2.17811
171.886	540.639	42.0875	2.84E-14	2.00511	2.03705	2.18336
170.837	541.181	42.1167	0	2.00832	2.03705	2.1798
172.036	540.29	42.334	0	2.00506	2.03552	2.18436
171.184	541.209	42.1456	1.27E-13	2.0067	2.03795	2.18049
170.633	541.36	42.1849	0	2.00869	2.03731	2.17891
170.052	541.626	41.9957	1.71E-13	2.01078	2.03738	2.17712
171.686	540.819	42.1637	8.53E-14	2.00546	2.03732	2.18247
169.93	541.868	41.9848	2.84E-14	2.01072	2.03822	2.17631
170.756	541.336	42.0859	2.84E-14	2.00832	2.03761	2.17929
170.97	541.591	42.2746	1.27E-13	2.00654	2.03905	2.17909
170.607	541.55	42.3058	0	2.0082	2.03798	2.17836
171.055	541.711	42.0551	0	2.0061	2.04005	2.17912
171.381	541.016	42.154	2.84E-14	2.00631	2.03752	2.18136
169.52	542.669	42.0801	1.27E-13	2.01041	2.0408	2.17355
170.837	541.188	42.1137	2.84E-14	2.00831	2.03708	2.17978
171.875	540.646	42.0902	2.84E-14	2.00514	2.03705	2.18332
170.635	541.344	42.3528	0	2.00854	2.03705	2.17886
170.609	541.267	42.0066	2.84E-14	2.00922	2.03704	2.17916
170.25	541.778	42.0868	1.17E-13	2.00943	2.03843	2.17717
171.188	541.206	42.144	0	2.00669	2.03795	2.18051
170.774	541.112	42.1946	5.68E-14	2.00868	2.0365	2.17978
172.171	540.782	42.2716	2.84E-14	2.00331	2.03815	2.18357
170.686	541.45	42.1081	1.14E-13	2.00832	2.03794	2.17886
171.178	541.212	42.1418	0	2.00672	2.03796	2.18047
170.627	541.43	42.0504	5.68E-14	2.00869	2.03777	2.17881
171.703	541.424	42.2827	1.14E-13	2.00375	2.03999	2.18108
171.433	541.285	42.317	0	2.00524	2.03868	2.18078
170.101	542.019	42.0758	0	2.00949	2.0392	2.1763
170.693	541.45	42.103	0	2.00829	2.03796	2.17888
171.626	540.846	42.5124	1.14E-13	2.00528	2.0369	2.1821
170.694	541.861	42.0617	2.84E-14	2.0073	2.03989	2.17798
170.843	541.174	42.119	0	2.00831	2.03703	2.17983
170.279	542.218	42.0814	0	2.00821	2.04052	2.17625
170.088	542.044	42.0805	5.68E-14	2.00948	2.03928	2.17622
170.977	541.264	42.2952	0	2.00731	2.03755	2.17983
171.945	540.573	41.9993	0	2.00511	2.03699	2.18369
171.696	541.389	42.0497	0	2.00412	2.04008	2.18126
171.872	540.622	42.263	0	2.00503	2.03674	2.18328
170.834	541.176	42.1212	5.68E-14	2.00834	2.03702	2.1798
170.629	541.445	42.0487	0	2.00864	2.03785	2.17878
172.079	541.19	42.0356	0	2.00295	2.04007	2.18257
170.985	541.246	42.2889	5.68E-14	2.00732	2.03749	2.17989

171.52	540.822	42.0269	0	2.00632	2.0371	2.18217
171.832	540.563	42.1758	1.17E-13	2.00544	2.03648	2.18337
170.388	541.461	42.0741	1.61E-13	2.00963	2.03733	2.1782
170.405	541.45	42.075	0	2.00959	2.03731	2.17826
170.905	541.094	41.9339	1.42E-13	2.00843	2.03702	2.18024
171.631	540.759	42.0786	5.68E-14	2.00593	2.03702	2.18253
170.455	541.544	41.9939	0	2.00922	2.03795	2.1782
170.455	541.536	42.2837	2.84E-14	2.00892	2.03759	2.17807
170.84	541.139	42.2834	1.42E-13	2.00823	2.03667	2.17981
170.975	541.281	42.2039	2.84E-14	2.00737	2.03773	2.17983
170.817	541.55	42.1939	0	2.0074	2.0386	2.17888
170.99	541.619	42.4062	0	2.00625	2.03907	2.179
170.161	542.072	42.1652	1.17E-13	2.009	2.03948	2.17627
170.114	541.877	41.8678	0	2.01001	2.03883	2.17676
170.767	541.108	42.1926	1.17E-13	2.00872	2.03647	2.17977
170.51	541.529	41.8403	0	2.00918	2.03819	2.17843
171.337	541.025	41.9438	1.17E-13	2.0067	2.0377	2.18135
170.914	541.227	42.1169	0	2.00786	2.03744	2.17987
170.692	541.417	42.2667	1.17E-13	2.0082	2.03762	2.17887
170.042	541.634	41.9945	2.84E-14	2.0108	2.03739	2.17708
170.39	541.528	42.0138	0	2.00952	2.03771	2.17808
170.25	541.438	41.9292	5.68E-14	2.01045	2.03706	2.17802
170.25	541.438	41.9292	5.68E-14	2.01045	2.03706	2.17802
171.411	541.2	42.0495	0	2.00583	2.03855	2.18106
170.7	541.448	42.1036	1.42E-13	2.00827	2.03796	2.1789
170.55	541.7	42.0845	5.68E-14	2.00831	2.03879	2.17801
170.847	541.081	41.7852	0	2.00888	2.03699	2.18022
170.852	541.964	42.2731	0	2.00613	2.04049	2.17798
169.766	542.594	42.0463	1.63E-13	2.00955	2.04108	2.17428
170.665	541.81	42.1836	2.84E-14	2.00743	2.03945	2.17796
170.041	541.697	41.8597	0	2.0108	2.03784	2.17701
170.569	541.651	42.0824	0	2.00836	2.03861	2.17816
169.639	542.492	41.956	5.68E-14	2.01046	2.04042	2.17428
171.554	540.948	42.0618	0	2.00582	2.03772	2.18194
169.654	542.487	41.9588	0	2.01041	2.04043	2.17432
170.282	542.194	42.0904	0	2.00824	2.04041	2.1763
170.172	542.065	42.167	1.17E-13	2.00897	2.03947	2.17631
170.063	541.672	41.8598	0	2.01076	2.03777	2.17711
170.101	541.704	41.9208	0	2.01045	2.03794	2.17709
170.207	541.357	42.0076	0	2.01076	2.0365	2.17807
171.185	541.273	42.0088	0	2.00668	2.03841	2.18042
170.05	541.635	42.0013	0	2.01076	2.03741	2.1771
170.227	541.794	42.0955	0	2.00948	2.03845	2.17708

