NASA/TP–20210026807

Spatial Grid-Based Object Localization from a Single Passive Sensor: A Deep Learning-Integrated Approach

Patrick D. Geitner
University of Rochester, Rochester, New York

Joshua Gundugollu
Georgia Institute of Technology, Atlanta, Georgia

Douglas M. Trent
Science Applications International Corporation, Hampton, Virginia

July 2022

NASA STI Program Report Series

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

- Help desk contact information:

https://www.sti.nasa.gov/sti-contact-form/

and select the "General" help request type.

NASA/TP–20210026807

Spatial Grid-Based Object Localization from a Single Passive Sensor: A Deep Learning-Integrated Approach

Patrick D. Geitner
University of Rochester, Rochester, New York

Joshua Gundugollu
Georgia Institute of Technology, Atlanta, Georgia

Douglas M. Trent
Science Applications International Corporation, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

July 2022

Available from:

NASA STI Program / Mail Stop 148

NASA Langley Research Center

Hampton, VA 23681-2199

Fax: 757-864-6500

Spatial Grid-Based Object Localization from a Single Passive Sensor: A Deep Learning-Integrated Approach

Patrick D. Geitner
NASA Langley Research Center, Hampton, Virginia
pdgeitner@gmail.com

Joshua Gundugollu
NASA Langley Research Center, Hampton, Virginia
joshuagundugollu@gmail.com

Douglas M. Trent
NASA Langley Research Center, Hampton, Virginia
douglas.m.trent@nasa.gov

Abstract

Ongoing efforts at NASA's Langley Research Center have produced a single passive sensor system for detecting ground objects and pinpointing their real-world location to a desired level of precision. The Langley center serves as a test range for unmanned aerial systems (UAS) and real-time knowledge about the location of people on campus is needed to inform least-risk UAS flight operations.

The proposed system provides this knowledge through a camera combined with a convolutional neural network and an algorithm that projects an imaginary grid of square cells from the ground plane onto the perspective view of the camera. The position of detected objects on the camera's projected grid determines their location in the real-world. The imaginary grid is easily mapped to a universal coordinate system, such as longitude and latitude, to provide both relative and absolute positional information of the detected objects.

This simple system is shown to be accurate and effective, with decisive advantages over alternative multi-sensor and active sensor approaches. Extensions to the system are described to allow adaptation to a variety of other use cases.

TABLE OF CONTENTS

1. Introduction

1.1. Use Case

NASA's Langley Research Center (LaRC) is collaborating with the Federal Aviation Administration and industry partners to conduct research, test innovative technologies, and develop standards for the future operation of small unmanned aircraft systems (UAS) in urban environments. The LaRC campus itself will serve as a test facility for this effort through its City Environment Range Testing for Autonomous Integrated Navigation (CERTAIN) flight range. To minimize risk to Langley's human population, operators of UAS drones need to understand the moment-to-moment location and movement of people on the ground to better plan flight paths and identify safe-to-ditch locations.

1.2. WAHLDO System

The use case above is the motivation for the Wide Area Hazard Locator for Drone Overflight (WAHLDO) system that employs cameras and machine learning to find and localize people along critical corridors of the Langley campus. WAHLDO provides accurate real-time, on-demand knowledge of people locations to mitigate safety risks during UAS flight operations. The WAHLDO system depends on an innovative approach to object localization that is discussed in this report.

2. Design Approach

2.1. Detection and Localization

The WAHLDO system consists of an object detection component and an object localization component. Object detection is accomplished fairly easily. Advancements in deep learning and convolutional neural networks have created models that can detect a large variety of objects and identify their position within a two-dimensional (2D) image. Model architectures such as the Mask Region-based Convolutional Neural Network (Mask R-CNN) developed by Facebook AI Research are able to identify the exact pixels in an image that comprise a detected object.[1] Much more difficult is the task of using the 2D positional information to calculate where the detected objects are located in the three-dimensional space of the real world.

2.2. Alternative Solutions

Several methods were explored for determining the real-world location of objects detected in a camera image:

Optics Equations. In a thin lens optics system, the Lens Equation and Magnification Equation can be used to determine an object's distance.[2] Object distance (d_o) is calculated from object height (h_o), image height (h_i), image distance (d_i), and focal length (f) of the lens, as shown in **Figure 1**.

$$\textit{Lens Equation:}\quad \frac{1}{f} = \frac{1}{d_o} + \frac{1}{d_i} \qquad \textit{Magnification Equation:}\quad M = \frac{h_i}{h_o} = -\frac{d_i}{d_o}$$

$$d_o = \frac{f \cdot d_i}{f - d_i} \qquad d_o = \frac{-d_i \cdot h_o}{h_i}$$

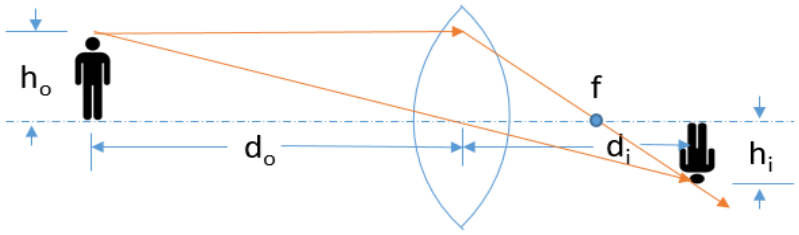

Figure 1. Deriving object distance from camera parameters and known object height.

Thus, for example, knowing the parameters of the camera and the height of a golf flag pin, the distance to the green may be estimated. This method fails when the camera parameters are not easily ascertained, and objects of unknown height are being detected.

Deep learning for distance estimation. When camera parameters are not available, deep learning models may be trained to estimate the distance from the camera to the detected object. An example is the DisNet architecture developed by Haseeb et al.[3] Correct distances are labeled for objects of various classes and the model learns the average height of an object class and the latent camera parameters. Once an object is detected and its class is identified, the distance to the object can be inferred. The main problem with this approach is that any error in an object's estimated height will have a significant effect on the predicted distance. The LaRC use case detects humans which inherently have different heights. Furthermore, human objects may be in different poses that greatly vary their perceived height. Height errors of 50% or more could be expected for the human object class and an equal error in distance estimation would result.

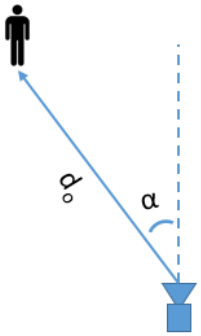

Figure 2. Object angle, α, from camera sight line is needed for localization.

In addition to the drawbacks discussed, these two alternative methods are incomplete. They provide ranging information for their detected objects, but do not provide direction. To properly localize detected objects in three-dimensional space, the direction angle **α** of the object from the sight line of the camera would also have to be measured, as illustrated in **Figure 2**.

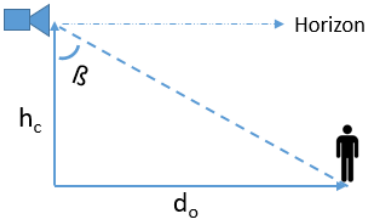

Figure 3. Object distance may be determined by camera orientation.

Sensor trigonometry. If the direction angle **α** is known**,** even when object height and camera internal parameters are not. a trigonometric approach may be taken by leveraging knowledge of the camera's external orientation. When the camera is pointed in the direction of the object, as in **Figure 3**, the height of the camera above ground $\mathbf{h_c}$ and the angle up to the object **ß** may be used to calculate object distance. Note that camera angle **ß** varies from zero degrees when the camera looks straight down, to ninety degrees when looking at the horizon. The object distance $\mathbf{d_o}$ is given as:

$$d_o = h_c \cdot \tan ß$$

When the camera is not pointed in the direction of the object, two trigonometric functions are combined, one in the vertical plane (angle **ß**) and one in the horizontal plane (angle **α**). This situation is illustrated in **Figure 4**. The object distance is now given as:

$$d_o = \frac{h_c \cdot \tan ß}{\cos \alpha}$$

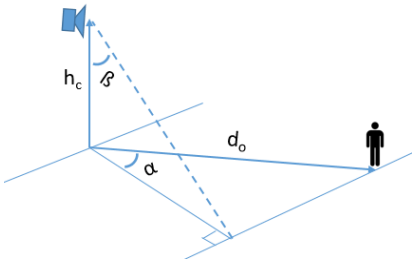

Figure 4. Object distance from two angles.

With the sensor trigonometry method, both angles need to be measured precisely, especially as the angles approach 90 degrees. If either angle is off slightly, large discrepancies in estimated distance may result.

For each of the above three methods, once both distance and direction are obtained, that information would need to be mapped to a common coordinate system such as latitude and longitude, to provide a meaningful summary of detected object counts. A final common problem with the three methods is that they defer the calculation of distance and direction and coordinate mapping until run time. That is, these calculations must be performed repeatedly for every instance of detected object, at the time of detection.

The use case at Langley requires a simpler, faster, and more accurate solution to the object localization problem. To optimize performance, a system that can calculate distance, direction, and perform coordinate mapping in a single operation in advance is desirable. This system should also function independently of any knowledge of object height, the internal parameters of the camera, or the camera's external orientation.

2.3. Proposed Method: A Spatial Grid

LaRC Park is a previous effort between NASA Langley and the Aerospace Systems Design Laboratory (ASDL) at the Georgia Institute of Technology. The project uses fixed cameras and a Mask R-CNN deep learning model to determine parking lot occupancy at LaRC by detecting vehicles in an image where known parking zones have been labeled. As shown in **Figure 5**, the occupancy of various parking zones (handicap, visitor, compact car, etc.) is determined from the intersection of a labelled parking zone (in yellow) with model-generated bounding boxes around detected vehicles (in green). This method provides the intuition for a new approach to object localization using a single passive sensor.

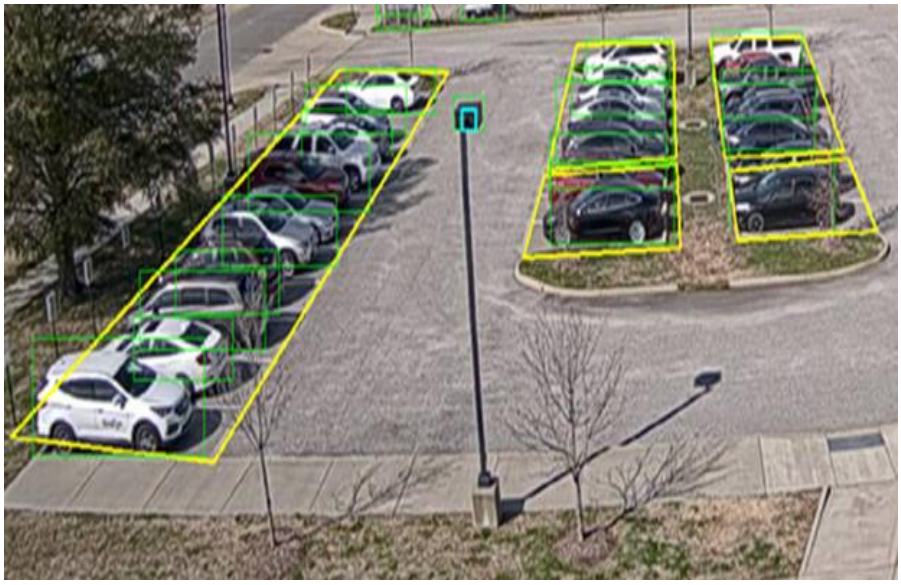

Figure5. LaRC Park: R-CNN parking lot occupancy detection at NASA Langley

If a stationary camera placed above ground level could observe a grid of square cells, physically situated on the ground plane, it would be a simple matter to localize detected objects on the labeled grid. The cells would be contiguous and uniform, but of arbitrary size, depending on the desired level of location accuracy, the distance of objects being detected, and the resolution of the camera. The cells could also be aligned and labeled according to a larger coordinate system, such as longitude and latitude

While physically drawing a permanent grid of cells on the ground would be impractical, could such a grid be visualized virtually and projected into the camera's view? The trick would be transposing the grid of square cells into the proper perspective of the camera, given its orientation to the ground. This is the key design challenge addressed in the following sections. The theory of invariant transforms is presented, equations to calculate the transposed grid are derived, an algorithm to draw the grid is proposed, and assumptions, limitations and advantages are discussed. The report concludes with examples and ideas to extend the design to other use cases as a general-purpose object detection and location system.

