Adaptive Synthetic Vision

Simon J. Julier[a], Dennis Brown[b], Mark A. Livingston[b], and Justin Thomas[c]

[a]ITT AES, Alexandria, VA;
[b]Naval Research Laboratory, Washington, DC;
[c]Reallaer LLC, Saint Leonard, MD

ABSTRACT

Through their ability to safely collect video and imagery from remote and potentially dangerous locations, UAVs have already transformed the battlespace. The effectiveness of this information can be greatly enhanced through synthetic vision. Given knowledge of the extrinsic and intrinsic parameters of the camera, synthetic vision superimposes spatially-registered computer graphics over the video feed from the UAV. This technique can be used to show many types of data such as landmarks, air corridors, and the locations of friendly and enemy forces. However, the effectiveness of a synthetic vision system strongly depends on the accuracy of the registration — if the graphics are poorly aligned with the real world they can be confusing, annoying, and even misleading.

In this paper, we describe an adaptive approach to synthetic vision that modifies the way in which information is displayed depending upon the registration error. We describe an integrated software architecture that has two main components. The first component automatically calculates registration error based on information about the uncertainty in the camera parameters. The second component uses this information to modify, aggregate, and label annotations to make their interpretation as clear as possible. We demonstrate the use of this approach on some sample datasets.

Keywords: synthetic vision, augmented reality, registration error, display management

1. INTRODUCTION

UAVs are becoming increasingly important in battlefield operations. UAVs provide a safe means to collect video and imagery from remote and potentially dangerous locations. However, if this information is to be of use, the operators must have situation awareness (SA) of the UAV itself. Drury identifies nine dimensions for this SA.[1] These include: current 3D spatial relationships between the UAV and other objects, predicted future 3D spatial relationships between the UAV and other objects, and the current status of the UAV with respect to its mission. Although this information can be provided by providing electronic maps and other displays to supplement video and other sensor feeds,[2] *synthetic vision* (SV) offers a compelling means of displaying this information by directly overlaying it onto the video stream. Using data about the UAV's position and attitude, together with a model of the environment, computer-generated graphics can be created and overlaid on video imagery in real-time. The types of information that can be shown include the names of buildings, the location of friendly and enemy troops, and air access corridors. Because this information is overlaid directly on the display, the human operator does not have to perform this cognitively stressful and error-prone task themselves.

However, the effectiveness of such a system is undermined by the presence of *registration errors*. Registration errors arise when the computer generated graphics do not align with the real world*. There can be many causes including tracking errors (the true position and attitude of the camera is not the same as the one assumed by the SV system), calibration errors (the intrinsic camera parameters such as focal length and lense distortion effects

Further author information: (Send correspondence to Simon Julier)
S Julier: E-mail: simon.julier@nrl.navy.mil, Telephone: +1-202-767-0275
Dennis Brown: E-mail: dennis.g.brown@nrl.navy.mil, Telephone: +1-202-404-7334
Mark A. Livingston: E-mail: mark.livingston@nrl.navy.mil, Telephone: +1-202-767-0380
Justin Thomas: Email: thomas@reallaer.com

*Note that this is a different concept from the one which arises from image mosaicking where it refers to the misalignment between a succession of images.

are known imperfectly), and modeling errors (the database of the environment is incorrect). These errors can produce displays which are annoying, distracting, and, at worst, downright misleading. Although such errors are very important, very few researchers in the synthetic vision community have considered their impact.

The same difficulties plague a research field known as *augmented reality* (AR).[3] AR superimposes computer generated graphics on a user's view of the real world. It can be used to provide battlefield situation awareness, guide surgeons, visualize site plans on buildings, and even provide virtual signs for advertising purposes. Although many applications explicitly (or implicitly) assume perfect registration, these conditions cannot be met in practice. Therefore, a number of authors have recently begun to develop systems which dynamically adapt the display to account, in real-time for estimated registration errors.[4,5]

Although the effects of registration errors can be mitigated for high altitude, high-value UAVs (where high quality sensors can be used and 2D image processing algorithms can be deployed), many UAVs fly at low altitudes and so registration becomes a full 3D alignment (and thus significantly harder) problem. Furthermore, lower altitude UAVs tend to be smaller and thus use lower quality sensors. Although researchers are attempting to develop accurate tracking algorithms, research in the AR community has shown that, even in highly instrumented environments, perfect registration is almost impossible to achieve. Therefore, we believe that any practical system must provide a user interface that is robust to the effects of these errors.

In this paper we describe a prototype adaptive synthetic vision technique for lower-altitude, lower-value UAVs. The prototype, implemented using the OSGAR toolkit,[6] uses statistical information about tracking errors to calculate the statistics of the registration errors. In turn, this information is used to update and change the display dynamically at run time. We believe that our method of addressing registration error will create more informative displays, especially in the case that the error is very large, or that the objects of interest are small objects in large clusters of similar objects, where any error can cause the operator to choose the wrong real-world object.