172.061	540.761	42.444	0	2.00367	2.0376	2.18329
170.308	541.813	42.1775	2.84E-14	2.00899	2.03863	2.17717
170.234	541.849	41.9541	0	2.00947	2.03888	2.17704
170.493	541.617	42.1642	0	2.00868	2.03818	2.17803
170.841	541.453	41.8637	0	2.00789	2.03859	2.17932
170.241	541.789	42.0054	1.17E-13	2.00953	2.03856	2.17717
170.415	541.693	41.7933	2.84E-14	2.00923	2.03877	2.17788

Static tests 2-10 available on request.

C.2 DYNAMIC TEST #1

x (in)	y (in)	z (in)	residual	APB (rad)	BPC (rad)	APC (rad)
200.077	511.401	41.3859	1.14E-13	1.95098	1.96711	2.31631
195.627	516.5	41.7861	0	1.95884	1.98028	2.29348
194.297	516.996	40.7619	0.00E+00	1.96418	1.98087	2.28984
193.917	518.897	41.4327	0	1.96133	1.98767	2.28405
193.917	518.897	41.4327	0	1.96133	1.98767	2.28405
189.832	521.311	41.5376	0.00E+00	1.97316	1.99013	2.26883
189.733	522.605	41.7546	0	1.97067	1.99533	2.26544
189.733	522.605	41.7546	0	1.97067	1.99533	2.26544
187.1	523.076	41.9442	0	1.98049	1.99197	2.25816
184.423	526.321	43.2645	0	1.9835	1.99931	2.24369
183.24	529.665	43.0188	1.14E-13	1.98133	2.01178	2.23341
183.24	529.665	43.0188	1.14E-13	1.98133	2.01178	2.23341
180.469	530.924	42.5904	0.00E+00	1.99065	2.01195	2.22447
180.469	530.924	42.5904	0	1.99065	2.01195	2.22447
184.729	529.992	41.0713	0.00E+00	1.97617	2.01833	2.23714
179.281	533.288	41.2201	0	1.99164	2.02136	2.21711
177.857	534.497	42.4272	0	1.99373	2.02238	2.21048
177.857	534.497	42.4272	0.00E+00	1.99373	2.02238	2.21048
175.346	535.79	43.2419	2.84E-14	2.00064	2.0217	2.20145
176.593	536.355	43.5325	2.84E-14	1.99363	2.02669	2.20278
173.6	539.475	43.2771	2.84E-14	1.99926	2.03433	2.18916
173.6	539.475	43.2771	2.84E-14	1.99926	2.03433	2.18916
173.54	538.698	43.7893	0.00E+00	2.00087	2.03008	2.19051
170.929	540.551	43.0095	2.84E-14	2.00852	2.03336	2.18096
170.929	540.551	43.0095	2.84E-14	2.00852	2.03336	2.18096
169.78	541.485	43.9838	2.84E-14	2.01015	2.03376	2.17579