3. Theory

3.1. Object vs. Image Space

For consistency and clarity, the three-dimensional real world will be referred to as *object space* and the camera's two-dimensional view onto the real world will be called *image space*. As shown in **Figure 6**, a uniform grid of square cells in object space (on the left) will appear flattened and foreshortened in the image space (on the right) of a camera looking out onto the grid from above.

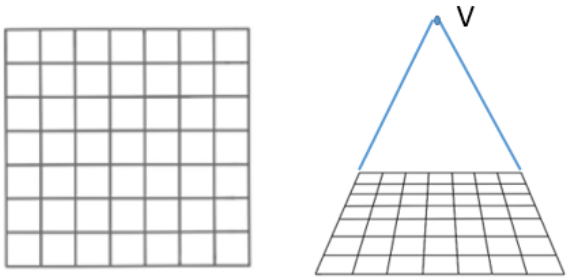

Figure 6. A grid of square cells in object space (left) and the same grid as viewed in image space (right).

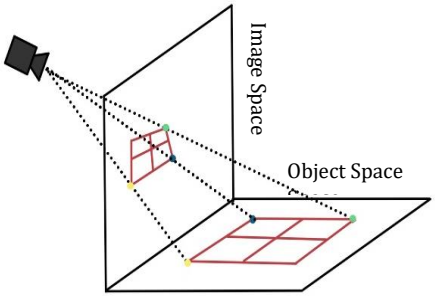

Figure 7. How a ground plane grid in object space is projected into the image space of the camera.

In object space, the lines of a grid of square cells are parallel. In image space, the lines converge to a point V on the horizon line called a *vanishing point*. This convergence and foreshortening in 2D image space preserves information about distances in the 3D object space. The task at hand is how to project an object-space grid from the ground plane into image-space, maintaining the perspective of the camera. This task is conceptualized in **Figure 7**.

Fortunately for present purposes, in the 4th century AD the Greek mathematician Pappus of Alexandria defined a property of four collinear points called the Cross-Ratio. This ratio is invariant to projective transformations and holds the key for projecting the object-space grid onto the camera's image space in proper perspective.[4]

3.2. Cross-Ratio

The Cross-Ratio equation is defined below for collinear points A, B, C, and D as illustrated in **Figure 8.**

$$Cross\ Ratio(A,B,C,D) \equiv \frac{\overline{AC} \cdot \overline{BD}}{\overline{AD} \cdot \overline{BC}}$$

Despite a projective transformation that maps points A, B, C, D to A', B', C', D' in **Figure 8,** the cross ratios of the two sets of points are identical. Using this property, the cross-ratio value for the grid in object space will be the same as the cross-ratio value of corresponding points in image space. A deeper explanation of the cross-ratio and its projective invariance is offered in **Appendix A.**

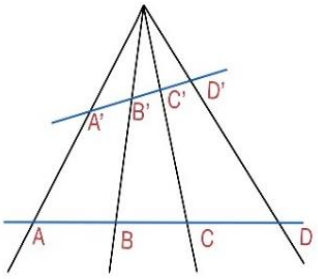

Figure 8. Collinear points A', B', C', D' are a projective transformation of A, B, C, D.

Since the square cells of the object space grid have sides of equal length, the cross-ratio of four adjacent points on the grid (three collinear line segments) will have a value of $4/3$. Therefore, the cross-ratio for any three adjacent line segments in image space must also be $4/3$. This is explored further in **Appendix B.**

Looking at **Figure 9**, if two of the line segments in image space are known, say, $\overline{AB}$ and $\overline{BC}$, then the third line segment $\overline{CD}$ can be calculated to satisfy the invariant cross-ratio value of $4/3$. Similarly, once $\overline{CD}$ is known, the next segment $\overline{DE}$ can be calculated. This process can be repeated to calculate the ever-decreasing line segment lengths towards the vanishing point. The equations for using the cross-ratio to calculate adjacent line segment lengths, both towards and away from the vanishing point, are derived in **Appendix B**.

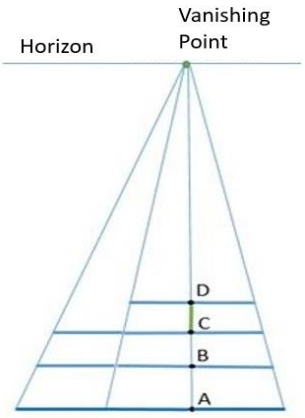

Figure 9. Perspective grid line segments in image space.

In this manner, the cross ratio is used to translate the square grid from object space into a perspective grid in image space, while preserving relative distance information. Since three adjacent collinear points are needed to generate the next point of the image-space grid, a total of nine points, or four complete grid cells are needed as initial input (see **Figure 10**). The method does not work if fewer than the nine points are known. While this approach is functional, it will be more convenient if fewer initial points are required to generate the entire image-space grid. The next section discusses a short cut to reduce the required minimum initial points from nine down to four.

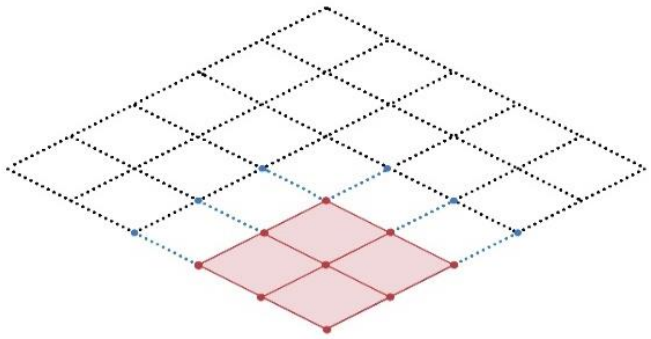

Figure 10. Using the initial approach to construct the image-space grid from nine defined points, marked in red.

3.3. Vanishing Points

In the image-space representation of the grid, the lines in the grid of squares will converge to a vanishing point on the horizon. Since the vanishing point is collinear to points that converge to it, it can function as one of the collinear points to calculate new points using the cross ratio. The location of the vanishing point can be determined by finding the intersection of the line segments that converge in image space.

Note that the vanishing point is an imaginary concept. In object space, the grid's line segments are parallel. The vanishing point is infinitely far away. A line segment formed by a grid point and the vanishing point will have infinite length. Thus, in calculating the cross-ratio, a limit must be used. Working with collinear points A, B, and C and vanishing point, V, the cross-ratio calculation would be as follows:

$$Cross\ Ratio(A,B,C,V) = \frac{\overline{AC}\cdot\overline{BV}}{\overline{AV}\cdot\overline{BC}} = \lim_{x\to\infty}\left(\frac{\overline{AC}\cdot x}{\overline{BC}\cdot x}\right) \to \lim_{x\to\infty}\frac{\frac{d}{dx}[\overline{AC}\cdot x]}{\frac{d}{dx}[\overline{BC}\cdot x]} = \frac{\overline{AC}}{\overline{BC}},\ by\ L'Hôpital's\ rule$$

Since the line segments that make up the grid of square cells all have the same length in object space, the cross-ratio will equal 2. With this known value for the cross-ratio, only two collinear points and a vanishing point are needed to calculate subsequent collinear points. Since the vanishing point is pointed to in image space by the opposite sides of a grid cell, the entire perspective grid can be generated from a single initial cell of four points. The equations for building out the grid using this vanishing point approach are given in **Appendix C.**

Figure 11 illustrates the vanishing point approach. The initial points of A, B, C, and D are given as input. The intersection between lines $\overline{AB}$ and $\overline{DC}$ will determine the left vanishing point L. The three collinear points: A, B, and L can then be used to calculate the next point E that maintains a cross-ratio value of 2. The equations for this calculation are given in **Appendix C**. This process can be repeated to calculate all necessary points towards and away from each vanishing point to build out the entire perspective grid. An algorithm to perform this task is discussed in the next section.

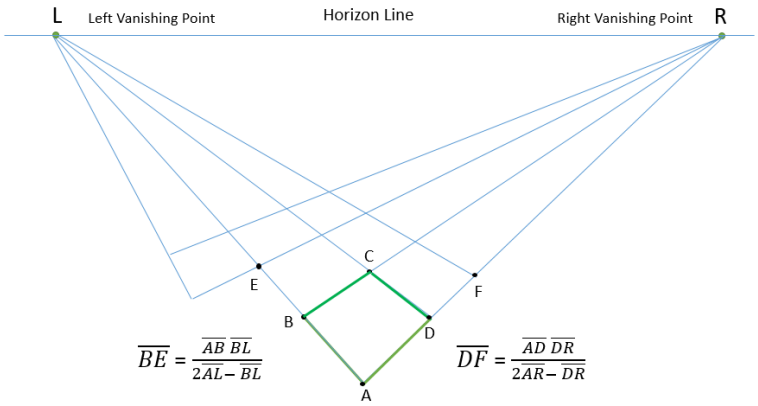

Figure 11. Grid construction with the short cut approach. Only four points, A, B, C, and D are needed to construct the entire perspective grid.

4. Algorithm Implementation

4.1. Considerations

Following the approach outlined in the previous section, an algorithm has been designed to generate a perspective grid from a single initial reference cell of four points. First, the following items must be considered to put the approach into practice:

Local Coordinate System. Typically, the initial reference square (points A, B, C, D in **Figure 11**) will be aligned to the universal coordinate system of interest. For example, if the chosen universal coordinate system is longitude and latitude, the sides of the reference cell will align to the four cardinal points of the compass. The algorithm considers the reference square to have the origin X,Y coordinate of (0,0). By convention, the ray from point A to the right vanishing point is considered the X axis, and the ray from point A to the left vanishing point is considered the Y axis. Adjacent cells along these rays to the vanishing points are labeled with increasing x and y values, as shown in **Figure 15**. Note that cells may also be constructed to the left and right of these rays and these cells will contain a negative coordinate. Combining input from the local coordinate systems of multiple sensors into a single universal coordinate system is discussed in **Section 4.5.**

One-point-perspective. The theory outlined in **Section 3** is based on the two-point-perspective case. In one-point perspective, there is a single vanishing point and one pair of a cell's opposite sides will point to it. The other pair of a cell's opposite sides will be parallel, and their intersection at a vanishing point infinitely remote. When the algorithm detects parallel sides, or a second vanishing point sufficiently distant, then all the line segments in a given horizontal row pointing to that distant vanishing point are given equal length.

Zero-point perspective. There is also a degenerate case that needs to be considered. This is the situation where the camera is pointing straight down and the reference cell is perfectly square, with two pairs of parallel opposite sides. In this case there are no vanishing points and the grid line segments in image space are parallel. The grid cells in image space will all be squares of equal size. Looking back at the central projection of **Figure 8**, this situation occurs when collinear points A', B', C', D' are parallel to points A, B, C, D.

Stopping Criteria. When the horizon is within the camera's view, since it is infinitely distant in object space, the algorithm would continue generating grid cells toward the vanishing point forever. Because of this, a stopping criterion is required. A pixel threshold is chosen to stop calculating new cell points that are closer than a given number of pixels from the previous calculated point. On the other hand, when the horizon is not in the image, a trade-off must be made between information gain and precision. If the algorithm stops generating new cells when any portion falls outside the image, there will be area on the image margins without grid cells where a detected object cannot be localized. Conversely, if the algorithm stops only when all of the cell would fall outside of the image, then the image will contain partial cells that cannot provide a full count of objects within their object-space perimeter.