The structure of this paper is as follows. Section 2 describes the problem statement. Section 3 describes the core abstraction in our adaptive system, the *scene graph*, and how errors can be propagated through this graph. The adaptation system is described in Section 4. Conclusions are drawn in Section 5.

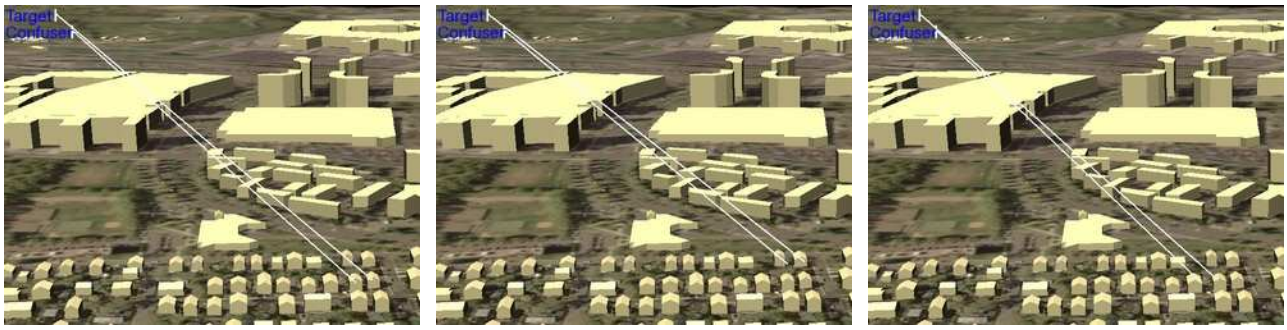
2. PROBLEM STATEMENT

Consider the following situation: a UAV flies over an urban environment and uses a camera to capture images of the scene. This information is fed back to an operator who attempts to interpret this information to provide SA to other users in the battlespace. However, this can only occur if the operator has SA of the state of the UAV itself. Drury identifies nine releveant dimensions.[1] These include the current 3D spatial relationships between the UAV and other objects in the enviroment, the predicted future 3D spatial relationships between the UAV and other objects, and the UAV's status with respect to its mission.

In addition to understanding the UAV's status, the operator must be able to interpret the video and imagery data. This task can potentially be extremely difficult: the UAV is potentially viewing a very small part of the environment from a very different point of view than the operator. To assist the operator graphical cues are inserted into the video display to assist the operator. The use of such a system is illustrated in Figure 1(a): the environment consists of a large number of similar-looking buildings, and the synthetic vision system provides the cues to show the operator identifying information to disambiguate the identity of each building[†].

Several commercial implementations of this technology are available, including systems built by Calisto Data AB,[8] Rapid Imaging Software, Inc.,[9] and Nav3D Corporation.[10] These implementations overlay computer graphics on the UAV's video feed, using GPS and inertial systems to determine the UAV's attitude and position the graphics. Information comes directly from various sensors on the aircraft, from databases such as geographic information systems (GIS), from terrain models, and from other sources.

[†]Throughout this paper, we use computer generated models of the environment to illustrate the concepts and ideas. Although the focus of our work is synthetic vision, many of the arguments can be applied to *augmented virtuality* (AV) as well.[7] AV allows users to navigate (pan, tilt, and zoom) through a virtual model of the world. They can thus see the world from different perspectives than those that are actually available. Furthermore, different users can see different types of data.



(a) Correctly aligned annotations. (b) Misalignment due to error in x-axis rotation. (c) Misalignment due to error in y-axis rotation.

Figure 1. The effects of registration errors. Building names are indicated by the label at the top left of the screen and a call out line which points to the building. A heading error of 1° causes the callout lines to point to the wrong buildings.

Beyond these straightforward efforts, researchers are finding innovative ways to apply SV to UAVs. Goktogan and Sukkarieh[11] outline a system that might be used to help control teams of UAVs, including showing the locations of other team members and of targets in the augmented display. Quigley et al.[12] show an icon representing the UAV's future desired location and attitude on the augmented display so that the operator can more easily maneuver into that position. Bath and Paxman[13] use computer vision and markers in the environment to register the computer graphics on their augmented display.

However, the subject of registration error has not been thoroughly investigated in the community. Registration error occurs when the computer-generated graphics do not align with the environment. There are many causes including incorrect tracking of the UAV, sensor calibration errors, alignment errors, and even latency. Although many of these difficulties are minimized in 2D image registration problems (the UAV flies at a high enough altitude that the scene is essentially 2D and the problem becomes one of aligning a 2D image in a 2D view), they are not entirely mitigated — the effects of cloud cover, for example, could generate multiple uncertain solutions. Furthermore, when UAVs fly at lower altitudes, the 3D structure of the scene must be taken into account. These issues are illustrated in Figures 1(b) and 1(c): errors of 1° in pitch or heading cause the labels to be placed on the incorrect buildings[‡]. An operator viewing this scene would have no indication that the display is misleading.

