Proceedings of the 2004 Winter Simulation Conference
R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, eds.

A REAL-TIME PANORAMIC VISION SYSTEM FOR AUTONOMOUS NAVIGATION

Sumantra Dasgupta
Amarnath Banerjee

Department of Industrial Engineering
3131 TAMU
Texas A&M University
College Station, TX 77843 U.S.A.

ABSTRACT

The paper discusses a panoramic vision system for autonomous navigation purposes. It describes a method for integrating data in real-time from multiple camera sources. The views from adjacent cameras are visualized together as a panorama of the scene using a modified correlation based stitching algorithm. A separate operator is presented with a particular slice of the panorama matching the user's viewing direction. Additionally, a simulated environment is created where the operator can choose to augment the video by simultaneously viewing an artificial 3D view of the scene. Potential applications of this system include enhancing quality and range of visual cues, navigation under hostile circumstances where direct view of the environment is not possible or desirable.

1 INTRODUCTION

Panoramic images are often used in Augmented Reality, digital photography and autonomous navigation. There are various techniques for the creation of panoramic images (Benosman and Kang 2001; Szeliski 1996). Based on the amount of time taken to render the panorama after the actual pictures have been taken, one can categorize panoramic imaging as real-time and offline. Applications such as autonomous navigation need real-time imaging whereas applications such as virtual walkthrough can work on offline panoramic imaging. Panoramic imaging can also be categorized based on the capture devices used. One can either use (i) a static configuration of multiple cameras, (ii) a revolving single camera or (iii) a combination of cameras and mirrors, generally known as catadioptrics (Gluckman *et al.*, 1998). In the first case, the final presentation of the panorama is real-time because images from various cameras can be simultaneously captured at 30fps, which leaves time for post-processing (mainly stitching) and still meet the real-time constraints. In the second case, the camera needs time to revolve which makes the effective capture rate fall drastically below real-time requirements. In the third method, the panorama of the scene is captured by reflection from the mirror element and presented to the camera capturing frames at rates of around 30fps. So, the presentation of the image can be made real-time after post-processing (here post-processing is mainly that of mapping curvilinear coordinates to linear coordinates).

We develop a method for integrating data in real-time from multiple camera sources. The camera setup is static with respect to the base to which it is attached. Movement is manifested in the form of motion of the base (e.g. cameras attached to an automobile). This is referred to as *direct motion* of camera setup. Movement can also be manifested in the form of head movement of a user wearing a head mounted display (HMD). In this case, the cameras do not move, but the user is able to see in the direction to which s/he is looking. This is referred to as *indirect motion*. Here, we consider a scenario where a user wearing a HMD sits within a moving vehicle to which the cameras are rigidly attached.

The procedure starts with the video sequence from various cameras being stitched together to form a *panoramic* image. Then the image is rendered according to the viewpoint of the user (or *indirect motion*) and the orientation of the van (*direct motion*). This is referred to as *windowing*, and the rendered view is referred to as the *window view*. The *window view* of the panoramic image is presented to the user for viewing. For separate *monitoring* purposes, the whole panoramic image sequence is also fed to a separate video display with an *on-screen video control-panel* to a separate operator. An advanced form of rendering/viewing is done by superposing the real-time video onto a 3D model of the view area using chroma-keying. This adds a level of redundancy and extra security for navigation purposes.

For the present approach, multiple camera configuration has been used because the configuration is smoothly scalable in both horizontal and vertical directions. Moreover the configuration can be easily extended for stereo viewing. In the present approach, multiple cameras are arranged in a static octagonal setup. The method is described in the following steps.

2 CAMERA CONFIGURATION

For horizontal panoramic viewing, the cameras can be arranged in a regular horizontal setup (Figure 1). For vertical viewing, the cameras can be stacked up (Figure 2). For all-round viewing, the cameras can be arranged in a regular 3D arrangement (Figure 3). For all these setups, there will be blind spots very close to the camera setup (precisely at any distance less than the focus). This is not a major issue in practical applications, if the user is interested in objects at and beyond the focus. However, it is undesirable to have blind-spots in the direction of motion. Hence, one of the cameras always faces directly forward. Here, we use a configuration similar to Figure 1.