4.2. Algorithm Description

With these considerations in mind, the grid generation algorithm takes the four corners of the reference cell as input to create the grid in the perspective of the camera's image space. The main steps of the algorithm are shown in **Figure 12.**

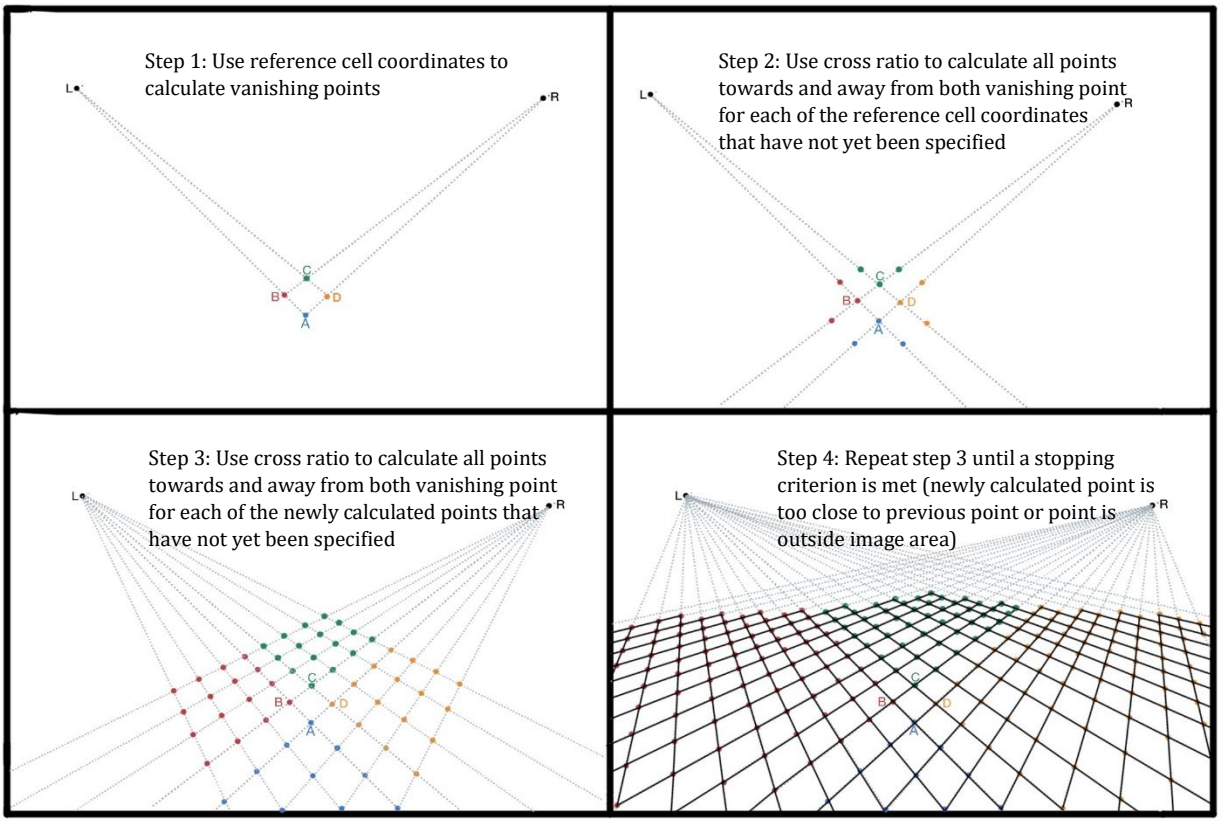

Figure 12. Steps taken by the algorithm to build a grid in the perspective of the camera from a single reference cell of four points.

Step 1. First, the algorithm uses the equations given in **Appendix D** to extrapolate the sides of the reference cell to their points of intersection, namely, the left and right vanishing points L and R, which are defined by their pixel coordinates. The four vertices of the reference cell A, B, C, D are then placed into a queue.

Step 2. Observe that each cell vertex on the grid has four neighbors: one each in the direction of the two vanishing points, and one each in the direction away from the vanishing points. The points in the queue are removed one at a time and for each neighboring point that is vacant, the cross-ratio formulas of **Appendix C** are applied to define the missing neighbor(s). Any newly defined points are added to the queue.

Step 3. Starting with point A from the queue, two new points are defined away from the vanishing points. Next, point B generates two new points, one away from vanishing point R and one toward vanishing point L. Point C generates two new points toward the vanishing points. These offspring points are labeled in **Figure 12** in the color of their parent reference cell point. The grid grows organically as the process spirals outward in clockwise fashion.

Step 4. When a newly defined point meets a stopping criterion, such as falling outside the camera image or getting too crowded toward the horizon, that point is discarded and not added to the queue. Eventually the queue is exhausted as points are processed and no more new points are added. A final pass deals with partially defined cells, either to complete them for information gain (**Figure 13**) or to dismember them for precision (**Figure 14**). The perspective grid is now complete.

4.3. Algorithm Grid Output

Figures 13 and **14** visualize the output of the perspective grid generation algorithm. The image is a photograph of a chess board and the initial reference cell is highlighted in blue. Despite the intentionally different camera angles, the algorithm is able to complete the grid with a high degree of accuracy for both images, as evidenced by the close grid alignment to the chess board cells.

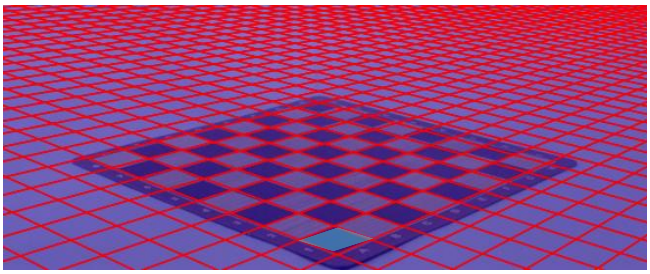

Figure 13. Algorithm output for level horizon and low camera angle. In this case, the algorithm is asked to generate all cells, even if a portion of a cell falls outside the camera image.

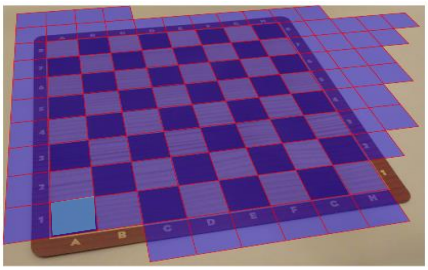

Figure 14. Algorithm output for tilted horizon and higher camera angle. Here, only cells that fall completely inside the image are generated.

4.4. Object Localization on Image Space Grid

Once the perspective grid has been generated, a deep learning model may be used to detect objects in the camera image and encompass them by a bounding box. Assuming the real-world object is resting on the ground, the center bottom of the bounding box may be taken as a reasonable single point to represent the object's location in image space. The grid cell that contains the object's single point representation is the cell that determines the object's location in object space.

A simple method to match objects with grid cells is to loop through the entire list of grid cells generated by the algorithm until one is found that contains the object. For example, the Contains_Point method of MatplotLib's polygon patch objects [5] may be used to check if an (X,Y) pixel point is within a polygon defined by grid cell vertices. This can be a compute-intensive process.

Depending on the size of the reference square as seen by the camera, the algorithm may generate many hundreds of grid cells. This is especially true if the camera's view contains the horizon. As cells approach the horizon, they become infinite in number. A technique to reduce the search space for matching an object's point representation to a specific grid cell is depicted in **Figure 15**. The sides of the initial reference cell $\overline{AB}$ and $\overline{CD}$ have been used previously to define the vanishing point L. Instead of checking each grid cell individually, the search space can be divided into rows of constant X or Y coordinate. For example, the polygon ELB could be checked for containing the object's point representation first, then ALD and then FLG. When a row of constant X is found that matches, then the individual cells on that row can be searched to find the exact Y coordinate.

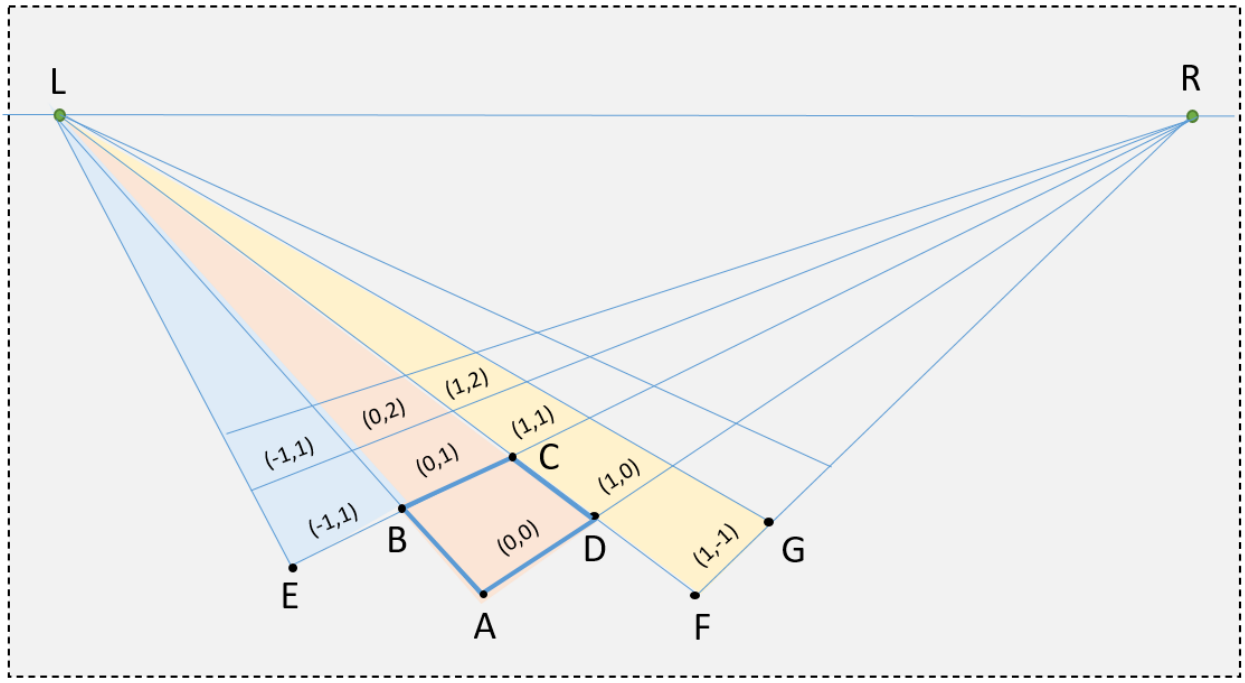

Figure 15. Limiting the search space to match an object's point representation to a perspective grid cell.

4.5. Mapping Local to Universal Coordinates

Returning briefly to the initial use case for WAHLDO, the goal is to have a comprehensive view of people locations for CERTAIN flight range operations across the Langley center. To meet this need, a universal grid has been defined for the center, aligned to the cardinal points of the compass (N, E, S, W). The grid fully encompasses the center and is sub-divided into 10x10-meter contiguous cells. The result is a rectangular grid 178 cells wide by 368 cells tall. Starting with cell (0,0) in the bottom left southwest corner, the cells are labeled with increasing X values to the east, and increasing Y values to the north, mimicking the first quadrant of a standard X-Y plot.

The Langley center has 764 acres and cannot be covered by a single camera. Moreover, only some areas are used for flight operations and are of interest. In practice, multiple cameras are required. The object localizations of each camera need to be combined into a single comprehensive view on the universal Langley grid.

The previous section describes the local coordinate system used by a camera's image-space grid. Mapping the camera's local coordinate system to a universal coordinate system involves a two-step process. As a transformation, the two steps can be thought of as a rotation and a translation:

Rotation. The first step of the process depends upon the yaw of the camera, or its angle of rotation around the vertical axis, relative to the coordinate system of the universal grid. In the LaRc setup, when the camera is pointing between North and East, the so-called "first quadrant," it has the same orientation as the universal grid and no rotational transformation is needed. If the camera is pointing between any of the other cardinal compass points, then its X,Y coordinates need to be swapped and/or negated, as shown in **Figure 16**. If a camera happens to point directly at a

cardinal compass point, it sees a single vanishing point in one-point perspective and the vanishing point is considered to be the left vanishing point. Thus, for example, a camera pointing due north is in the first quadrant.

Translation. The second step involves identifying the camera's reference square on the universal grid, and then applying that universal grid cell's coordinates as an offset to the camera's local coordinates. For example, if the camera's reference square is (2,1) on the universal grid, then 2 is added to all X values of the camera grid local coordinates, and 1 is added to all Y values. Now that the camera is on the same coordinate system as the universal grid, its object detection counts can be copied over to the universal grid.