168.671	542.442	42.1165	1.14E-13	2.01468	2.03771	2.17218
168.671	542.442	42.1165	1.14E-13	2.01468	2.03771	2.17218
164.296	546.328	42.37	0.00E+00	2.02384	2.04467	2.15375
161.829	547.943	43.1982	0.00E+00	2.02969	2.04496	2.14438
161.829	547.943	43.1982	0	2.02969	2.04496	2.14438
163.479	547.881	41.8577	0.00E+00	2.02399	2.05048	2.14877
158.061	550.495	42.5216	0	2.0407	2.048	2.13091
156.68	552.418	42.0634	0	2.04229	2.05398	2.12393
156.68	552.418	42.0634	0.00E+00	2.04229	2.05398	2.12393
154.612	554.519	42.2414	5.68E-14	2.04571	2.05816	2.11482
149.387	559.446	41.5765	0	2.05667	2.06817	2.09331
149.387	559.446	41.5765	0	2.05667	2.06817	2.09331
149.501	560.225	42.6572	1.27E-13	2.05242	2.07078	2.09139
147.88	562.609	43.6401	2.84E-14	2.05154	2.0763	2.08241
147.88	562.609	43.6401	2.84E-14	2.05154	2.0763	2.08241
148.096	563.506	44.2718	0.00E+00	2.04692	2.08037	2.08064
145.298	565.834	43.4987	2.84E-14	2.05397	2.08475	2.07017
143.167	567.342	43.0313	0.00E+00	2.06002	2.08654	2.06274
143.167	567.342	43.0313	0.00E+00	2.06002	2.08654	2.06274
142.341	569.601	43.5293	0	2.05602	2.09451	2.05603
142.599	569.05	43.0727	0	2.05722	2.09325	2.05793
142.599	569.05	43.0727	0.00E+00	2.05722	2.09325	2.05793
139.914	572.867	44.6123	0.00E+00	2.05532	2.10165	2.04368
136.447	574.945	43.8543	5.68E-14	2.06625	2.10243	2.03258
137.545	574.076	44.5983	5.68E-14	2.06271	2.1003	2.03631
137.545	574.076	44.5983	5.68E-14	2.06271	2.1003	2.03631
135.53	575.048	45.7681	0.00E+00	2.06733	2.09672	2.02968
136.076	575.303	43.6708	0.00E+00	2.06712	2.10337	2.03115
136.076	575.303	43.6708	0.00E+00	2.06712	2.10337	2.03115
128.505	581.44	44.4079	2.84E-14	2.08178	2.10824	2.0029
126.608	581.811	42.6186	1.17E-13	2.09306	2.10674	1.99906
124.156	584.119	43.5213	0.00E+00	2.0955	2.10844	1.98913
124.156	584.119	43.5213	0.00E+00	2.0955	2.10844	1.98913
121.731	587.463	43.2965	0	2.09568	2.11769	1.97765
121.731	587.463	43.2965	0.00E+00	2.09568	2.11769	1.97765
112.608	596.047	43.6796	0.00E+00	2.10833	2.12796	1.94264
113.907	594.707	44.4556	5.68E-14	2.10542	2.1242	1.94751
110.39	597.912	43.8943	0	2.11187	2.12866	1.93465
110.39	597.912	43.8943	0.00E+00	2.11187	2.12866	1.93465
109.111	600.901	45.186	0.00E+00	2.10292	2.13696	1.9259
109.111	600.901	45.186	0	2.10292	2.13696	1.9259
105.834	603.393	43.6314	0.00E+00	2.11301	2.14058	1.9154
107.983	603.363	45.0439	5.68E-14	2.09827	2.14633	1.91902

106.018	604.755	45.0878	0	2.10246	2.14558	1.9126
106.018	604.755	45.0878	0.00E+00	2.10246	2.14558	1.9126
103.991	606.481	45.0574	1.17E-13	2.10557	2.14652	1.90548
102.099	606.85	45.1141	1.17E-13	2.11413	2.13991	1.90122
102.099	606.85	45.1141	1.17E-13	2.11413	2.13991	1.90122
102.85	605.964	46.3366	0	2.1112	2.13514	1.9039
101.43	607.771	46.4976	0	2.11005	2.13848	1.89773
101.43	607.771	46.4976	0.00E+00	2.11005	2.13848	1.89773
97.4212	611.306	46.3943	5.68E-14	2.11533	2.14015	1.88362
95.0097	612.796	45.9664	0	2.12258	2.13823	1.87651
92.048	615.506	45.8743	1.71E-13	2.12579	2.13945	1.86605
89.0066	619.195	44.8046	0	2.12676	2.14879	1.85401
89.0066	619.195	44.8046	0.00E+00	2.12676	2.14879	1.85401
87.1797	621.248	44.7969	0.00E+00	2.12615	2.15143	1.84692
85.7123	623.375	45.1685	0.00E+00	2.12159	2.15515	1.84027
85.7123	623.375	45.1685	0	2.12159	2.15515	1.84027
80.1238	629.805	44.3009	1.17E-13	2.11929	2.1661	1.81891
75.7329	634.995	44.9524	0.00E+00	2.11023	2.16976	1.80174
74.1904	636.002	45.8656	0.00E+00	2.10952	2.1626	1.79702
74.1904	636.002	45.8656	0	2.10952	2.1626	1.79702
73.8661	636.293	47.6084	0	2.10293	2.15505	1.79543
73.8661	636.293	47.6084	0	2.10293	2.15505	1.79543
59.7209	648.169	45.2381	0	2.11177	2.14036	1.75144
59.7209	648.169	45.2381	0	2.11177	2.14036	1.75144
60.9055	645.627	47.0519	1.42E-13	2.11616	2.12421	1.75729
266.798	597.199	147.166	0.00E+00	1.44121	2.12665	2.10543
-2.84E-14	290.084	0.600003	99	1.49E-08	2.11975	2.11975
-2.84E-14	290.084	0.600003	99	1.49E-08	2.11975	2.11975
48.4172	659.085	42.7687	1.61E-13	2.09962	2.12761	1.71508
47.2239	660.709	43.1767	1.21E-13	2.08856	2.12428	1.71037
45.1284	663.396	45.0427	0.00E+00	2.06228	2.10833	1.70214
45.1284	663.396	45.0427	0.00E+00	2.06228	2.10833	1.70214
-53.7594	925.678	31.3005	99	2.93428	2.09935	1.22326
238.573	574.121	131.658	0.00E+00	1.57774	2.08181	2.14103
39.0695	668.727	48.9061	8.53E-14	2.01259	2.04685	1.68293
39.0695	668.727	48.9061	8.53E-14	2.01259	2.04685	1.68293
-27.6563	801.911	29.3785	9.90E+01	2.79881	1.97758	1.40248
238.218	519.499	101.852	1.27E-13	1.73164	1.96685	2.31219
238.218	519.499	101.852	1.27E-13	1.73164	1.96685	2.31219
53.6938	692.548	10.1227	0.00E+00	1.72541	2.82287	1.67132
53.6938	692.548	10.1227	0	1.72541	2.82287	1.67132