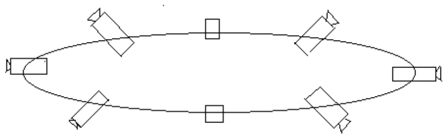
Figure 1: A Planar Camera Configuration

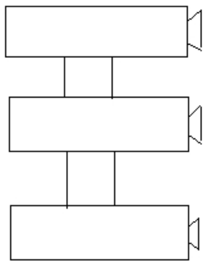
Figure 2: A Stacked Camera Configuration

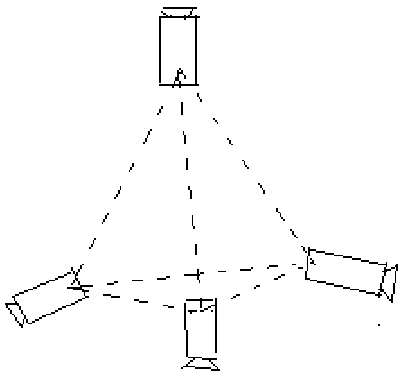
Figure 3: 3D Camera Configuration

3 READING DATA FROM MULTIPLE CAMERA SOURCES

In case of analog cameras, each image source transfers images in an analog format (usually NTSC at 30fps). This analog data is converted to digital data via a video capture card (equipped with a A/D converter) and is stored in memory for further processing. In case of digital cameras, no digital conversion is necessary. It has been observed in practice that the whole process of capture and transfer of images from a consumer grade NTSC analog camera (with standard capture cards) is faster than that of its digital counterpart (this observation is purely empirical). Based on the above observation, the setup that has been used here uses analog cameras (Canon PTZ cameras) with capture cards (made using TI converters). The general architecture of the system is outlined in Figure 4.

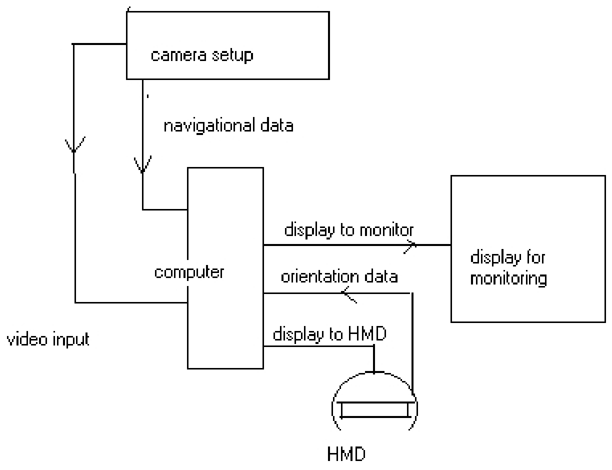

Figure 4: System Architecture

4 STITCHING PROCESS

The cameras are arranged cylindrically on a plane. Every adjacent pair of camera is so arranged that there is a degree of overlap between the images captured by the two (Figure 5). Stitching is the process of eliminating the overlap from adjacent cameras and restoring continuity in the images. Two popular stitching techniques are discussed.

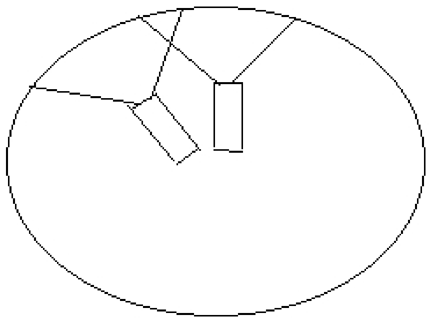
Figure 5: Field of View and Overlap between Adjacent Cameras

It has been observed that, when the individual shots of a scene differ just by the panning angle(yaw angle) of the camera, then a simple warping operation, matching with the motion of the camera, is enough to render a seamless panorama. Since the cameras are arranged cylindrically, the images from the cameras are cylindrically warped (Benosman and Kang 2001). The world coordinates **w** = (*X, Y, Z*) are mapped to 2D screen co-ordinates (*x,y*) using:

$$x = \arctan\left(\frac{X}{Z}\right);\ y = \frac{Y}{\sqrt{X^2 + Z^2}}$$

where *Z* is the focal length of the camera in pixels and *X,Y* are the coordinates before warping. Once the images have been warped, construction of a panoramic mosaic can be done using pure translation (Szeliski 1996).

Since, multiple cameras are being used, their optical centers are offset from the true center or the origin of the configuration. This causes the previously mentioned cylindrical warping unsuitable for the present application. A more detailed 8-parameter perspective projection is used.

Any two adjacent cameras are chosen. The left camera is taken as the reference camera and we try to warp the image from the right camera onto the image from the left camera. The original camera configuration is made in a way such that images from adjacent cameras overlap. In the present setup, the images from adjacent cameras overlap by 5 degrees. The perspective projection applied to the overlap regions can be described as follows.