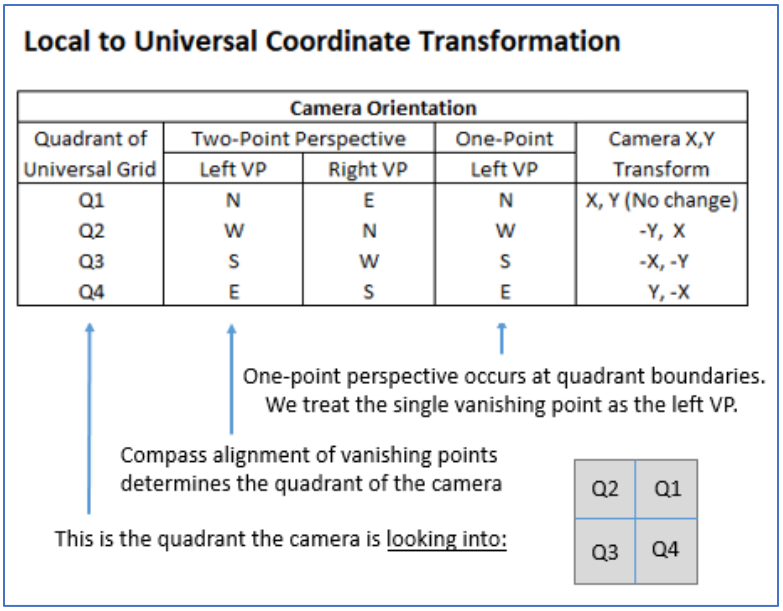

Figure 16. How to transform local grid coordinates to universal grid coordinates, depending on quadrant seen by the camera.

An example of this coordinate transformation process is given in **Figure 17**. Care must be taken to accept input from only one camera for each universal grid cell, to avoid double counting or overwriting.

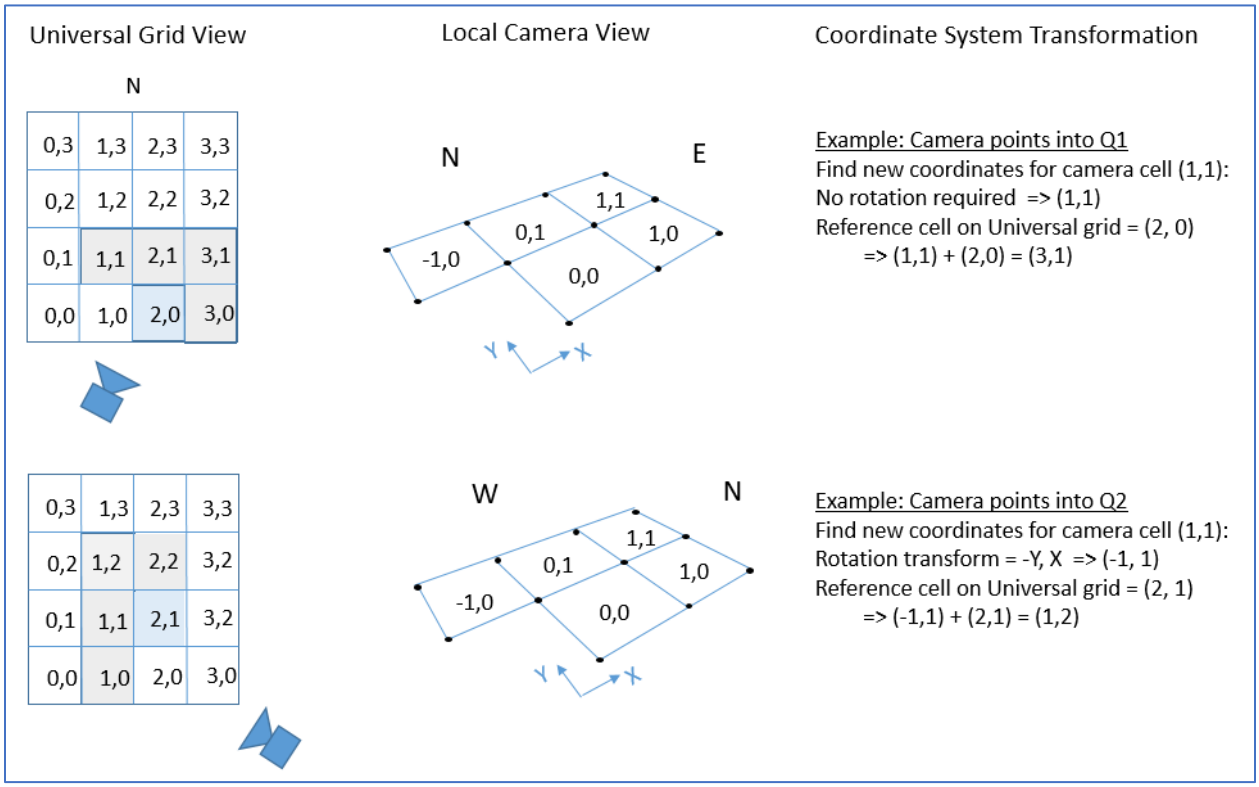

Figure 17. Process of transforming local camera cell coordinates to a universal grid system.

5. System Description

5.1. Overview

The algorithm of **Section 4** may be incorporated into a larger object detection and localization system as follows:

1. A stationary camera is positioned above ground level to observe an area where detecting and localizing objects is of interest.
2. A single reference cell of the grid is defined in the object space at the desired precision. Typically, the initial reference cell will align to some universal coordinate system, such as longitude and latitude. In **Figure 18**, a 15x18.5-foot initial cell is marked on the field, aligned with the yard markers and stadium orientation.
3. A single frame of the camera image is taken and the pixel coordinates of the reference cell corners, as viewed by the camera, are determined. These coordinates are the inputs to the perspective grid generation algorithm.

Figure 18. A single reference cell (marked in blue) of the object-space grid is defined on the camera image.
Photo by Al Soot on Unsplash

4. The perspective grid generation algorithm builds out the entire image-space representation of the grid. The grid is overlaid onto the camera's view. In practice, since the camera's position is fixed, the image-space grid is calculated once and stored in a database.
5. If needed, for example when multiple cameras are working in concert, the camera's grid cells may be labeled according to the coordinate system of a universal grid, using the transformation techniques described in **Section 4.5.**
6. At run time, a still frame from the camera is fed into a region-based convolutional neural network (R-CNN) to detect objects and find their positions in image space.
7. The intersection between the image-space position of detected objects and the overlaid perspective grid is used to determine the position of the objects in real-world object space.
8. The total count of detected objects in each grid cell is time stamped and recorded in a database.

5.2. Assumptions and Limitations

The object detection and localization system described above is implemented under the following simplifying assumptions:

Fixed camera orientation. The camera is assumed to be distortion-free and positioned at a stationary height and angle. If camera orientation drifts, the image-space grid will lose alignment with object space and localization error will increase, becoming worse for objects that are further from the camera. This can be remedied by recalibrating the grid vertices whenever camera orientation changes.

Camera separation from plane. The cameras must be positioned some minimal distance above the ground plane to obtain ranging information as well as direction. Higher camera angles also help avoid the situation where one object blocks another immediately behind it. As the camera is positioned closer to the ground, extreme foreshortening occurs that causes a loss of perspective and increased localization error. A camera placed directly on the ground would see the entire perspective grid collapse to an unusable horizontal line.

Unobstructed view. The objects to be detected are assumed to be within the line of sight of the camera, without obstruction from surrounding buildings, vegetation, or terrain. This is a requirement of almost all object detection systems unless reflections are used to reconstruct obstructed areas.[6]

Clear weather. A passive optical sensor depends on fair weather to see its target. Although RADAR performs well in inclement weather, this is a common problem for many passive and active detection systems, including LIDAR. For

night conditions and low-light operations, infrared cameras may be employed to detect objects having an infrared signature.

Objects on ground. Detected objects are assumed to be situated on the ground plane. Gravity helps reinforce this assumption. The bottom middle of a detected object's bounding box is taken as the point location of the object. If an object is floating above the ground, it will be localized farther away than reality.

Balanced Pose. The center-point of the bounding box touching the image-space grid is assumed to be an accurate point representation of the detected object's location. If the pose of the object extends the bounding box in a direction away from its center of mass, the point representation may be inaccurate.

Uniform grid. The derived equations based on the cross-ratio formula assume that grid cells are all squares or rectangles in object space, in order to have their opposite sides pointing to the same vanishing point in image space.

Additionally, the described approach has the following inherent limitations:

Grid precision. The +/- tolerance of localization precision is determined by the chosen grid cell sizes. In practice, obtaining the smallest possible cell size with least imprecision will depend on the camera's resolution, viewing angle, and distance from the area of interest.

Accurate object detection. Much depends on how accurately the R-CNN reports the location of a detected object in image space. Also, if the object detection model reports bounding box vertices in pixel integer values only, rounding error will be introduced. For its part, the grid construction algorithm avoids rounding error by defining cell vertices in floating point values.

Accurate calibration. The initial reference square is assumed to be accurate in dimension and alignment to the camera or to a universal grid. All grid cell construction and object localization measurements spring from the reference square. If the initial square is inaccurate, the garbage-in, garbage-out principle will apply.

Flat surface. The object-space ground plane, over which the grid is constructed, is assumed to be locally flat. The curvature of the earth has been ignored. If ground elevation deviates from level, objects at higher elevations will be localized further away than they really are and objects at lower elevations will be localized closer than reality. A method to relax the flat surface assumption is proposed in **Appendix E**.

6. Results

6.1. Advantages

The object localization system outlined above has a number of distinct advantages over alternative methods:

Single sensor. Various methods have previously been developed to determine relative positional information of objects in an image using multiple cameras.[7] However, this approach comes with the attendant challenge of uniquely identifying multiple objects as seen from different angles to triangulate their position. With a single sensor approach, this problem is avoided entirely.

Passive sensor. High precision object localization techniques have been created by leveraging LiDAR and Radar backscatter.[8] However, active sensor approaches have the challenge of $1/d^4$ signal attenuation -- $1/d^2$ outbound and $1/d^2$ returning. Also, active sensor solutions reveal the location of the detector, which may be undesirable in certain use cases. A passive sensor solution avoids the outbound attenuation problem and does not reveal the location of the detector.

Simplicity. In addition to the advantages of the single passive sensor outlined above, this approach does not require extensive apparatus. In many cases, existing equipment can be leveraged to implement the proposed object localization system as it only requires a stationary, above-ground camera.

Performance. The calculation-heavy parameters of the camera and geometry of its orientation are embedded in the image space grid. Because the camera orientation and parameters are fixed, the grid construction algorithm is run only once, in advance. Thereafter, only the detection component is run as needed. The distance-aware grid quickly localizes image-space items back into object space.

Flexible cell granularity. The selected size of the grid cells determines the granularity or "binning" of detected object counts, and the accuracy of their localization. Thus, the grid generation algorithm is flexible to tradeoff the convenience of larger bins (smoother histograms, faster search) with the greater localization precision of smaller cells, allowing for project-specific implementations.

Ethical AI. The system exhibits ethical artificial intelligence traits. Objects are identified as belonging to a class, but are not individually recognized. Only the counts of detected objects are stored, never any personally identifying information. The use of the perspective grid enables transparency and explainability. The grid provides visibility and insight into how the location of objects is determined. Both the grid and object detections can be visualized over the image to ensure the algorithm is operating correctly and to increase confidence in localizations.

Robust to camera orientation. The perspective grid generation algorithm is robust to the downward angle of the camera, and even its tilt relative to the horizon. Information about the camera's angle, tilt and height are not needed to generate the grid. The horizon does not need to be present in the camera's image.

Integration with machine learning. The perspective grid generation algorithm fits well with machine learning-based object detection algorithms. Because the system is modular with the grid generation and object detection occurring separately, a variety of task-specific object detection models can be used.

Distributed solution. While some active sensor solutions, like Radar, require an enclosure to protect the sensor and limit attenuation, cameras can be deployed over very wide areas with minimal infrastructure.

Coordinate-based system. Using the grid as a foundation abstracts away from dealing with angles and trigonometry in the image space, and instead provides a simple (x, y) coordinate system that is easy for other algorithms and applications to build upon.

Relative location information. By localizing to a grid coordinate system, the proposed method provides distance and direction information among all detected objects, as well as from the objects to the camera.

Universal Alignment. The grid-based system also permits easy mapping to a universal coordinate system such as latitude and longitude, providing absolute location information.

Digital Transformation. The essence of the proposed system is the digitization of object space into a grid and the transformation of that grid into the image-space perspective of the sensor. The result is a robust and flexible platform that can be applied to solve problems across many disciplines and is suitable for adaptations and extensions, as discussed in **Section 7.**

6.2. Examples

The perspective grid generation algorithm is easily integrated with a region-based convolutional neural network (R-CNN) model to form the proposed object localization system. Pre-trained implementations of R-CNN and Mask R-CNN are widely available. The current iteration uses the Matterport Mask R-CNN implementation[9] trained on the Microsoft COCO dataset[10] which includes 80 object categories including people. With the image-space bounding box coordinates provided by the model and the perspective grid vertices generated by the algorithm, object-space positional information can be extracted from the image by determining which grid cell the object is situated in.