Calhoun et al.[14] suggest several ways to combat registration error. One way is by blending, either by adjusting the transparency of the overlay, or in the case of showing augmentations that resemble real objects on the display, blending the edges of the graphical version of the object. Another way they suggest combating delay-based registration error (delay between the position update of the UAV and the screen update) is accurately predicting the UAV's path. Finally, they want the operator to be able to recalibrate the system at will.

Many of these difficulties have been encounted in computer graphics in the context of *augmented reality* (AR).[3] AR superimposes computer graphics on a user's view of the real world and is often compared to a heads up display on an aircraft. However, many AR applications are often directed towards individuals such as surgeons, technicians or dismounted warfighters. As a result, AR systems often suffer from substantial registration errors. A number of authors have recently begun to develop systems which dynamically adapt the display to account, in real-time for estimated registration errors.[4,5] In particular, a prototype system known as OSGAR has been developed to create adaptive display algorithms based on registration error. This system dynamically calculates the impact of various error sources (including errors in camera intrinsic parameters, camera extrinsic parameters and errors in models of the environment) and provides mechanisms for developing displays that automatically reconfigure themselves depending on the effects of these errors.

Given the strong relationship between synthetic vision and augmented reality, we have begun to explore how OSGAR can be applied to a synthetic vision system. To understand how OSGAR operates, it is first necessary

[‡]These errors are comparable to those specified for INS for UAVs. Crossbow's NV420, for example, is specified to have an error of ±0.5° rms for roll and pitch and ±3.0° for heading.

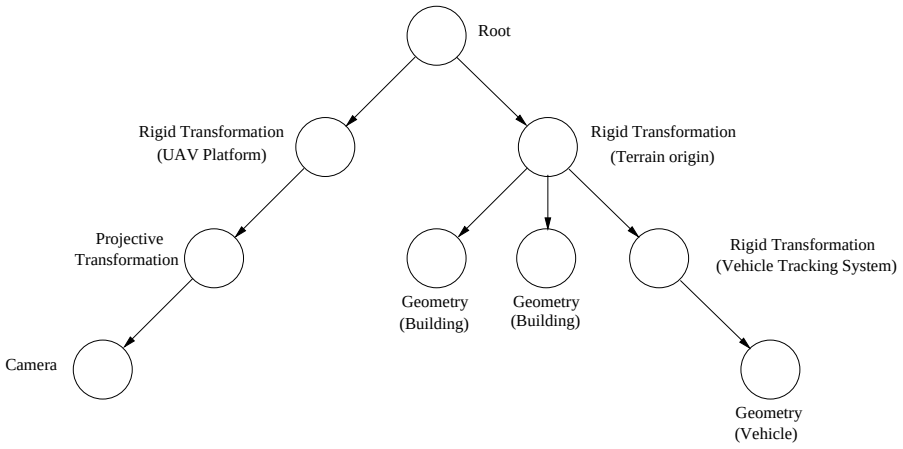


Figure 2. A scene graph representation of the environment. This is a subset of the full graph and only contains nodes related to the physical configuration of the environment.

to understand the core abstraction used in OSGAR — the scenegraph — and how errors propagate through it.

3. SCENE GRAPHS AND ERROR PROPAGATION

Our adaptive synthetic vision system exploits a *scene graph* to quantify the registration error.

3.1. Scene Graph Representation

A scene graph is directed acyclic graph that encodes the entire state of an environment for a 3D rendering system. Some nodes are associated purely with graphics and include lighting, transparency, and text labels. Other nodes have a direct correspondence with the physical configuration of the real environment and encode aspects such as geometry in the environment and the transformations between objects.

A subset of a graph, illustrating geometry, is shown in Figure 2 which shows the simplified view of a UAV flying over a terrain with two buildings and a tracked vehicle. The root node defines the base of the tree. The left-hand branch defines the state of the UAV, the right-hand branch the environment. On the left, there is a rigid transformation (to specify the attitude of the UAV in the environment). This is followed by a projective transformation. Together, these two are used to specify the state of the camera used to observe the scene. On the right, a rigid transformation specifies the coordinate frame within which the terrain is specified. The static environment is attached to this coordinate frame. The tracked vehicle position is specified relative to this frame.

In the scene graph, the transformations between nodes are not guaranteed to be rigid transforms. Instead, they are arbitrary 4×4 matrices. Specifically, let $\mathbf{M}_i^j$ be the relative transformation from node i to node j ,

$$\mathbf{M}_i^j = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix}.$$

Each element can take arbitrary values. To parameterize such a general matrix, a number of authors have developed methods to decompose an arbitrary matrix into a set of primitive operations including rotation, translation, and scale.[15,16] However, these decompositions are constructed by applying potentially expensive nonlinear operations (such as singular value decomposition).