In planar perspective projection, an image is warped into another using

$$x' = Mx = \begin{bmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & m_8 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where x=(*x,y,1*) and x'=(*x',y',1*) are the homogeneous coordinates. For calculation purposes, the previous equation can be rewritten as,

$$x' = \frac{m_0x + m_1y + m_2}{m_6x + m_7y + m_8};\ y' = \frac{m_3x + m_4y + m_5}{m_6x + m_7y + m_8}.$$

In order to recover the parameters, we aim to minimize the squared of intensity errors,

$$E = \sum_i \left[I'\left(x_i', y_i'\right) - I(x_i, y_i) \right]^2 = \sum_i e_i^2$$

over the region of overlap between the left and right images. Pixels lying outside the boundary of the overlap do not contribute to the error calculation. It is observed that the points (*x',y'*) do not fall on integral pixel coordinates and hence bilinear interpolation is used for intensity mapping in *I'*.

The minimization is performed using the Levenberg-Marquardt iterative nonlinear minimization algorithm (Press *et al*., 1992). Partial derivatives of e_i are calculated with respect to each of the perspective projection parameters. For example,

$$\frac{\partial e_i}{\partial m_0} = \frac{x_i}{d_i}\frac{\partial I'}{\partial x'};\ \frac{\partial e_i}{\partial m_6} = -\frac{x_i}{d_i}\left(x_i'\frac{\partial I'}{\partial x'} + y_i'\frac{\partial I'}{\partial y'}\right)$$

where $d_i = m_6x + m_7y + m_8$ and $\frac{\partial I'}{\partial x'}, \frac{\partial I'}{\partial y'}$ are the image intensity gradients of *I'* at (*x',y'*). These partial derivatives are used to derive the Hessian matrix ***A*** and the weighted gradient vector ***b*** as follows,

$$a_{kj} = \sum_i \frac{\partial e_i}{\partial m_k}\frac{\partial e_i}{\partial m_j} \text{ and } b_j = -\sum_i e_i \frac{\partial e_i}{\partial m_j}.$$

These are then used to update the projective matrix parameter estimate **m** by using

$$\Delta\text{m} = A^{-1}b.$$

So, for the next iteration

$$\mathbf{m}_{\text{new}} = \mathbf{m} + \Delta\mathbf{m}.$$

The complete registration algorithm can be described as follows,

For each pixel *i* at (x_i, y_i) in the reference image,

1. Compute its position in the adjacent image $\left(x_i', y_i'\right)$.
2. Compute the error e_i, at the following pixel location. Also calculate the gradient image at that location using bilinear interpolation on *I'*.
3. Compute the partial derivatives of error e_i with respect to the 8 projective parameters.
4. Add each pixel contribution to the components of ***A*** and ***b***.
5. Solve ***A*Δm**=***b*** and update the perspective projection parameters using $\mathbf{m}_{\text{new}} = \mathbf{m} + \Delta\mathbf{m}$.

The method stops when error starts to increase.

Perspective matrix is calculated for each camera pair and the global perspective matrix $\mathbf{m}_{\text{global}}$ is calculated by averaging out over all such matrices. This whole operation is performed offline. Since the camera configuration is static, the value of $\mathbf{m}_{\text{global}}$ is used for subsequent real-time applications.

Once the perspective parameter $\mathbf{m}_{\text{global}}$ has been calculated, the overlap from the right image is warped into the left image. In order to maintain right continuity of the overlap with non-overlapping portions of the right image, weighted warping is used. The basis is, closer is the overlap pixel to the right non-overlapping boundary, the lesser is it warped to the left image. Figure 6 illustrates the operation.

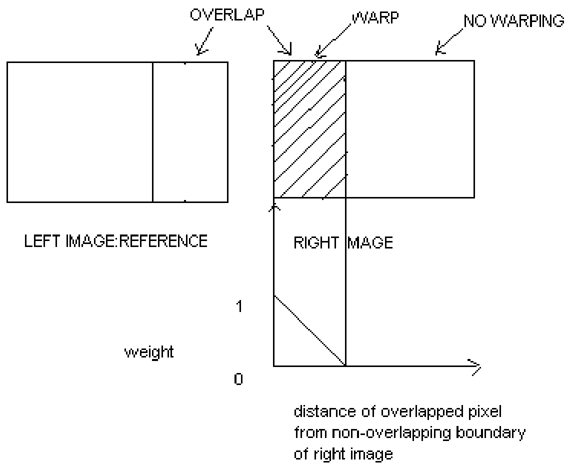
OVERLAP
WARP
NO WARPING
LEFT IMAGE:REFERENCE
RIGHT IMAGE
1
weight
0
distance of overlapped pixel from non-overlapping boundary of right image

Figure 6: Weighted Warping

In theory, the previous method should complete the stitching process, but some practical issues must be considered. The motion of the vehicle might make a particular camera to become misaligned with the rest of the setup. As described later, an operator sitting on a separate terminal can notice the misalignment and reorient the camera.