Figure 19 shows an example of the integrated system in use. The example demonstrates the simplicity and efficacy of the proposed system. Starting with an above ground level camera and the definition of a single 15x18.5-foot reference cell, the system generates the perspective grid across the playing field to detect, count and localize people onto the grid. The people counts in object space are easily mapped to a universal coordinate system (in this case, the stadium, a rotation of about 75 degrees counter-clockwise) and displayed in the form of a heat map.

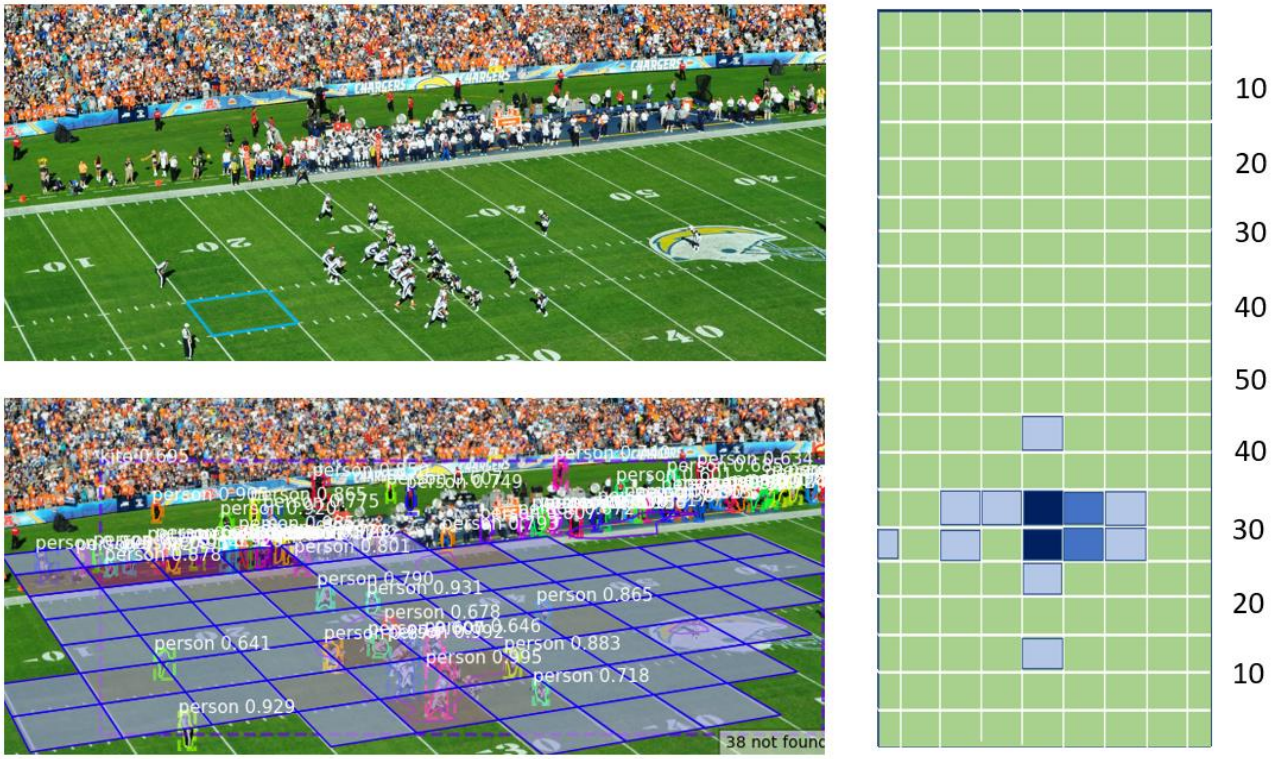

Figure 19. Starting with a single reference cell (top left in blue), the perspective grid generation algorithm calculates the remaining cells. Deep learning is used to detect people in the image. The detected objects are localized into grid cells and mapped to a universal grid.
Photo by Al Soot on Unsplash

The football field is a convenient example because of the measured markings on the field. However, previously marked terrain is not required. The system is easily configured for a space with no markings at all, which is the case for the Langley CERTAIN range application. **Figure 20** shows how the process is accomplished.

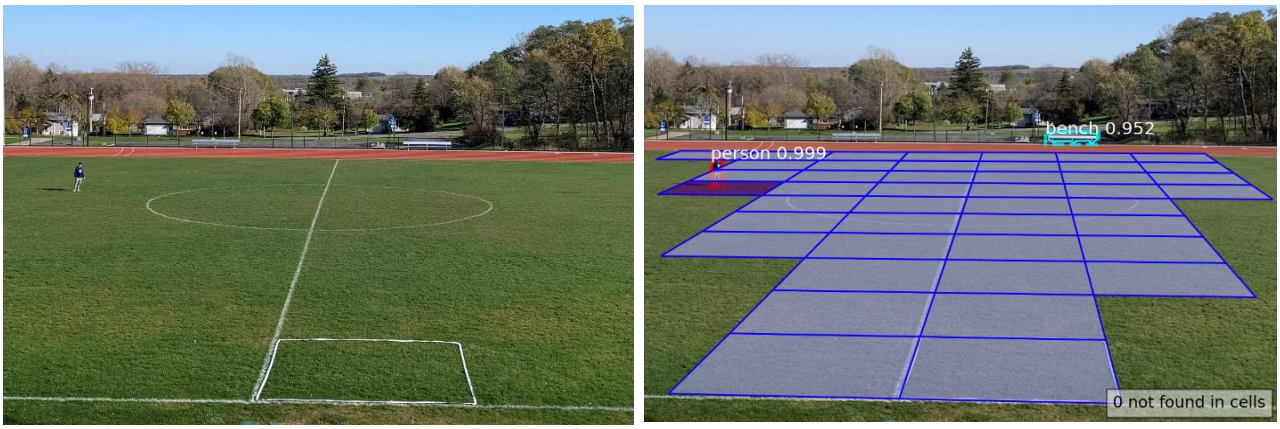

Figure 20. A 5x5 meter reference cell is used to generate a perspective grid and localize objects to its granularity.

A 5x5-meter reference cell is taped to the ground in the area of interest, providing the necessary input for the algorithm to generate the grid in perspective across the entire field. Since the camera is stationary, the grid is generated once, and the pixel coordinates of the cell vertices are stored in a database. For subsequent object detections, the database is queried to localize objects without recalculating the grid.

The grid generation algorithm is flexible to accept reference cells of varying sizes. If more granular location information is required, a smaller reference cell may be chosen. In **Figure 21**, a 1x1-meter reference cell was taped onto the same area of interest to generate a denser grid with increased precision. The algorithm easily handles the increased grid density, but the localization system becomes more susceptible to any small errors in the pixel locations generated by the object detection model.

There is also a general tradeoff between precision and latency. The number of grid cells must increase by the square of the precision distance desired (eg, 1-meter precision requires 25 times as many cells as 5-meter precision). Compute time increases with larger cell counts. For each object located, its point representation can be compared to a list of cell polygons to see if the point is contained within. A method for reducing the search time is given in **Section 4.4**.

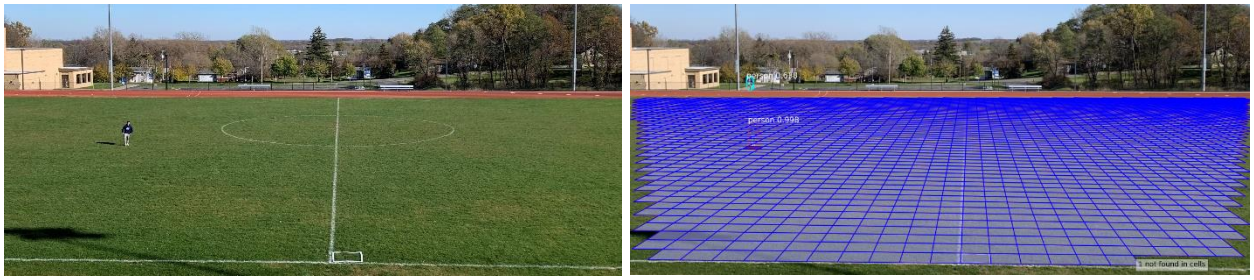

Figure 21. The same region of interest with perspective grid generated from a 1x1-meter reference cell.

The above examples demonstrate the ability of the system to detect and localize objects on the ground plane to a desired level of precision. The next section describes the robustness and flexibility of the system to be repurposed to a variety of additional use cases.

7. Extensions

7.1. Other Use Cases

The described object localization method can be extended to a variety of use cases in many disciplines by relaxing certain assumptions and making slight changes to the system implementation:

Sensor-based origin. The examples described so far have involved stationary cameras and have assumed the object location information being sought has a universal measure, such as longitude and latitude or a fixed (X,Y) grid. Other use cases may be more interested in measuring location relative to the sensor, and the sensor may be mobile, while maintaining the same vertical distance from the plane of interest. Examples include a tank surveying a battlefield, or the bridge of a ship scanning the ocean ahead for traffic. Under the sea, a submarine at known depth could generate an inverted perspective grid, superimposing it on the view of the water surface from below. This would enable localization of floating objects, as shown in **Figure 22**.

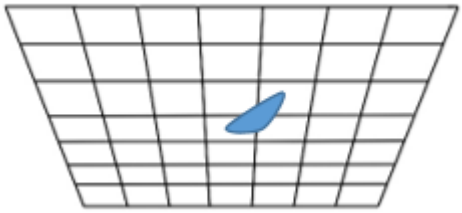

Figure 22. A submarine's inverted perspective grid view of a ship on the ocean surface above.

Object motion prediction. The grid-based approach provides a convenient way to analyze and predict the motion of a detected object. Various methods can be developed and employed to predict an object's probability of traversing a string of adjacent grid cells over a given period of time. One method would be to train a recurrent neural network that uses information about an object's current state and previous states to predict the probability of a given path, similar to the work done by Kim et al. for vehicle trajectory prediction.[11] **Figure 23** illustrates how this could be implemented for modeling the motion of a detected human.

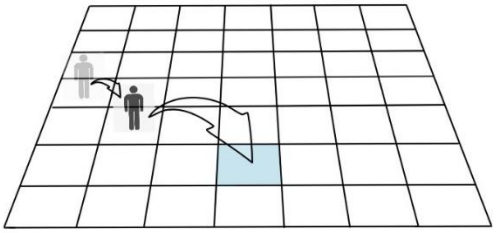

Figure 23. Grid-based trajectory modeling of a detected object's motion

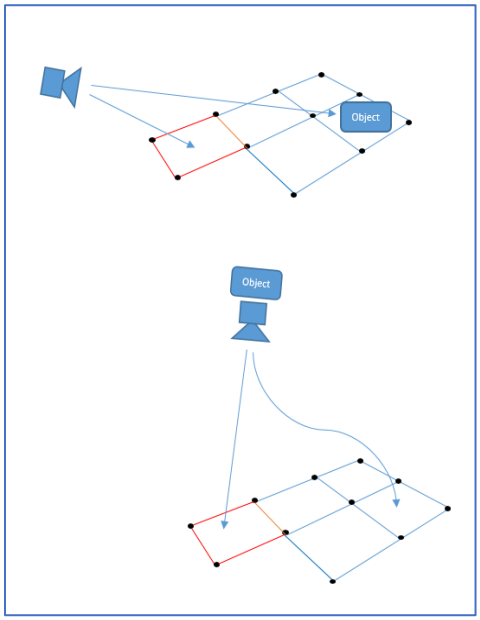

Figure 24. Camera attached to the object to determine position and attitude relative to the surface.

Pose estimation. The current stationary camera system detects the location of an object relative to a known reference square, but does not determine the orientation of the detected object. However, the sensor could be attached to the object itself, to observe the reference square on the ground below, as depicted in **Figure 24**. Then, the object/camera could detect its own orientation relative to the surface, both in position and attitude:

Position information could be gleaned from the sensor's distance and angle to the reference square. Distance would be estimated from the relative size of the square, more distant squares being smaller. Angle would be estimated from the proximity of the reference cell to the horizon. The closer the sensor is to the ground, the smaller the viewed distance from the reference cell to the horizon.