$\mathbf{M}_r^n$ is the cumulative transformation matrix from the root node $\mathbf{R}$ to an arbitrary node $\mathbf{N}$. This transformation is given by:

$$\mathbf{M}_r^n = \prod_{\forall i \in P} \mathbf{M}_{i-1}^i, \tag{1}$$

where p is the path from the root node $\mathbf{R}$ to an arbitrary node $\mathbf{N}$ and i are nodes in this path.

3.2. Error Propagation Through the Scene Graph

The scene graph offers a direct means by which uncertainty can be injected into the transformations in the scene. Uncertainty is represented by a covariance matrix in the transformation nodes in the scene graph. Any transformation matrix is considered to be the mean of a probability distribution function (PDF) for that transformation, instead of just a single discrete transformation. If no uncertainty information is specified for a transformation, it is assumed to be exact. The mean of the PDF (i.e., the original transformation) is used for culling, rendering, and so on, as before.

Because the graph can be extremely large, a significant number of these decomposition operations might be performed leading to significant computational costs. Therefore, to simplify the implementation all errors are expressed directly in terms of the elements of the transformation matrix. In other words, the uncertainty is a 16-dimensional state that corresponds to each element in the transformation matrix. This can be seen as a straightforward generalization of the approach taken by the Matrix Kalman Filter.[17, 18]

The transformation at each node is specified by the estimated transformation $\hat{\mathbf{M}}_i^j$. This can be represented as a 16×1 vector by the vec operator,

$$\hat{\mathbf{m}}_i^j = \text{vec}\left(\hat{\mathbf{M}}_i^j\right) = \begin{pmatrix} m_{11} \\ m_{12} \\ \vdots \\ m_{44} \end{pmatrix}$$

The covariance matrix for $\hat{\mathbf{m}}_i^j$ is the 16×16 covariance matrix $\mathbf{P}_i^j$. The error introduced at a node is assumed to be an additive matrix,

$$\hat{\mathbf{M}}_i^j = \mathbf{M}_i^j + \delta\mathbf{M}_i^j.$$

where $\delta\mathbf{M}_i^j$ has the properties

$$\begin{aligned} \text{E}\left[\text{vec}\left(\delta\mathbf{M}_i^j\right)\right] &= \mathbf{0} \\ \text{E}\left[\text{vec}\left(\delta\mathbf{M}_i^j\right)\text{vec}\left(\delta\mathbf{M}_i^j\right)^T\right] &= \mathbf{Q}_i^j. \end{aligned}$$

Therefore, the error in the transformation from the root node r to a node i is given by the recursion

$$\begin{aligned} \mathbf{M}_r^i &= \mathbf{M}_{i-1}^i\mathbf{M}_r^{i-1} \\ &= \left(\hat{\mathbf{M}}_{i-1}^i + \delta\mathbf{M}_{i-1}^i\right)\left(\hat{\mathbf{M}}_r^{i-1} + \delta\mathbf{M}_r^{i-1}\right) \\ &= \hat{\mathbf{M}}_{i-1}^i\hat{\mathbf{M}}_r^{i-1} + \hat{\mathbf{M}}_{i-1}^i\delta\mathbf{M}_r^{i-1} + \delta\mathbf{M}_{i-1}^i\hat{\mathbf{M}}_r^{i-1} + \delta\mathbf{M}_{i-1}^i\delta\mathbf{M}_r^{i-1} \end{aligned}$$

Neglecting second order error terms, the mean and error terms propagate according to the equations

$$\begin{aligned} \hat{\mathbf{M}}_r^i &= \hat{\mathbf{M}}_{i-1}^i\hat{\mathbf{M}}_r^{i-1} \\ \delta\mathbf{M}_r^i &= \hat{\mathbf{M}}_{i-1}^i\delta\mathbf{M}_r^{i-1} + \delta\mathbf{M}_{i-1}^i\hat{\mathbf{M}}_r^{i-1} \end{aligned}$$

Transforming these expressions to recursive propagations for $\hat{\mathbf{m}}_r^i$, the covariance matrix can be calculated.

Although this representation is computationally expensive (as the transformation is represented by a 16-dimensional state vector), it is an extremely general representation that can be used to represent the effect of *any* transformation. Furthermore, the main propagation equations are linear. Nonlinearities caused by, for example, uncertainty in rotation, appear in the calculation of $\mathbf{Q}_i^j$. Specifically, we assume that the relative transformation can be written as the nonlinear function

$$\text{vec}\left(\hat{\mathbf{M}}_i^j\right) = \mathbf{f}_{ij}\left[\theta_i^j\right]$$

where θ_i^j are the parameters expressing the transformation. For example, if $\hat{\mathbf{M}}_i^j$ is a rigid transformation, θ_i^j would quantify the relative translation and rotation. If $\hat{\mathbf{M}}_i^j$ specifies a perspective projection, θ_i^j would specify the field of view, the near and far clipping planes, and the aspect ratio.