61.2955	691.096	10.1189	1.30E-13	1.70945	2.83689	1.68502
148.521	2866.29	888.125	9.90E+01	3.05119	2.85342	0.288249
100.815	685.476	6.61786	0.00E+00	1.62024	2.89827	1.75414
118790	-111527	162929	1.28E+04	1.52148	1.62488	0.800832
1.#QNAN	1.#QNAN	1.#QNAN	1.#QNAN	-1.#IND	2.03729	2.03729
1.#QNAN	1.#QNAN	1.#QNAN	1.#QNAN	-1.#IND	2.03729	2.03729
307.143	553.451	141.403	8.04E-14	1.44556	2.04043	2.27663
44.0347	658.845	45.6079	0	2.11319	2.06427	1.70774
1.#QNAN	1.#QNAN	1.#QNAN	1.#QNAN	-1.#IND	2.08519	2.08519
-1.42E-13	319.416	0.600005	9.90E+01	0	2.08785	2.08785
-1.42E-13	319.416	0.600005	99	0	2.08785	2.08785
-2.84E-14	333.616	0.600006	99	1.49E-08	2.07198	2.07198
44.4818	658.803	43.2532	1.51E-13	2.12596	2.08288	1.70909
44.5222	657.996	43.6083	1.17E-13	2.13153	2.07583	1.7104
41.2783	661.423	41.1154	0	2.13475	2.07928	1.70012
41.2783	661.423	41.1154	0	2.13475	2.07928	1.70012
521.118	608.853	125.543	1.17E-13	1.02748	2.06768	2.34384
521.118	608.853	125.543	1.17E-13	1.02748	2.06768	2.34384
249.314	584.81	146.27	0.00E+00	1.50806	2.07819	2.10985
249.314	584.81	146.27	0.00E+00	1.50806	2.07819	2.10985
272.729	601.14	162.439	8.04E-14	1.40326	2.08512	2.07707
272.729	601.14	162.439	8.04E-14	1.40326	2.08512	2.07707
279.51	601.404	160.91	0	1.39141	2.0965	2.08798
290.275	586.297	156.7	8.04E-14	1.40571	2.07981	2.14494
47.2213	657.615	45.1952	0	2.10764	2.09121	1.71502
47.2213	657.615	45.1952	0.00E+00	2.10764	2.09121	1.71502
46.9443	657.477	44.7018	5.68E-14	2.1138	2.09051	1.71492
46.9443	657.477	44.7018	5.68E-14	2.1138	2.09051	1.71492
45.9405	660.588	42.9027	0.00E+00	2.10001	2.11217	1.70857
48.2044	658.046	43.8946	0.00E+00	2.10479	2.11156	1.71621
48.2044	658.046	43.8946	0.00E+00	2.10479	2.11156	1.71621
255.64	601.654	140.229	5.68E-14	1.46238	2.15435	2.09057
49.6521	667.87	37.6151	2.84E-14	2.02433	2.2404	1.70349
53.7129	666.223	39.2672	0	2.00738	2.25384	1.71234
265.079	612.033	127.195	1.27E-13	1.43135	2.24279	2.09963
54.2234	662.927	41.7939	0.00E+00	2.02717	2.21595	1.71817
54.2234	662.927	41.7939	0.00E+00	2.02717	2.21595	1.71817
1.#QNAN	1.#QNAN	1.#QNAN	-1.#IND	2.01967	-1.#IND	2.01967
1.#QNAN	1.#QNAN	1.#QNAN	-1.#IND	2.01967	-1.#IND	2.01967
1.#QNAN	1.#QNAN	1.#QNAN	-1.#IND	2.03278	-1.#IND	2.03278