Attitude information would also come from the reference cell. Extending its opposite-side line segments establishes the two vanishing points and these define the horizon line, which yields object pitch and roll information. Camera angle relative to the grid cell rays to the vanishing points determines yaw.

As the object moves, the grid would need to be continuously recalculated to provide new position and attitude information. This method could be used, for example, to assist a UAS drone landing itself at a Vertiport, or a spacecraft landing itself on the lunar surface.

7.2. Extension to 3D

The current method uses a two-dimensional (2D) grid of cells to ascertain location information in the 2D ground plane where objects are assumed to reside. If a third dimension were added to the grid generation algorithm, an entirely new class of use cases could be contemplated for objects that were not required to be resting on the ground. For example, given the location of a flying object, its altitude could be determined. Or given the altitude of an object, its location could be calculated. In both these cases, and also in the case of objects on the ground, the height of an object could be estimated. Finally, with knowledge of the third dimension embedded in the perspective grid, the assumption of a flat ground plane could be relaxed. The 2D perspective grid could be adjusted to reflect uneven terrain, to deliver even more accurate localization results.

The 3D capability is obtained by adding one more data point to the reference cell, as described in **Appendix E**. The new data point provides a height reference at the foreground-most corner of the reference cell. From there on, that reference height is scaled down at each cell vertex according to the decreasing ratio of distance to the vanishing points. The grid generation algorithm stores the changing value for height at each cell vertex. The value is stored in specific unit terms, say object-space meters per image-space pixel. The reference height value for a particular cell may be taken as the average reference heights of its four vertices.

Armed with the reference height at each cell and vertex, the following may be derived:

Altitude of object above ground. Given the location of the object on the perspective grid (ie, the cell it is hovering directly above) a line may be drawn from that location cell to the object and measured. The line should be perpendicular to the horizon. The pixel height of the line multiplied by the reference height at the cell yields the object altitude above ground. This is the case for object 1 in **Figure 25**.

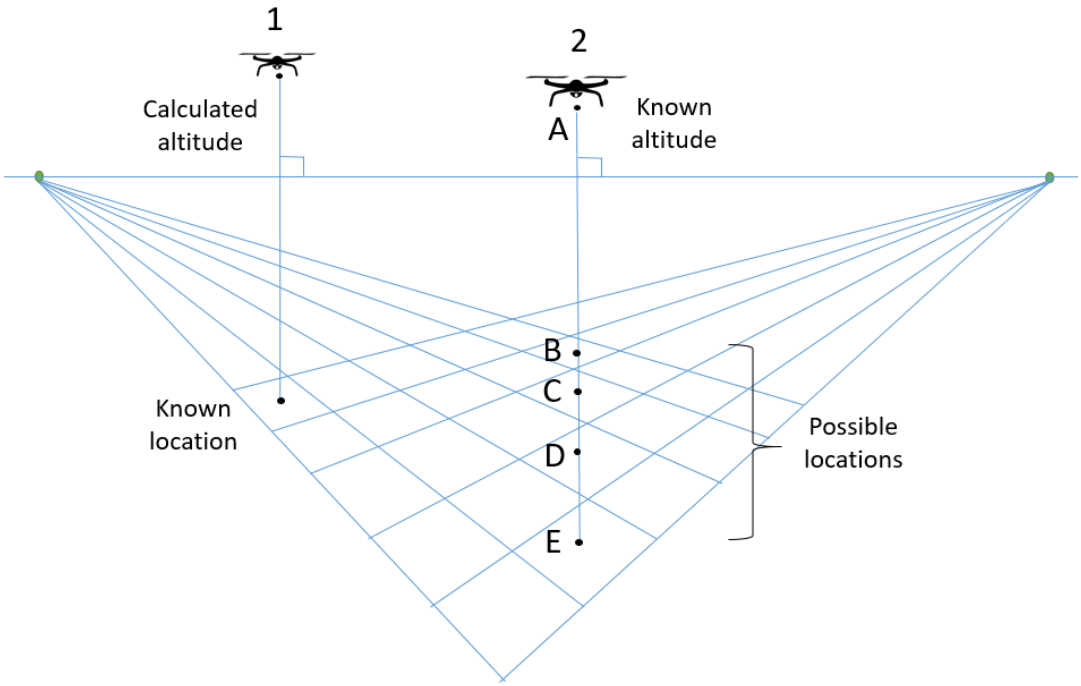

Figure 25. Given either one of object altitude or location, calculate the other.

Location of object above ground. Given the altitude of an object above ground, a vertical line may be drawn perpendicular to the horizon, such that it intersects a column of perspective grid cells. This is the case for object 2 in **Figure 25**. Each of the line lengths, AB, AC, AD, and AE are multiplied by the reference height at their corresponding cell locations and compared to the known altitude. The closest match determines the object location.

Height of object. This calculation is similar to the altitude of an object above ground. Given a bounding box around an object, a line is drawn from the bottom center to the top center of the bounding box and measured. The reference height of the grid cell at the object's location (which may be some distance below if the object is flying) is multiplied by the line length to obtain the object height.

Uneven terrain. The reference height value, say in object-space meters per image-space pixel, is stored at each vertex of the perspective grid. If the terrain of interest is uneven and changes in elevation relative to the initial reference square are known, for instance by a topographic contour map, then the position of the cell vertices can be adjusted up or down for more accurate location of objects. The real-world height deviation at each vertex would be divided by the corresponding reference height value to determine the pixel height adjustment required. This is illustrated in **Figure 26.**

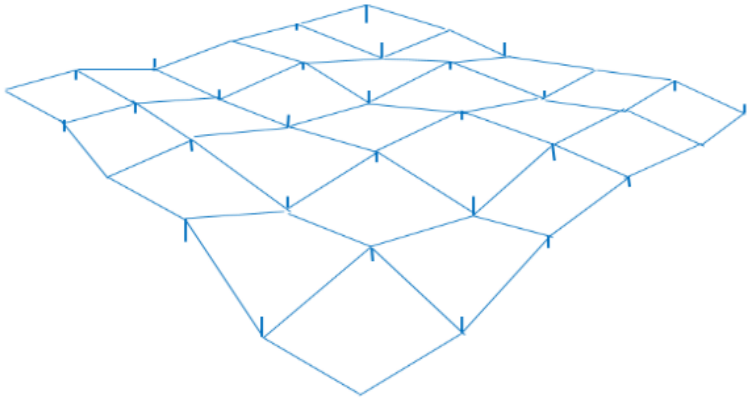

Figure 26. Adjustment of the grid for uneven terrain. Cell vertices are raised or lowered according to their known elevation.

8. Conclusion

In this report, a single passive-sensor system is described for determining the three-dimensional location of objects detected in security camera feeds over critical areas for UAS test flights. Using a single camera positioned above ground and a single reference cell defined on the surface of interest, an algorithm is defined using the cross-ratio to project an imagined grid of square cells from object space onto the image space. This method integrates well with machine learning object detection algorithms to localize objects into three-dimensional space at a specified level of precision. The robustness, simplicity and flexibility of this approach make it suitable for extension to a variety of other use cases.

9. Acknowledgements

The WAHLDO project was funded by NASA Langley ACT2 Digital Transformation under the auspices of Patricia Glaab, Carrie Rhoades, Jill Marlowe and Kevin Rivers. Valuable guidance and customer requirements were provided by the CERTAIN flight range leadership: Lou Glaab, Matt Coldsnow, Ersin Ancel, and Sloan Glover. We are especially grateful for the technical support of our LaRC OCIO Data Science team colleagues: Ed Mclarney, Tony Arviola, Jeff Brandt and Hari Ilangovan. Finally, a shout out to the Georgia Tech Aerospace System Design Lab and to all the Langley interns who worked on the LaRC Park precursor project.

10. References

1. He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). *Mask r-cnn*. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).
2. Physics tutorial: Refraction and the Ray model of light. The Physics Classroom. (n.d.). Retrieved December 1, 2021, from https://www.physicsclassroom.com/class/refrn/Lesson-5/The-Mathematics-of-Lenses.
3. Haseeb, M.A. (2018). *DisNet: A novel method for distance estimation from monocular camera*.
4. Pappus. (1986). Proposition 129. In A. Jones (Ed.), *Pappus of Alexandria, Book 7 of the collection*. essay, Springer.
5. The Matplotlib development team. (2021). Matplotlib.patches.patch¶. matplotlib.patches.Patch - Matplotlib 3.5.0 documentation. Retrieved December 10, 2021, from https://matplotlib.org/stable/api/_as_gen/matplotlib.patches.Patch.html.
6. Velten, A., Willwacher, T., Gupta, O., Veeraraghavan, A., Bawendi, M. G., & Raskar, R. (2012). Recovering three-dimensional shape around a corner using ultrafast time-of-flight imaging. Nature Communications, 3(1). https://doi.org/10.1038/ncomms1747
7. Nedevschi, Sergiu & Vatavu, Andrei & Oniga, Florin & Meinecke, Marc-Michael. (2008). *Forward Collision Detection using a Stereo Vision System*. Proceedings - 2008 IEEE 4th International Conference on Intelligent Computer Communication and Processing, ICCP 2008. 115 - 122. 10.1109/ICCP.2008.4648362.
8. Chen, S., Liu, B., Feng, C., Vallespi-Gonzalez, C., & Wellington, C. (2020*). 3d point cloud processing and learning for autonomous driving*. arXiv preprint arXiv:2003.00601.
9. Abdulla, W. (2017). *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. GitHub repository. Retrieved December 10, 2021, from https://github.com/matterport/Mask_RCNN
10. Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). *Microsoft coco: Common objects in context*. In European conference on computer vision (pp. 740-755). Springer, Cham.
11. Kim, B., Kang, C. M., Kim, J., Lee, S. H., Chung, C. C., & Choi, J. W. (2017, October). *Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network*. In 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC) (pp. 399-404). IEEE.
12. Pamfilos, P. (2021). Crossratio - 2012. CrossRatio.pdf. Retrieved December 6, 2021, from http://users.math.uoc.gr/~pamfilos/eGallery/problems/CrossRatio.pdf.
13. Venugopalan, K. (1964). 115. A proof of the fundamental Cross-Ratio property. The Mathematical Gazette, 48(364), 211-212. doi:10.1017/S0025557200252200

11. Appendix A – Explanation of Cross-Ratio Invariance

The literature contains rigorous proofs of cross-ratio invariance.[12, 13] This section offers an intuitive explanation of why the cross-ratio doesn't change for different collinear points on a projection from a central point V, as shown in **Figure 27**.

The explanation involves a logical argument in four steps:

1. The lengths of the collinear line segments are directly proportional to the area of the triangles each segment forms with point V. Therefore, triangle areas can be used as a proxy for calculating cross-ratios.
2. The areas of the line segment triangles may be rewritten to remove height h and the line segment length as terms, using instead the other two sides of the triangle and their included angle.
3. Writing out the cross-ratio in terms of the new area definitions, all of the triangle sides cancel out, leaving a cross-ratio defined solely by the angles formed with point V.
4. Therefore, since all collinear points share the very same angles with point V, they all must have the same cross-ratio.

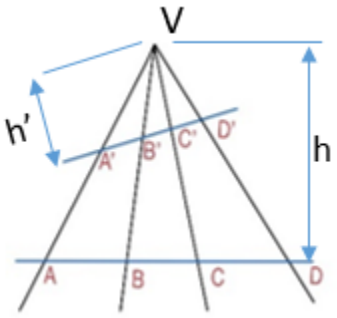

Figure 27. The cross-ratios of collinear points ABCD and A'B'C'D' are invariant.