Irrespective of the particular form, it is assumed that θ_i^j is a random variable with mean $\hat{\theta}_i^j$ and covariance $\mathbf{P}_{\theta_i^j}$. Let $\nabla_{\theta_i^j}\mathbf{f}$ be the Jacobian of $\mathbf{f}_{ij}[\cdot]$ evaluated at $\theta = \hat{\theta}_i^j$. Taking outer products and expectations yields

$$\mathbf{Q}_i^j = \nabla_{\theta_i^j}\mathbf{f}\mathbf{P}_{\theta_i^j}\nabla_{\theta_i^j}\mathbf{f}^T.$$

3.3. Error Propagation to the Camera

The previous subsection described how errors can be propagated down the scene graph. These errors are view independent; the error in the position of the vehicle in Figure 2, for example, does not depend on the location of the UAV. However, the effects of these errors depend on the viewpoint of the UAV. For example, the further the camera is from the target, the smaller the effect of position errors. The effects of orientation errors do not, however, diminish with distance. Therefore, it is necessary to project the view independent error model into the frame of the UAV.

Consider the problem of determining the pixel coordinate $\mathbf{y}$ of a point $\mathbf{p}$ rigidly attached to node i. The projection can be broken into two steps: first, transform from world coordinates to the coordinate frame fixed to the camera (with coordinates $\mathbf{p}'$), and second apply the perspective transformation to project the point to the view plane.

The transformation from $\mathbf{p}$ to $\mathbf{p}'$ is governed by the (homogeneous) transformation matrix $\mathbf{M}_i^c$,

$$\mathbf{p}' = \mathbf{M}_i^c\mathbf{p}.$$

$\mathbf{M}_i^c$ is the composite transformation from node c to node i. In Figure 2, for example, it includes the transformation from the camera node to the root (including the terms from the UAV's attitude sensors), and then from the root down to the leaf with the geometry (including tracks from remote tracking systems).

$\mathbf{y}$ is given by applying a perspective transformation to $\mathbf{p}'$. The projection matrix, $\mathbf{P}$, is given by:

$$\mathbf{P} = \begin{bmatrix} \frac{f}{a} & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & \frac{z_{near}+z_{far}}{z_{near}-z_{far}} & \frac{2z_{near}z_{far}}{z_{near}-z_{far}} \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{2}$$

where $f = \cot[\frac{\text{fov}}{2}]$, a is the aspect ratio and z_{near} and z_{far} are the near and far clipping distances.

Therefore,

$$\mathbf{y}' = \mathbf{P}\mathbf{p}'. \tag{3}$$

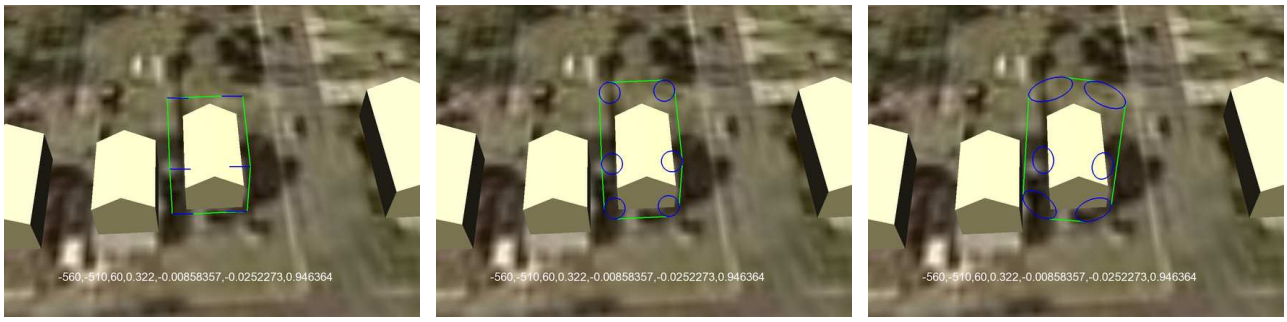
and the pixel coordinate on the viewport is

$$\mathbf{y} = \mathbf{y}_0 + \frac{1}{2y_4'}\begin{bmatrix} w(y_1' + y_4') \\ h(y_2' + y_4') \end{bmatrix} \tag{4}$$

where $\mathbf{y}_0$ is the top left corner of the viewport, w and h are the width and height of the viewport, and y_i' is the ith coefficient in $\mathbf{y}'$.