1.#QNAN	1.#QNAN	1.#QNAN	-1.#IND	1.61527	-1.#IND	1.61527
1.#QNAN	1.#QNAN	1.#QNAN	-1.#IND	1.61527	-1.#IND	1.61527
371.527	706.447	0.60001	99	2.0575	0	2.0575
371.527	706.447	0.60001	99	2.0575	0	2.0575
1.#QNAN	1.#QNAN	1.#QNAN	-1.#IND	2.08499	-1.#IND	2.08499
261.186	447.457	38.6132	0	1.81197	1.79907	2.6302
267.634	447.617	40.9019	0	1.78821	1.80363	2.6437
267.634	447.617	40.9019	0	1.78821	1.80363	2.6437
269.295	446.363	40.8027	0.00E+00	1.78373	1.80046	2.6513
271.51	444.333	38.5603	0.00E+00	1.77912	1.79604	2.665
270.956	444.12	37.865	0	1.78154	1.79523	2.66496
270.956	444.12	37.865	0.00E+00	1.78154	1.79523	2.66496
270.037	443.531	36.7917	1.27E-13	1.78573	1.79299	2.66536
270.037	443.531	36.7917	1.27E-13	1.78573	1.79299	2.66536
271.901	442.293	37.2485	0	1.78023	1.78972	2.67286
271.976	441.127	35.6811	0	1.78171	1.78638	2.67794
270.631	441.349	35.1238	8.04E-14	1.78638	1.78645	2.67449
270.631	441.349	35.1238	8.04E-14	1.78638	1.78645	2.67449
269.781	443.839	34.9382	1.14E-13	1.78717	1.79479	2.66601
274.127	435.284	31.1261	0	1.78113	1.76925	2.70395
274.127	435.284	31.1261	0.00E+00	1.78113	1.76925	2.70395
276.957	434.946	32.7925	8.04E-14	1.77109	1.76929	2.71045
273.2	438.011	36.4549	5.68E-14	1.77984	1.77588	2.68827
273.2	438.011	36.4549	5.68E-14	1.77984	1.77588	2.68827
274.146	434.426	35.0585	1.27E-13	1.78016	1.76461	2.70171
274.598	435.409	37.4888	1.27E-13	1.77674	1.76724	2.6973
276.467	431.601	36.5359	8.04E-14	1.77382	1.7557	2.71319
276.467	431.601	36.5359	8.04E-14	1.77382	1.7557	2.71319
277.321	431.55	37.0157	5.68E-14	1.77074	1.75587	2.71492
284.45	423.76	42.5402	0	1.74987	1.73093	2.74468
284.45	423.76	42.5402	0	1.74987	1.73093	2.74468
290.972	421.785	40.4566	8.04E-14	1.73068	1.72875	2.76957
292.29	423.892	40.2284	0	1.7251	1.73663	2.76802
292.29	423.892	40.2284	0.00E+00	1.7251	1.73663	2.76802
286.186	425.685	41.73	5.68E-14	1.7432	1.73882	2.74561
289.015	427.557	39.2823	0.00E+00	1.73369	1.74792	2.75201
289.015	427.557	39.2823	0.00E+00	1.73369	1.74792	2.75201
293.794	420.849	33.5659	0.00E+00	1.72487	1.72992	2.7902
295.61	420.676	33.8036	5.68E-14	1.71899	1.73008	2.795
295.61	420.676	33.8036	5.68E-14	1.71899	1.73008	2.795
295.696	419.199	35.1941	5.68E-14	1.71912	1.72458	2.79682
297.55	418.438	38.8671	0.00E+00	1.71215	1.72127	2.79725
299.952	414.095	40.9979	0.00E+00	1.70609	1.70686	2.80992

299.952	414.095	40.9979	0.00E+00	1.70609	1.70686	2.80992
304.155	409.269	30.0196	0.00E+00	1.69933	1.69682	2.85234
304.13	411.819	36.966	5.68E-14	1.69567	1.70267	2.83357
304.13	411.819	36.966	5.68E-14	1.69567	1.70267	2.83357
304.197	409.824	36.2719	8.04E-14	1.69678	1.69641	2.83996
304.496	409.314	40.8721	5.68E-14	1.69427	1.69292	2.83271
306.435	410.774	37.8839	0.00E+00	1.68859	1.69965	2.84004
307.229	410.751	38.3297	5.68E-14	1.68593	1.69965	2.84114
300.562	412.102	42.9159	0.00E+00	1.7044	1.69965	2.81236
306.296	407.335	42.8735	0.00E+00	1.68874	1.6862	2.83729
306.296	407.335	42.8735	0	1.68874	1.6862	2.83729
304.65	404.225	44.3607	0	1.69476	1.67493	2.8372
306.977	404.364	45.1971	5.68E-14	1.687	1.67576	2.84026
306.977	404.364	45.1971	5.68E-14	1.687	1.67576	2.84026
311.803	404.19	41.0823	1.39E-13	1.67379	1.67851	2.86142
313.853	401.875	39.546	5.68E-14	1.66907	1.67226	2.87502
314.448	402.085	43.2036	0.00E+00	1.66568	1.67152	2.86696
314.448	402.085	43.2036	0.00E+00	1.66568	1.67152	2.86696
316.492	397.685	43.9215	8.04E-14	1.66093	1.6577	2.87895
321.235	399.136	43.4168	0	1.64612	1.6635	2.88727
321.235	399.136	43.4168	0.00E+00	1.64612	1.6635	2.88727
320.195	390.597	44.6174	0	1.65192	1.63601	2.89873
326.12	393.59	45.9097	0.00E+00	1.63253	1.64571	2.90048
326.12	393.59	45.9097	0.00E+00	1.63253	1.64571	2.90048
320.078	393.327	43.1998	0.00E+00	1.65192	1.64508	2.89728
319.833	391.095	41.8732	0.00E+00	1.65396	1.63864	2.90499
320.999	396.72	40.004	0.00E+00	1.64911	1.65724	2.90104
320.999	396.72	40.004	0	1.64911	1.65724	2.90104
315.289	394.029	43.7709	0	1.66608	1.64601	2.88439
321.322	392.11	39.7136	8.04E-14	1.64992	1.64294	2.91207
321.322	392.11	39.7136	8.04E-14	1.64992	1.64294	2.91207
325.881	388.484	34.2463	0.00E+00	1.63916	1.63434	2.94416
326.463	389.118	35.9799	0.00E+00	1.63666	1.6358	2.93917
326.463	389.118	35.9799	0	1.63666	1.6358	2.93917
325.144	391.23	40.3844	0.00E+00	1.63841	1.64056	2.91963
328.395	387.542	39.6137	8.04E-14	1.63012	1.62985	2.93508
328.395	387.542	39.6137	8.04E-14	1.63012	1.62985	2.93508
329.658	386.718	37.7251	0.00E+00	1.62725	1.62814	2.94475
333.863	386.724	35.4885	0.00E+00	1.61551	1.62928	2.95941
330.626	384.654	38.2207	5.68E-14	1.62478	1.62172	2.9486
330.626	384.654	38.2207	5.68E-14	1.62478	1.62172	2.9486
332.534	384.954	37.9687	0.00E+00	1.61913	1.62289	2.95222
330.871	383.346	39.6047	0.00E+00	1.62393	1.61724	2.94674