Here's the detail for each step:

1. **Areas can proxy for line segment lengths**. Referring to **Figure 27**, the area of triangle AVB = ½ h AB. Triangle area is proportional to line segment length. In a ratio with any other line segment, the ½ h terms cancel out:

$$\frac{Area\ \Delta\ AVC}{Area\ \Delta\ AVD} = \frac{\frac{1}{2} h\ \overline{AC}}{\frac{1}{2} h\ \overline{AD}} = \frac{\overline{AC}}{\overline{AD}}$$

2. **Areas can be defined in terms of two sides and their included angle.** Instead of defining the area of Δ AVB as ½ h AB, let's think of AV as the base and x as the height as in **Figure 28**. Now the area is defined as ½ x AV, where x = BV sin ∠AVB.

$$Area\ \Delta\ AVB = ½\ \text{AV BV sin} \angle\text{AVB}$$

Similarly, we can write out the areas of the four components of the cross-ratio:

$$Area\ \Delta\ AVC = ½\ \overline{AV}\ \overline{CV}\ \text{sin} \angle\text{AVC}$$
$$Area\ \Delta\ BVD = ½\ \overline{BV}\ \overline{DV}\ \text{sin} \angle\text{BVD}$$
$$Area\ \Delta\ AVD = ½\ \overline{AV}\ \overline{DV}\ \text{sin} \angle\text{AVD}$$
$$Area\ \Delta\ BVC = ½\ \overline{BV}\ \overline{CV}\ \text{sin} \angle\text{BVC}$$

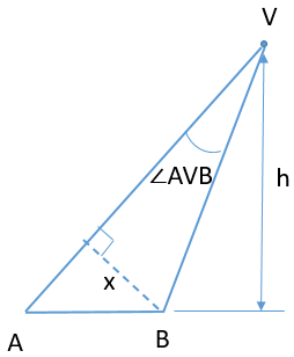

Figure 28. Triangle area in terms of the rays to point V.

3. **Cross-ratio is completely defined by angles at point V.** When the cross-ratio is expressed in terms of triangle area, all the triangle sides cancel:

$$\text{Cross Ratio} = \frac{\overline{AC}\,\overline{BD}}{\overline{AD}\,\overline{BC}} = \frac{Area\ \Delta\ AVC \cdot Area\ \Delta\ BVD}{Area\ \Delta\ AVD \cdot Area\ \Delta\ BVC}$$

$$= \frac{(\tfrac{1}{2}\ \text{AV CV sin}\ \angle\text{AVC}) \cdot (\tfrac{1}{2}\ \text{BV DV sin}\ \angle\text{BVD})}{(\tfrac{1}{2}\ \text{AV DV sin}\ \angle\text{AVD}) \cdot (\tfrac{1}{2}\ \text{BV CV sin}\ \angle\text{BVC})}$$

$$= \frac{(\tfrac{1}{2}\ \tfrac{1}{2}\ \text{AV BV CV DV}) \sin\angle\text{AVC} \cdot \sin\angle\text{BVD}}{(\tfrac{1}{2}\ \tfrac{1}{2}\ \text{AV BV CV DV}) \sin\angle\text{AVD} \cdot \sin\angle\text{BVC}}$$

$$= \frac{\sin\angle\text{AVC} \cdot \sin\angle\text{BVD}}{\sin\angle\text{AVD} \cdot \sin\angle\text{BVC}}$$

4. **Cross-ratio is invariant for all collinear line segments**. The cross-ratio is completely defined by the angles with point V that the collinear line segments subtend. Since all collinear line segments subtend the very same angles, all have the same cross-ratio.

.

12. Appendix B – Cross Ratio with Adjacent Points

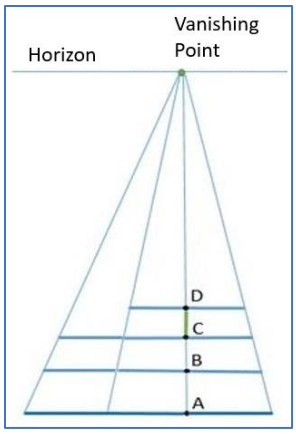

Figure 29. Cross-Ratio line segments in one-point perspective.

Four collinear points A, B, C, and D in image space are depicted in **Figure 29**. Whenever any three of the points are known, a defined cross-ratio value can be used to calculate the unknown fourth point.

Recall the definition of the Cross Ratio:

$$Cross\ Ratio(A,B,C,D) \equiv \frac{\overline{AC}\cdot\overline{BD}}{\overline{AD}\cdot\overline{BC}}$$

In object space, a uniform grid of square cells will have a cross ratio value of:

$$(\overline{AC}\cdot\overline{BD}) / (\overline{AD}\cdot\overline{BC}) = (2\cdot 2) / (3\cdot 1) = 4/3$$

12.1. Exterior Points Toward Vanishing Point

If D is the unknown point, then given $\overline{AB}$ and $\overline{BC}$ and Cross-Ratio = 4/3, we can find $\overline{CD}$:

4/3 = $\overline{AC}\,\overline{BD}$ / $\overline{BC}\,\overline{AD}$

= AC(BC+CD)/BC(AC+CD)

= (AC BC + AC CD) / (AC BC + BC CD)

Cross-multiplying:

3(AC BC + AC CD) = 4(AC BC + BC CD)

3(AC CD) – 4(BC CD) = AC BC

CD(3AC – 4BC) = AC BC

$\overline{CD} = \overline{AC}\ \ \overline{BC} / (3\overline{AC} - 4\overline{BC})$

Calculation Example

Given $\overline{AB} = 1, \overline{BC} = \frac{4}{5}$:

$$\overline{CD} = \frac{\overline{AC}\,\overline{BC}}{3\overline{AC} - 4\overline{BC}}$$

$$= \frac{\left(\frac{9}{5}\right)\left(\frac{4}{5}\right)}{3\left(\frac{9}{5}\right) - 4\left(\frac{4}{5}\right)} = \frac{\frac{36}{25}}{\frac{11}{5}} = \frac{36}{55}$$

12.2. Exterior Points Away from Vanishing Point

If A is the unknown point, then given $\overline{BC}$ and $\overline{CD}$ and Cross-Ratio = 4/3, we can find $\overline{AB}$:

4/3 = $\overline{AC}\,\overline{BD}$ / $\overline{BC}\,\overline{AD}$

= (AB+BC)BD/BC(AB+BD)

= (AB BD + BC BD) / (AB BC + BC BD)

Cross-multiplying:

3(AB BD + BC BD) = 4(AB BC + BC BD)

3(AB BD) – 4(AB BC) = BC BD

AB(3BD – 4BC) = BC BD

$\overline{AB} = \overline{BC}\ \ \overline{BD} / (3\overline{BD} - 4\overline{BC})$

Calculation Example

Given $\overline{BC} = \frac{4}{5}$, $\overline{CD} = \frac{36}{55}$:

$$\overline{AB} = \frac{\overline{BC}\,\overline{BD}}{3\overline{BD} - 4\overline{BC}}$$

$$= \frac{\left(\frac{4}{5}\right)\left(\frac{16}{11}\right)}{3\left(\frac{16}{11}\right) - 4\left(\frac{4}{5}\right)} = \frac{\frac{64}{55}}{\frac{64}{55}} = 1$$

12.3. Interior Point

If C is the unknown point, then given $\overline{AB}$ and $\overline{BD}$ and Cross-Ratio = 4/3, we can find $\overline{BC}$:

4/3 = $\overline{AC}\,\overline{BD}$ / $\overline{BC}\,\overline{AD}$

= (AB+BC)(BD)/BC(AB+BD)

= (AB BD + BC BD) / (AB BC + BC BD)

Cross-multiplying:

3(AB BD + BC BD) = 4(AB BC + BC BD)

3(AB BD) = 4 AB BC + BC BD

3(AB BD) = BC (4AB + BD)

$\overline{BC}$ = 3 $\overline{AB}\,\overline{BD}$ / (4 $\overline{AB}$ + $\overline{BD}$)

Calculation Example

Given $\overline{AB} = 1$, $\overline{BD} = \frac{16}{11}$:

$$\overline{BC} = \frac{3\,\overline{AB}\,\overline{BD}}{4\overline{AB} + \overline{BD}}$$

$$= \frac{3(1)\left(\frac{16}{11}\right)}{4(1) + \left(\frac{16}{11}\right)} = \frac{\frac{48}{11}}{\frac{60}{11}} = \frac{4}{5}$$

If B is the unknown point, then given $\overline{AC}$ and $\overline{CD}$ and Cross-Ratio = 4/3, we can find $\overline{BC}$:

4/3 = $\overline{AC}\,\overline{BD}$ / $\overline{BC}\,\overline{AD}$

= AC(BC+CD)/BC(AC+CD)

= (AC BC + AC DC) / (AC BC + BC CD)

Cross-multiplying:

3(AC BC + AC CD) = 4(AC BC + BC CD)

3(AC CD) = AC BC + 4(BC CD)

3(AC CD) = BC (AC + 4 CD)

$\overline{BC}$ = 3 $\overline{AC}\,\overline{CD}$ / ($\overline{AC}$ + 4 $\overline{CD}$)

Calculation Example

Given $\overline{AC} = \frac{9}{5}$, $\overline{CD} = \frac{36}{55}$:

$$\overline{BC} = \frac{3\,\overline{AC}\,\overline{CD}}{\overline{AC} + 4\overline{CD}}$$

$$= \frac{3\left(\frac{9}{5}\right)\left(\frac{36}{55}\right)}{\left(\frac{9}{5}\right)+4\left(\frac{36}{55}\right)} = \frac{\frac{972}{5(55)}}{\frac{243}{55}} = \frac{\frac{4(243)}{5(55)}}{\frac{243}{55}} = \frac{4}{5}$$

13. Appendix C – Cross Ratio with Vanishing Points

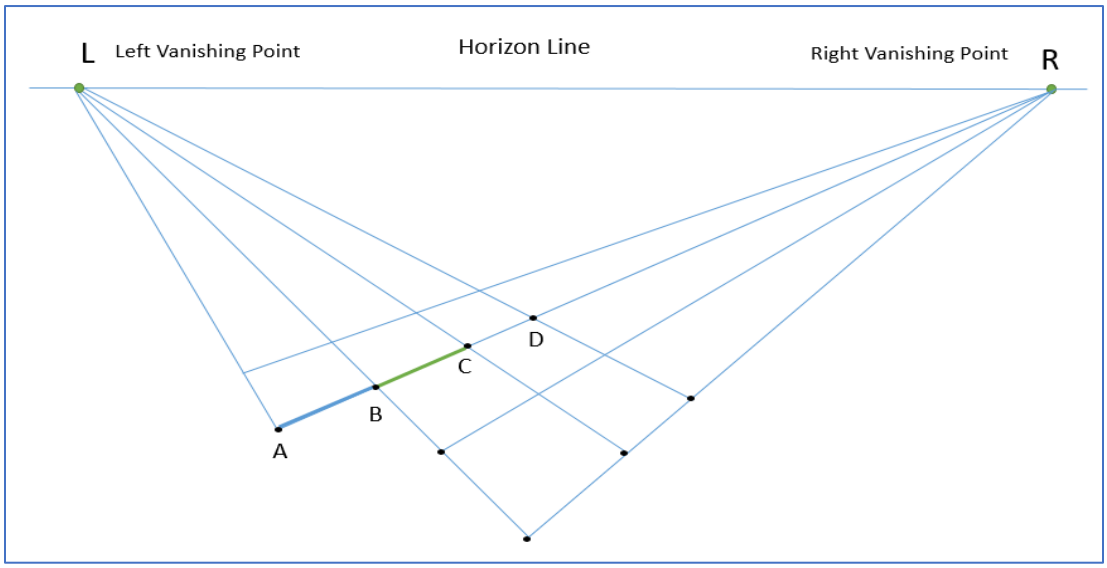

Figure 30. Cross-Ratio line segments in two-point perspective.

Three collinear points A, B, C in image space are depicted in **Figure 30**. Whenever any two of the points are known, a defined collinear vanishing point R and known cross-ratio value can be used to calculate the third unknown point.