5 GET NAVIGATIONAL DATA

The navigational data describes the position and relative orientation of the cameras with respect to the ground (*direct motion*). A GPS source is used to find the orientation and position of the camera setup at zero or home position. At each time interval, the navigational data, **nd** is a vector with 6 components:

$$\mathbf{nd}=[\text{latitude, longitude, altitude, roll, pitch, yaw}]^T.$$

This data is continuously refreshed as the cameras move and is fed to the stitching/rendering for adjustments to the image panorama.

6 GET ORIENTATION DATA FROM HMD

The orientation describes the orientation of the operator with respect to the cameras (*indirect motion*). The orientation data is refreshed as the operator move or turn their head. This data is fed to the stitching/rendering program along with the navigational data. Similar to the navigational data, the orientation data, **od** is a 6-tuple:

$$\mathbf{od}=[\text{latitude, longitude, altitude, roll, pitch, yaw}]^T.$$

If it is assumed that the user is in a sitting position and the movements are restricted to mainly head rotations, the navigational and orientation data differ only in the last three parameters describing orientation.

7 CREATE WINDOW VIEW

Here, we apply orientation and navigational data to the panoramic image to retrieve the portion of the panorama in the direction in which the operator is looking. It has been assumed that the camera setup is static with respect to the base to which it is attached. In order to appropriately calculate the viewpoint of the operator, the following formula is used:

$$\text{View-parameters, } \mathbf{vp} = \mathbf{od} - \mathbf{nd}.$$

As mentioned earlier, the two sets of data differ mainly in roll, pitch and yaw parameters. The translational parameters of latitude, longitude and altitude have been ignored.

In practice, yaw motion is the most dominant motion which the user undergoes as s/he looks from side to side. The relative yaw of the user with respect to the cameras is calculated as,

$$\text{Yaw} = \mathbf{od}[6] - \mathbf{nd}[6].$$

Once the appropriate yaw angle is calculated, the starting point of the window view is calculated as,

$$\textit{Start} = \text{Yaw*Width(Panoramic Image) in pixels}/360^o.$$

This formula translates the yaw angle in pixel coordinates. So, if the original panoramic image is denoted as ***P***, then the yaw adjustment chooses a portion of the panorama commencing at *Start* and ending according to the viewing format chosen by the user.

The yaw adjustment is the first adjustment done to the panoramic image. This is because, this is the most dominant motion of the camera setup/user and hence retrieves the most information about the current view-point. This is followed by the roll adjustment and finally by the pitch adjustment. The pitch adjustment is performed last because, for the horizontal setup of static cameras used here, there can be a loss of information due to pitch (it might define viewpoints which are partially or fully above and below the span of the camera in the vertical direction).

Let the cropped portion of the image after the yaw operation be denoted by ***yP***. The roll operator is applied next. The roll is calculated as,

$$\text{Roll} = \mathbf{od}[4] - \mathbf{nd}[4].$$

In theory (Forsyth and Ponce 2003), roll can be described by the rotation operation. For rotating (x, y) to (u, v) by θ, where both have the same origin, one has,

$$u = x\cos\theta - y\sin\theta;\ \ v = x\sin\theta + y\cos\theta.$$

Although the above formula describes 2D rotation in theory, its real-time implementation is quite time consuming. This is mainly because it requires applying some transformation (bilinear transformation) to the rotated image in order to deal with fractional pixel coordinates. Here, an approximation to the present formula is used. For rotating (x,y) to (u,v) by θ, where both have the same origin,

$$u = x - |y \sin\theta|;\ v = |x \sin\theta| + y.$$

The main assumption behind the approximation is that roll angle is small. This formula obviates the use of bilinear transformation but it introduces rectangular artifacts in the rotated image. The distortion is not much and it reduces as the resolution of the picture increases. This can be explained as follows. As the resolution increases, $|x\sin\theta|$ or $|y\sin\theta|$ have more probability of converging to a unique pixel position thus reducing the dimensions of the rectangular artifacts mentioned earlier.

The pitch operation is applied after applying yaw and roll operations. At the end of roll and yaw operations, the cropped panorama can be denoted as ***ryP***. The pitch operation can be described as,

$$\text{Pitch} = \mathbf{od}[5] - \mathbf{nd}[5].$$

After deriving the pitch angle from the above formula, the equivalent pitch shift in pixel coordinates is calculated as,

$$\textit{PitchShift}=\text{Pitch*Height(camera img) in pixels}/50^0.$$

At the end of the pitch operation, the image ***pryP*** is presented to the operator for viewing. This is the window view for the operator.