332.917	384.464	40.0408	0	1.61743	1.62066	2.94687
332.917	384.464	40.0408	0	1.61743	1.62066	2.94687
333.062	384.26	39.5887	5.68E-14	1.61721	1.62022	2.94894
333.062	384.26	39.5887	5.68E-14	1.61721	1.62022	2.94894
332.484	383.595	41.4821	0.00E+00	1.61842	1.61743	2.94273
333.913	383.326	40.3543	0	1.61469	1.61714	2.9492
334.26	381.184	39.9403	0	1.61434	1.61081	2.95438
334.26	381.184	39.9403	0.00E+00	1.61434	1.61081	2.95438
336.755	379.74	42.4121	0	1.6065	1.60564	2.95114
337.453	377.916	41.2589	5.68E-14	1.60528	1.60062	2.95849
337.453	377.916	41.2589	5.68E-14	1.60528	1.60062	2.95849
337.563	381.358	42.8764	0	1.60361	1.61037	2.94843
336.849	377.311	42.1882	8.04E-14	1.60683	1.59844	2.95509
338.551	376.134	41.6699	8.04E-14	1.6023	1.59517	2.96036
338.551	376.134	41.6699	8.04E-14	1.6023	1.59517	2.96036
336.802	377.12	39.807	0.00E+00	1.60784	1.59874	2.96394
340.67	377.346	40.9475	5.68E-14	1.59617	1.59908	2.96406
340.67	377.346	40.9475	5.68E-14	1.59617	1.59908	2.96406
337.655	376.95	39.4563	0.00E+00	1.60551	1.59839	2.9665
335.877	379.739	40.1217	0	1.60987	1.60646	2.95814
335.877	379.739	40.1217	0.00E+00	1.60987	1.60646	2.95814
334.722	378.499	39.1226	8.04E-14	1.61388	1.60303	2.96176
334.23	377.517	36.7998	0	1.6163	1.60084	2.97057
337.371	379.253	36.1666	0.00E+00	1.60691	1.60642	2.97487
337.371	379.253	36.1666	0.00E+00	1.60691	1.60642	2.97487
334.785	376.833	36.6087	0.00E+00	1.61486	1.59889	2.97302
334.693	378.626	38.4718	0	1.61415	1.60364	2.96382
334.693	378.626	38.4718	0	1.61415	1.60364	2.96382
335.167	376.425	39.4533	0.00E+00	1.61289	1.59672	2.96399
335.754	376.839	39.5537	5.68E-14	1.61105	1.59795	2.96386
335.754	376.839	39.5537	5.68E-14	1.61105	1.59795	2.96386
339.781	380.304	39.7942	8.04E-14	1.5985	1.6084	2.96377
337.149	376.678	38.1583	0.00E+00	1.60747	1.59801	2.97092
339.852	378.808	37.4236	0	1.59939	1.60471	2.97435
339.852	378.808	37.4236	0.00E+00	1.59939	1.60471	2.97435
339.508	380.045	38.0051	0.00E+00	1.59993	1.60824	2.9702
336.211	378.38	38.6068	0	1.60971	1.60292	2.96583
336.211	378.38	38.6068	0.00E+00	1.60971	1.60292	2.96583
336.301	379.376	38.5876	5.68E-14	1.60923	1.60593	2.96465
337.231	378.335	37.433	5.68E-14	1.60712	1.60323	2.97146
337.569	376.525	38.8381	0.00E+00	1.60605	1.59733	2.96917
337.569	376.525	38.8381	0.00E+00	1.60605	1.59733	2.96917
335.89	377.449	37.6787	8.04E-14	1.61115	1.60043	2.96994