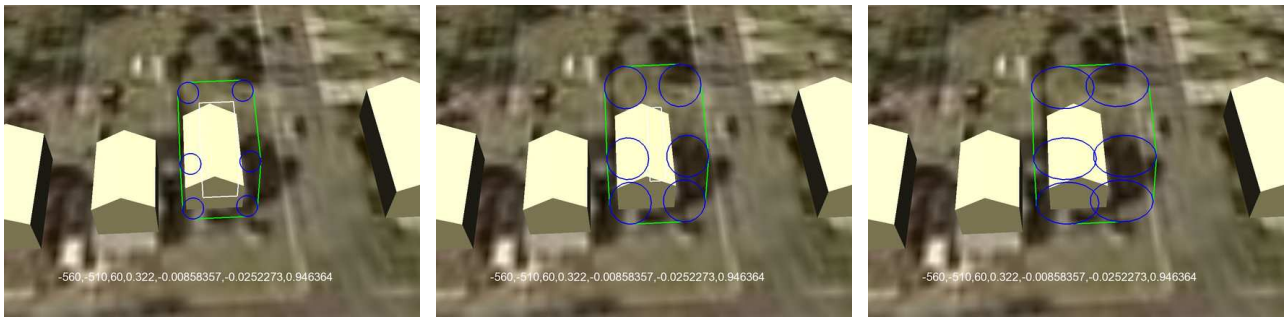
The scene graph applies the error propagation calculations to each vertex of a target object. This process yields a set of ellipses in the two dimensional viewport plane that define the geometry of the target object accounting for registration errors. An outer convex hull is calculated as the shape that contains all error ellipses. An inner convex hull is also calculated using the space between the error propagated ellipses, if any exists. The convex hulls and other similar measures form the basic primitives which can be used to develop the adaptive synthetic vision system.

The framework described in this section provides a rigorous mechanism to quantify the registration errors that arise from tracking and other kinds of errors. We now consider how this can be used to develop adaptive labeling strategies.



(a) Error in $y = 1°$. (b) Error in $x = 1°$, $y = 1°$. (c) Error in $x = 1°$, $y = 1°$, $z = 11°$

Figure 3. The error ellipses and convex hulls for different values of the camera error. The standard deviations in the camera are set to twice the actual error injected into the camera position.



(a) Large inner hull. (b) Small inner hull. (c) No inner hull.

Figure 4. The inner hulls for different levels of error. The inner hull is the largest space that is contained between the covariance ellipses used to construct the exterior convex hull. It is shown as the inner box.

4. ADAPTIVE SYNTHETIC VISION

The system has two main cases: error adaptation for a single object, and error adaptation for multiple, interfering objects.

4.1. Adaptation for a Single Object

First consider the case that there is a single object that is sufficiently far from other objects that their error bounds do not interfere with one another. We use the following hybrid strategy: place the label on the object, or use a call out line, if the annotation is *guaranteed* to lie on the object. If it is not guaranteed to lie on the object, draw the convex hull of the object and label that instead.

To determine if the label or callout line lies inside the object, the *inner hull* is calculated. This hull is illustrated in Figure 4. It is the largest convex object that lies within the covariance ellipses that define the convex hull for the object. It also defines the region that is *guaranteed* to completely enclose the object. A label is placed directly on the object if it will fit within the inner hull. If the label does not fit, a callout line is placed at the centroid of the inner hull. If the registration error becomes sufficiently large that the covariance ellipses overlap, the inner hull does not exist.

The result from this adaptive synthetic vision scheme is shown in Figure 5 for a set of images. The UAV flies over the target building. The figure shows how the system automatically adapts the display depending upon the configuration of the UAV relative to the target.