Recall the short cut version of the Cross-Ratio when a vanishing point is employed:

$$Cross\ Ratio(A,B,C,V) = \frac{\overline{AC}\cdot\overline{BV}}{\overline{AV}\cdot\overline{BC}} = \lim_{x\to\infty}\left(\frac{\overline{AC}\cdot x}{\overline{BC}\cdot x}\right) \to \lim_{x\to\infty}\frac{\frac{d}{dx}[\overline{AC}\cdot x]}{\frac{d}{dx}[\overline{BC}\cdot x]} = \frac{\overline{AC}}{\overline{BC}},\ by\ L'Hôpital's\ rule$$

In object space, a uniform grid of square cells will have a cross ratio value of:

$\overline{AC}\ /\ \overline{BC}\ =\ 2\ /\ 1\ =\ \mathbf{2}$

13.1. Exterior Points Toward Vanishing Point

If C is the unknown point, then given $\overline{AB}$ and $\overline{BR}$ and Cross-Ratio = 2, we can find $\overline{BC}$:

2 = $\overline{AC}\,\overline{BR}$ / $\overline{BC}\,\overline{AR}$

= (AB+BC)BR/BC(AB+BR)

= (AB BR + BC BR) / (AB BC + BC BR)

Cross-multiplying:

AB BR + BC BR = 2(AB BC + BC BR)

AB BR = 2(AB BC) + BC BR

AB BR = BC(2 AB + BR)

$BC = \overline{AB}\,\overline{BR}\ /\ (2\overline{AB} + \overline{BR})$

Calculation Example

Given $\overline{AB} = 1, \overline{BR} = 8$:

$$\overline{BC} = \frac{\overline{AB}\,\overline{BR}}{2\overline{AB} + \overline{BR}} = \frac{(1)(8)}{2(1) + (8)} = \frac{\mathbf{4}}{\mathbf{5}}$$

The same approach may be repeated down the line, now using $\overline{BC}$ and $\overline{CR}$ to find $\overline{CD}$:

2 = $\overline{BD}\ \overline{CR}$ / $\overline{CD}\ \overline{BR}$

= (BC+CD)CR/CD(BC+CR)

= (BC CR + CD CR) / (BC CD + CD CR)

Cross-multiplying:

BC CR + CD CR = 2(BC CD + CD CR)

BC CR = 2(BC CD) + CD CR

BC CR = CD(2 BC + CR)

$\overline{CD} = \overline{BC}\ \overline{CR} / (2\overline{BC} + \overline{CR})$

Calculation Example

Given $\overline{BC} = \frac{4}{5}$, $\overline{CR} = \frac{36}{5}$:

$$\overline{CD} = \frac{\overline{BC}\ \overline{CR}}{2\overline{BC} + \overline{CR}}$$

$$= \frac{\left(\frac{4}{5}\right)\left(\frac{36}{5}\right)}{2\left(\frac{4}{5}\right) + \left(\frac{36}{5}\right)} = \frac{\frac{144}{25}}{\frac{44}{5}} = \frac{144}{220} = \mathbf{\frac{36}{55}}$$

13.2. Exterior Points Away From Vanishing Point

If A is the unknown point, then given $\overline{BC}$ and $\overline{BR}$ and Cross-Ratio = 2, we can find $\overline{AB}$:

2 = $\overline{AC}\ \overline{BR}$ / $\overline{BC}\ \overline{AR}$

= (AB+BC)BR/BC(AB+BR)

= (AB BR + BC BR) / (AB BC + BC BR)

Cross-multiplying:

AB BR + BC BR = 2(AB BC + BC BR)

AB BR – 2(AB BC) = BC BR

AB (BR – 2 BC) = BC BR

$\overline{AB} = \overline{BC}\ \overline{BR} / (\overline{BR} - 2\ \overline{BC})$

Calculation Example

Given $\overline{BC} = \frac{4}{5}$, $\overline{BR} = 8$:

$$\overline{AB} = \frac{\overline{BC}\ \overline{BR}}{\overline{BR} - 2\ \overline{BC}}$$

$$= \frac{\left(\frac{4}{5}\right)\ 8}{8 - 2\left(\frac{4}{5}\right)} = \frac{\frac{32}{5}}{\frac{32}{5}} = \mathbf{1}$$

13.3. Interior Point

If B is the unknown point, then given $\overline{AC}$, $\overline{AR}$ and $\overline{BR}$ and Cross-Ratio = 2, we can find $\overline{BC}$:

2 = $\overline{AC}\ \overline{BR}$ / $\overline{BC}\ \overline{AR}$

= (AC)(BC + CR)/(BC AR)

= (AC BC + AC CR) / (BC AR)

Cross-multiplying:

AC BC + AC CR = 2(BC AR)

AC CR = 2(BC AR) – AC BC

AC CR = BC(2 AR – AC)

$\overline{BC} = \overline{AC}\ \overline{CR}\ /\ (2\overline{AR} - \overline{AC})$

Calculation Example

Given $\overline{AC} = \frac{9}{5},\ \overline{AR} = 9, \overline{CR} = \frac{36}{5}$

$$\overline{BC} = \frac{\overline{AC}\ \overline{CR}}{2\overline{AR} - \overline{AC}}$$

$$= \frac{\left(\frac{9}{5}\right)\left(\frac{36}{5}\right)}{2(9) - \left(\frac{9}{5}\right)} = \frac{\frac{324}{25}}{\frac{81}{5}} = \frac{\frac{81(4)}{5(5)}}{\frac{81}{5}} = \mathbf{\frac{4}{5}}$$

14. Appendix D – Calculation of Vanishing Points

The proposed algorithm uses vanishing points to simplify the cross-ratio equations and reduce the minimum initial information needed to that of a single reference grid cell. Since the simplified cross-ratio formula uses vanishing points as the fourth collinear point, the precise location of the vanishing point in image space must be determined.

The equations below in **Figure 31** locate the vanishing points by extending the opposite sides of the reference grid cell until the extensions converge at the horizon. These equations calculate the location of the vanishing points in terms of (X,Y) floating point pixel values. In the image space of the camera, (X,Y) pixel coordinates are the most convenient method for defining perspective grid cell vertices, and matching them to detected object bounding boxes.

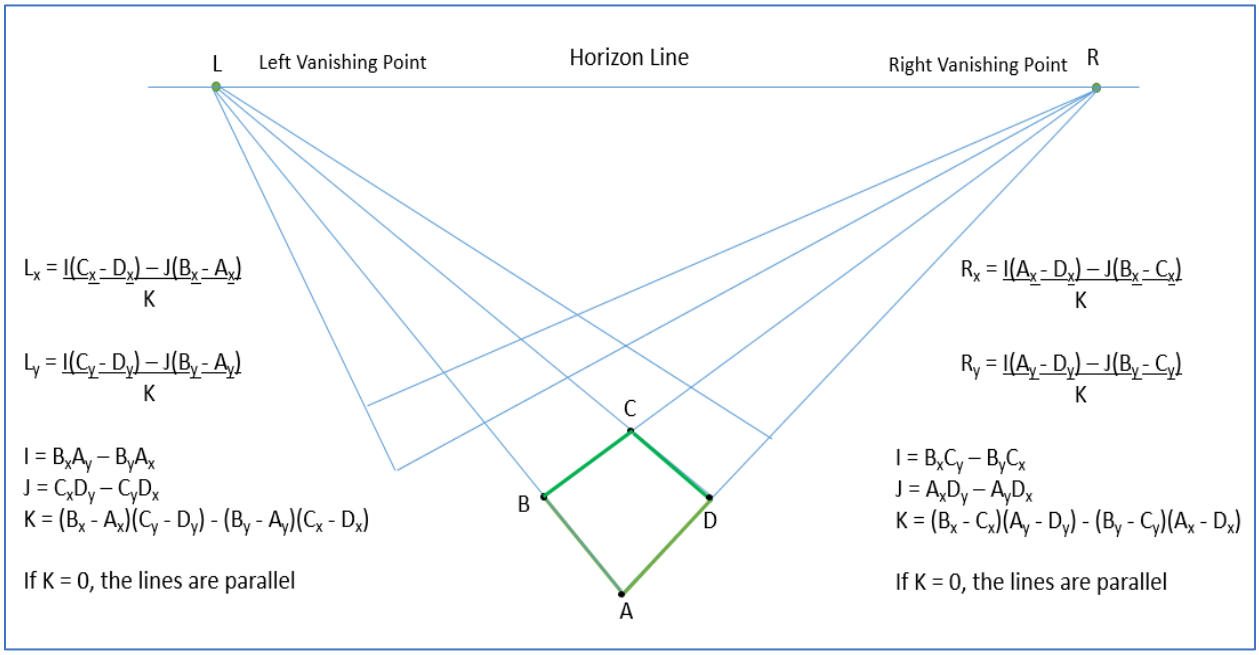

Figure 31. Using the reference cell corners to determine pixel coordinates of the vanishing points.

15. Appendix E – Calculation of Height

Suppose instead of constructing a flat grid of cells, the grid is constructed from cubes. The cubes would be contiguous and equally tall in object space, although not necessarily of a height equal to the length of the cell sides. In image space the cube heights would diminish towards zero as the cubes approach vanishing points on the horizon, just as the grid cells do.

If the heights of the cubes were known at each cell location on the grid, then the height of an object detected in a given cell could be estimated. Also, knowing equivalent object-space height in terms of image-space pixels at the cell vertices would allow for the vertices to be raised or lowered according to changes in terrain elevation as depicted in **Figure 26**. Note that if the camera is tilted, the concepts of height and "raise/lower" refer to travel on a line perpendicular to the horizon, which is established as the line connecting the left and right vanishing points, which are themselves calculated in **Appendix D.**

Referring to **Figure 32**, a method is now described to determine object-space height in perspective at each vertex of the grid cells. In addition to the minimum four points of the reference square, a pole of known height is raised vertically in object space at the most foreground point A. The object-space height is then known for line segment $\overline{AE}$, but this is only true at point A. For other points B and D closer to the vanishing points, the image-space representation of the same object-space height will diminish, resulting in shorter line segments $\overline{BG}$ and $\overline{DF}$**.**

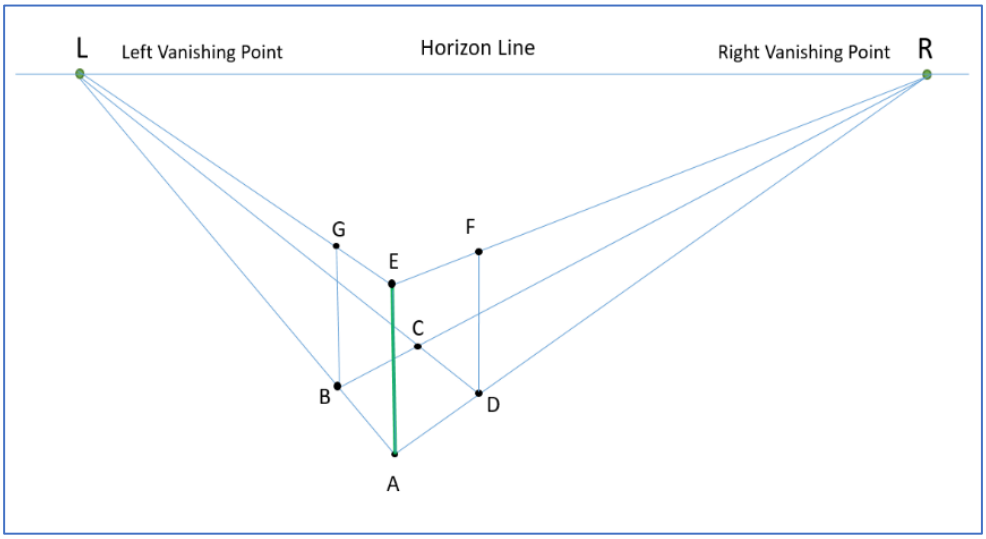

Figure 32. Calculation of image-space height at all cell vertices of the

To calculate $\overline{BG}$ and $\overline{DF}$ it is enough to observe that $\overline{AE}$, $\overline{BG}$, and $\overline{DF}$ are all perpendicular to the horizon and therefore parallel to each other. Thus, the triangles △ARE and △DRF have identical corresponding angles and the triangles are similar. Therefore, the lengths of the triangle sides are proportional, namely: $\overline{AE}$ / $\overline{AR}$ = $\overline{DF}$ / $\overline{DR}$. Given the initial points A, D, and E and the vanishing point R determined in **Appendix D**, the reference height at point D is:

$$\overline{DF} = \overline{AE} \cdot \overline{DR} / \overline{AR}$$

Similarly, the reference height at point B is:

$$\overline{BG} = \overline{AE} \cdot \overline{BL} / \overline{AL}$$

Following the methods of **Appendix C,** as each new grid cell point is defined by the previous, the distance of the new point to its collinear vanishing point becomes known. The new height at each point may then be calculated using the equations above. The height information, in units of object-space distance per image-space pixels, is stored in the database along with each cell vertex definition. The information is then available for detecting object height or adjusting the perspective grid for uneven terrain, as described above.