334.746	378.705	38.8047	0	1.61387	1.60376	2.96262
334.746	378.705	38.8047	0.00E+00	1.61387	1.60376	2.96262
335.535	379.779	38.9804	5.68E-14	1.61126	1.60697	2.96162
335.351	380.699	37.9838	0	1.61191	1.6101	2.96348
335.351	380.699	37.9838	0	1.61191	1.6101	2.96348
338.262	378.636	38.5975	5.68E-14	1.60367	1.60376	2.96827
334.666	377.576	38.3755	5.68E-14	1.61449	1.60051	2.9656
335.304	378.155	39.4665	0	1.61212	1.6019	2.96183
335.304	378.155	39.4665	0	1.61212	1.6019	2.96183
336.281	378.391	38.997	0.00E+00	1.60937	1.60282	2.96452
336.275	378.262	37.037	8.04E-14	1.61006	1.60311	2.97163
336.275	378.262	37.037	8.04E-14	1.61006	1.60311	2.97163
335.354	379.136	39.236	0	1.61184	1.60494	2.96137
334.799	381.277	37.9614	0	1.6134	1.61183	2.96185
334.799	381.277	37.9614	0	1.6134	1.61183	2.96185
333.996	381.929	40.1312	8.04E-14	1.61487	1.61298	2.95221
334.618	381.329	39.5884	0	1.61338	1.6114	2.95592
332.712	379.232	39.2687	5.68E-14	1.61959	1.60505	2.95725
332.712	379.232	39.2687	5.68E-14	1.61959	1.60505	2.95725
336.218	379.275	37.6183	0	1.60982	1.60596	2.96809
334.183	380.335	39.5566	5.68E-14	1.6149	1.60838	2.95684
334.183	380.335	39.5566	5.68E-14	1.6149	1.60838	2.95684
335.519	382.202	39.5512	8.04E-14	1.61054	1.61411	2.9561
332.462	382.304	40.1713	5.68E-14	1.61929	1.614	2.94912
333.52	382.769	40.472	0	1.61594	1.61537	2.94905
335.115	384.08	36.1871	0	1.61234	1.621	2.96387
335.115	384.08	36.1871	0	1.61234	1.621	2.96387
331.953	387.396	39.4504	5.68E-14	1.61966	1.62981	2.94226
332.658	384.112	38.5895	0	1.61878	1.6201	2.95181
331.305	382.264	35.409	0	1.6243	1.61544	2.96292
331.305	382.264	35.409	0	1.6243	1.61544	2.96292
334.82	383.194	34.4132	5.68E-14	1.61398	1.61885	2.97066
334.82	383.194	34.4132	5.68E-14	1.61398	1.61885	2.97066
335.317	383.733	35.6209	5.68E-14	1.61201	1.62013	2.96665
333.269	386.106	35.0505	0	1.61757	1.62749	2.96081
332.142	385.035	36.658	0	1.6207	1.62356	2.9556
332.142	385.035	36.658	0	1.6207	1.62356	2.9556
334.209	384.942	34.6316	8.04E-14	1.61525	1.6241	2.96589
332.021	385.898	35.9818	0	1.62103	1.62644	2.95599
332.021	385.898	35.9818	0	1.62103	1.62644	2.95599
331.254	386.054	37.5151	0	1.62277	1.62632	2.94951
335.013	384.451	35.5143	8.04E-14	1.61275	1.62235	2.96529
335.15	385.471	35.0118	0	1.61222	1.62566	2.96541

335.15	385.471	35.0118	0	1.61222	1.62566	2.96541
331.755	386.08	36.4987	0	1.62161	1.6268	2.95356
330.899	386.956	37.4873	8.04E-14	1.62357	1.62908	2.94734
330.899	386.956	37.4873	8.04E-14	1.62357	1.62908	2.94734
330.15	384.911	37.1956	8.04E-14	1.62647	1.62282	2.95055
331.855	385.129	36.9594	8.04E-14	1.62142	1.62373	2.95397
331.855	385.129	36.9594	8.04E-14	1.62142	1.62373	2.95397
334.551	387.653	35.5269	5.68E-14	1.61321	1.63218	2.95887
336.505	387.471	36.3842	0	1.60727	1.63143	2.9599
332.034	385.793	35.9888	8.04E-14	1.62102	1.62611	2.95618
332.034	385.793	35.9888	8.04E-14	1.62102	1.62611	2.95618
332.423	385.01	36.3914	0	1.61996	1.6236	2.95699
331.468	384.282	36.6053	5.68E-14	1.62292	1.62122	2.95589
331.468	384.282	36.6053	5.68E-14	1.62292	1.62122	2.95589
331.055	385.381	36.8116	8.04E-14	1.62377	1.62448	2.95256
331.647	384.253	36.3847	5.68E-14	1.62246	1.62122	2.95696
331.647	384.253	36.3847	5.68E-14	1.62246	1.62122	2.95696
332.331	384.631	35.8894	5.68E-14	1.62049	1.6226	2.95909
335.535	385.618	34.5066	5.68E-14	1.6112	1.6263	2.96745
334.018	385.367	34.9091	0	1.61561	1.62531	2.96392
334.018	385.367	34.9091	0	1.61561	1.62531	2.96392
332.807	385.621	34.9225	0	1.61911	1.626	2.96124
333.23	385.601	35.2134	0	1.61778	1.62588	2.96113
333.23	385.601	35.2134	0	1.61778	1.62588	2.96113
333.148	385.151	35.5201	8.04E-14	1.61805	1.62438	2.96081
331.529	386.223	34.9939	8.04E-14	1.62271	1.62773	2.95756
336.071	385.72	35.1819	0	1.6094	1.62641	2.96603
336.071	385.72	35.1819	0	1.6094	1.62641	2.96603
335.923	384.363	35.0152	0	1.61025	1.62229	2.96861
334.798	385.234	35.2491	0	1.61325	1.62483	2.96444
334.798	385.234	35.2491	0	1.61325	1.62483	2.96444
331.881	383.636	36.2479	0	1.62198	1.6194	2.95888
330.532	383.508	36.9841	5.68E-14	1.62578	1.61864	2.95437
330.532	383.508	36.9841	5.68E-14	1.62578	1.61864	2.95437
330.254	385.198	36.8383	0	1.6262	1.62384	2.95135
332.379	384.591	36.9356	5.68E-14	1.62003	1.62213	2.9559
331.831	384.458	37.152	8.04E-14	1.62161	1.6216	2.95448
331.831	384.458	37.152	8.04E-14	1.62161	1.6216	2.95448
331.322	384.734	35.766	5.68E-14	1.62349	1.62288	2.95748
330.69	384.072	35.7768	5.68E-14	1.62555	1.62079	2.95748
330.351	384.506	36.358	0	1.62625	1.62189	2.95427
330.883	386.244	35.9943	0	1.62431	1.6274	2.95323
330.594	384.913	35.1357	9.85E-14	1.62581	1.62357	2.95778