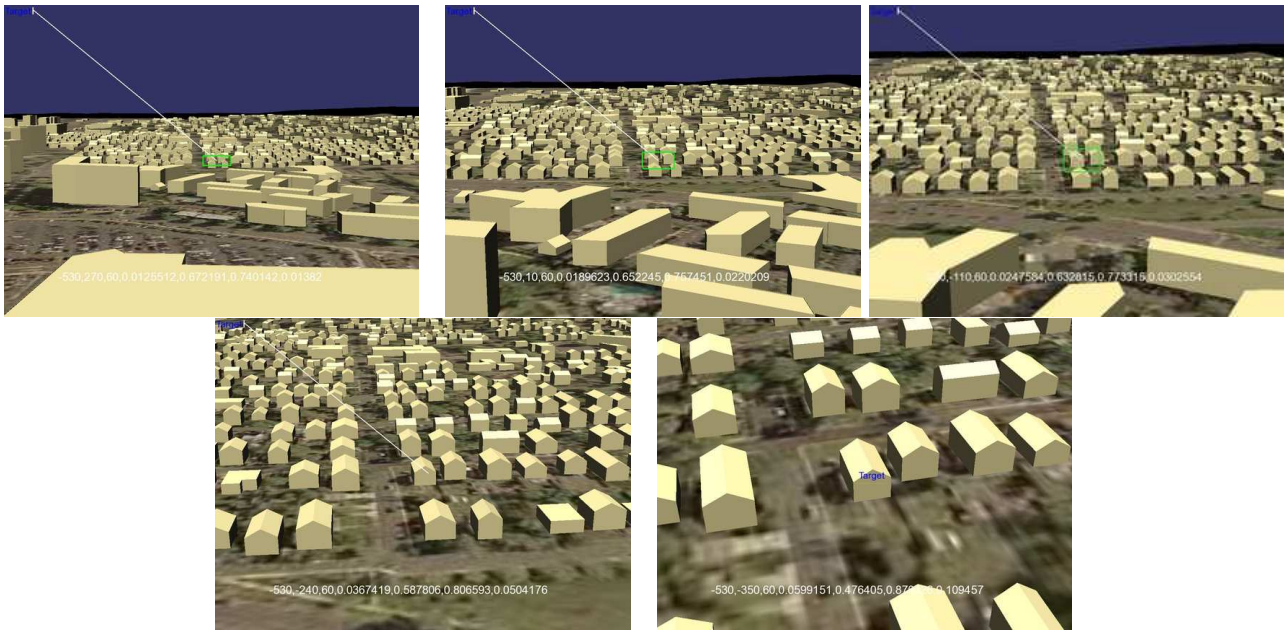


Figure 5. A set of images illustrating the single building adaptation scheme. Far from the target, where the inner hull does not exist, the convex hull is drawn. Once an inner hull exists, the call out points directly at the object. When there is sufficient space, the label is placed within the target.

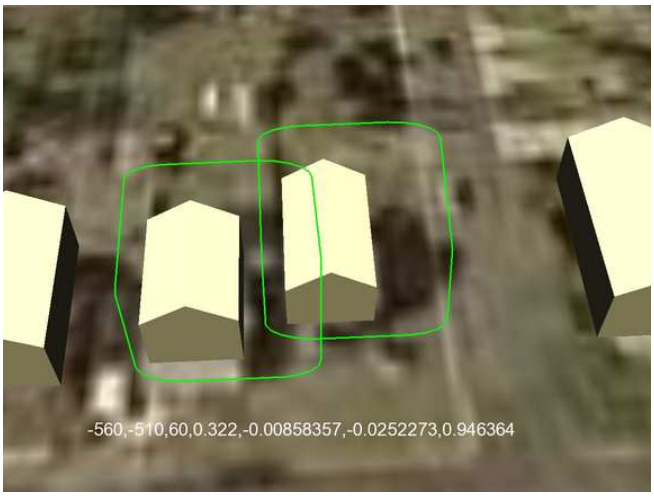


Figure 6. Interference between two nearby objects.

4.2. Registration Error Adaptation for Multiple Objects

Although the strategy from the previous section works with a single, unambiguous object, it does not work well with multiple closely spaced objects. The reason is illustrated in Figure 6 — the convex hulls for nearby objects overlap. This implies that *any* augmentation placed over one building has a non-trivial probability that it will appear over the other building. Cues such as "the augmentation applies to the building that it lies on the most" are indirect and can be misleading.

To address these issues, the interaction between multiple objects must be considered. We are developing methods to achieve this by using special types of grouping nodes. The idea is illustrated in Figure 7: the

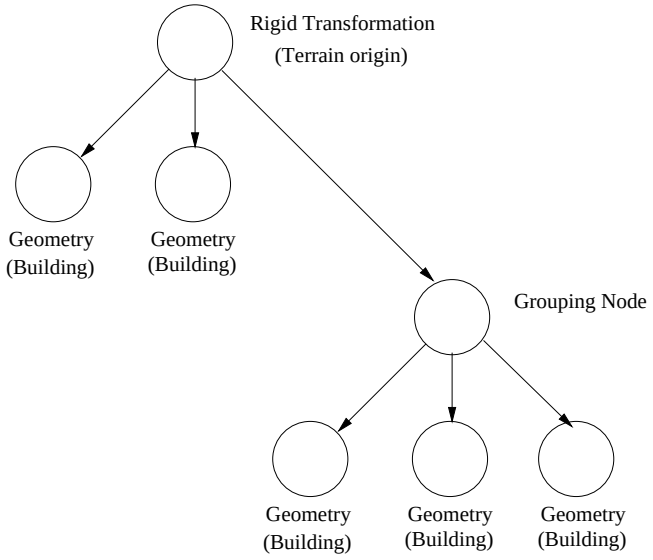


Figure 7. Using grouping nodes to combine geometry from multiple objects together.

grouping node contains, as children, all the objects that can interfere with one another and, from this, it can perform aggregation operations. One possibility is that the augmentation could be drawn as the convex hull together with disambiguating information. This could include textual (e.g., "the object to the left") or imagery (e.g., an inset could show a picture of the building in question, perhaps taken from roughly the point of the view of the UAV). These cues would be sufficient to allow the operator to disambiguate the input.