330.594	384.913	35.1357	9.85E-14	1.62581	1.62357	2.95778
329.532	384.509	37.0525	0	1.62847	1.62158	2.95059
330.433	384.559	35.3263	0	1.62632	1.6224	2.95754
334.087	385.379	35.8863	0	1.61511	1.62502	2.9609
334.087	385.379	35.8863	0	1.61511	1.62502	2.9609
335.309	385.988	35.4845	0	1.61147	1.6271	2.96328
331.552	385.186	35.7867	0	1.62268	1.62428	2.95703
331.552	385.186	35.7867	0	1.62268	1.62428	2.95703
334.237	384.918	35.8525	0	1.6148	1.62363	2.96206
331.026	384.48	36.135	0	1.62433	1.62195	2.95623
331.026	384.48	36.135	0	1.62433	1.62195	2.95623
330.573	385.033	35.9815	0	1.62557	1.62365	2.9549
333.209	385.271	36.0913	0	1.61766	1.62456	2.95889
333.209	385.271	36.0913	0	1.61766	1.62456	2.95889
330.992	386.175	36.2555	0	1.62392	1.62711	2.95275
332.18	384.413	36.3356	0	1.62086	1.62177	2.95778
332.629	384.345	36.2554	0	1.61957	1.62163	2.95894
332.629	384.345	36.2554	0	1.61957	1.62163	2.95894
334.898	384.77	35.9979	8.04E-14	1.61285	1.62316	2.96298
332.331	384.337	35.6462	5.68E-14	1.62065	1.62178	2.96038
332.592	384.365	35.9229	5.68E-14	1.61978	1.6218	2.95991
332.592	384.365	35.9229	5.68E-14	1.61978	1.6218	2.95991
332.418	384.4	35.6209	0	1.62038	1.62199	2.9605
333.5	384.808	35.0726	0	1.61724	1.6235	2.96346
333.5	384.808	35.0726	0	1.61724	1.6235	2.96346
331.692	384.654	37.3371	0	1.62191	1.62213	2.9533
331.09	385.173	37.1193	5.68E-14	1.62363	1.62374	2.95202
331.09	385.173	37.1193	5.68E-14	1.62363	1.62374	2.95202
331.88	386.583	36.8465	0	1.62098	1.62824	2.9518
330.61	386.292	37.2509	0	1.6247	1.62709	2.94874
330.61	386.292	37.2509	0	1.6247	1.62709	2.94874
331.194	384.732	36.4048	0	1.62367	1.62264	2.95524
330.995	385.181	37.7892	5.68E-14	1.62369	1.62352	2.94971
331.014	385.961	37.1918	5.68E-14	1.62361	1.62613	2.95026
331.014	385.961	37.1918	5.68E-14	1.62361	1.62613	2.95026
330.61	384.282	36.9181	8.04E-14	1.62536	1.62103	2.95337
331.086	384.024	37.3439	0	1.62388	1.62014	2.9533
331.086	384.024	37.3439	0	1.62388	1.62014	2.9533
329.512	385.004	37.9063	5.68E-14	1.6281	1.62279	2.947
330.287	384.804	37.7246	0	1.62591	1.62232	2.94932
330.287	384.804	37.7246	0	1.62591	1.62232	2.94932
331.208	384.173	37.2292	0	1.62351	1.62065	2.95363
331.334	384.373	38.0263	5.68E-14	1.62282	1.62099	2.95094

331.111	383.424	35.925	5.68E-14	1.62442	1.61879	2.95891
331.111	383.424	35.925	5.68E-14	1.62442	1.61879	2.95891
331.261	383.797	37.338	0	1.62342	1.61947	2.95402
331.056	385.446	37.8418	0	1.62341	1.62432	2.94919
331.056	385.446	37.8418	0	1.62341	1.62432	2.94919
331.83	385.329	37.3451	0	1.62132	1.6242	2.95235
331.675	386.699	36.6858	0	1.6216	1.62863	2.95172
331.675	386.699	36.6858	0	1.6216	1.62863	2.95172
331.241	384.005	36.4538	0	1.62371	1.62041	2.95645
330.585	383.063	37.1093	5.68E-14	1.62569	1.61724	2.95483
333.24	385.19	37.3115	8.04E-14	1.6172	1.6239	2.95516
333.24	385.19	37.3115	8.04E-14	1.6172	1.6239	2.95516
333.676	384.947	36.7077	0	1.61618	1.62339	2.95828
332.16	385.041	36.3586	0	1.62074	1.62369	2.95658
332.16	385.041	36.3586	0	1.62074	1.62369	2.95658
331.989	384.584	36.8738	0	1.6212	1.6221	2.95543
330.772	384.68	37.1985	8.04E-14	1.62468	1.62217	2.95207
331.745	385.487	36.8975	0	1.62167	1.62484	2.95334
331.745	385.487	36.8975	0	1.62167	1.62484	2.95334
330.865	384.546	37.7825	5.68E-14	1.62425	1.62157	2.95061
331.704	384.347	37.2228	0	1.622	1.62122	2.95422
331.704	384.347	37.2228	0	1.622	1.62122	2.95422

Dynamic tests 2 and 3 available on request.