5. CONCLUSIONS

This paper has discussed the problem of registration errors in synthetic vision systems. We have argued that even small tracking errors, much smaller than those needed for an autopilot, lead to significant registration errors. We have demonstrated how error bounds are propagated through a hierarchical scene graph representation of the geometric relationships between objects in the scene and how these bounds are projected onto the final merged image. We demonstrated the system's use on a real-world dataset and have shown how it can disambiguate the correspondence between virtual annotations and real objects.

There are a number of important areas of research that need to be addressed. First, methods for automatically grouping objects must be developed. Any fielded system cannot rely on users manually grouping objects to flag potential interference. Second, the annotation methods described are highly dynamic and vary significantly in the way that they depict the same information. Care must be taken to ensure that the transitions between these representations are not confusing. One possibility would be to adapt space management algorithms for intelligent labeling.[19] Finally, no user studies have been carried out to evaluate the effectiveness of the adaptations. Such evaluations are vital if the performance benefits of these algorithms are to be quantified.

ACKNOWLEDGMENTS

The authors would like to thank Enylton Coelho (now at Bosch) and Blair MacIntyre at Georgia Tech for developing the OSGAR framework used to implement the adaptive registration system.

REFERENCES

1. J. L. Drury, L. Riek, and N. Rackliffe, "A Decomposition of UAV-Related Situation Awareness," tech. rep., The MITRE Corporation, 2005.

2. "Young, nathman seek low-cost, high- capability uavs," [online, cited 1 March 2006]. Available from: uav.navair.navy.mil/airdemo2005/2005\%20news/28\%20JUN\%2005.pdf.
3. R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent advances in augmented reality," *IEEE Computer Graphics and Applications* **21**, pp. 34–47, Nov/Dec 2001.
4. B. MacIntyre, E. M. Coelho, and S. Julier, "Estimating and adapting to registration errors in augmented reality systems," in *IEEE Virtual Reality (VR '02)*, pp. 73–80, March 2002.
5. E. M. Coelho, B. MacIntyre, and S. Julier, "osgAR: A scenegraph with uncertain transformations," in *International Symposium on Mixed and Augmented Reality (ISMAR 2004)*, pp. 6–15, November 2-5 2004.
6. E. M. Coelho, *Spatially Adaptive Augmented Reality*. PhD thesis, Georgia Tech, 2005.
7. J. A. Atherton, B. Hardin, and M. A. Goodrich, "Coordinating a multi-agent team using a mulitple perspective interface paradigm," in *Proceedings of the AAAI 2006 Spring Syposium: To Boldly Go Where No Human-Robot Team Has Gone Before*, 2006.
8. "Calisto Data AB — Projects," 2006. Available from: http://www.calisto.se/projects.htm.
9. "Rapid Imaging Software inc. — LandForm SmartCam3d," 2006. Available from: http://www.landform.com/documents/LF_SmartCam3D_Brochure.pdf.
10. "Nav3D Corp. — About synthetic vision," 2006. Available from: http://www.nav3d.com/aboutsynth/aboutsynth.html.
11. A. Goktogan and S. Sukkarieh, "An augmented reality system for multi-UAV missions," in *Proc. SimtecT*, May 2005.
12. M. Quigley, M. Goodrish, and R. Beard, "Semi-autonomous human-uav interfaces for fixed-wing mini-uavs," in *Proc. Intelligent Robots and Systems*, Sep 2004.
13. W. Bath and J. Paxman, "UAV localisation & control through computer vision," in *Proc. Australasian Conference on Robotics & Automation*, Dec 2004.
14. G. Calhoun, M. Draper, M. Abernathy, F. Delgado, and M. Patzek, "Synthetic vision system for improving unmanned aerial vehicle operator situation awareness," in *Enhanced and Synthetic Vision 2005*, J. Verly, ed., *Proc. SPIE* **5802**, pp. 219–230, 2005.
15. K. Shoemake and T. Duff, "Matrix animation and polar decomposition," in *Proceedings of Graphics Interface (GI '92)*, pp. 258–264, (Vancouver, BC, Canada), May 11–15 1992.
16. R. Hartley and A. Zisserman, *Multiple View Geometry*, Cambridge University Press, 2nd ed., 2003.
17. I. Y. Bar-Itzhack and J. Reiner, "Recursive attitude determination from vector observations: Dcm identification," *Journal of Guidance, Control and Dynamics* **7**, pp. 51–56, January – February 1984.
18. I. Y. B.-I. D. Choukroun, H. Weiss and Y. Oshman, "Kalman filtering for matrix estimation," *IEEE Transactions on Aerospace and Electronic Systems* **42**, January 2006.
19. B. Bell, S. Feiner, and T. Höllerer, "View management for virtual and augmented reality," in *Proc. UIST '01 (ACM Symp. on User Interface Software and Technology)*, pp. 101–110, (Orlando, FL, USA), 11–14 November 2001.