8 DISPLAY WINDOW VIEW TO THE USER

The user views the window view according to the aspect ratio chosen. It can be viewed as wide (aspect ratio 16:9) or normal (aspect ratio 4:3).

9 DISPLAY PANORAMA FOR SEPARATE MONITORING

The whole panorama is displayed to a separate monitor for a different operator to adjust camera misalignments, adjust other camera parameters such as contrast, brightness, color etc. The second operator can also implement some special adjustments as instructed by the user wearing the HMD.

The control of cameras and view from the cameras have been separated with the sole intention of keeping the user with HMD free to take over other controls, such as maneuvering the vehicle. Although, this has not been attempted in the present experimental setup, it will be implemented later as a separate camera feeding in a picture of the control panel of the vehicle onto the HMD (the control panel will be constantly present at the bottom of the stitched image and will not be affected by the relative roll, pitch and yaw motions).

10 AUGMENTED REALITY APPLICATION

Augmented reality is a procedure of augmenting a real scene with an artificial scene or vice-versa. In the present paper, the operator can actually view both the real and artificial scene simultaneously on the same screen. The artificial scene is created using GeoTiff reference images of the surrounding environment. We used a commercial artificial scene generation software for providing both the views simultaneously. A snapshot of the augmented reality display is shown in Figure 7.

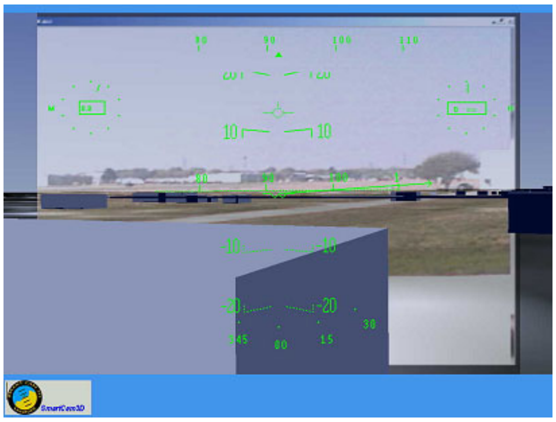

Figure 7: Augmented Reality Snapshot

11 RESULTS AND DISCUSSION

A few results are shown from the eight camera setup. Figure 8 demonstrates stitching from adjacent cameras. Figure 9 shows roll as described above. As indicated earlier, it is suitable only for small roll angles. When the roll angle increases distortion becomes pronounced, which is illustrated in Figure 10. Figure 11 shows the incorporation of pitch shift in the stitched image (the blank space signifies a portion of space not covered by the camera setup). For navigational purposes, the cameras were mounted on top of a van. A portion of a real-time panorama as shot and visualized in the moving van is shown in Figure 12.

The real-time constraint for the experimental setup requires that the stitched frames be presented faster than the retentivity of the human eye (1/12 seconds). The stitched and rendered images are generated at a rate of about 14fps with a resolution of 1024 by 768 (the maximum resolution possible in the HMD). The pitch action causes information loss. This is because all the cameras are situated on a particular plane and their field of view in the vertical direction is restricted to +/- 25^0 (the field of view (FOV) of the lens of the cameras used is 50^0 and they are centered on the horizon) above and below the center line. So, any pitch beyond +/- 25^0 cannot be anticipated by the present setup. In future, a more elaborate 3D camera setup will be attempted. The stitching algorithm is practical but is limited due to the fact that stitching is done only at a particular distance from the camera. So, as objects move closer or beyond that distance, stitching is affected. To obviate this, catadioptric systems will be explored in future. To speed up the process further, hardware implementations of various portions of the algorithm (especially roll) will be attempted.

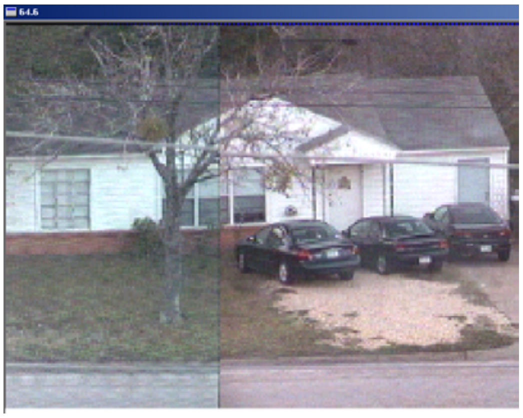
Figure 8: Stitching and a Yaw of 60^0

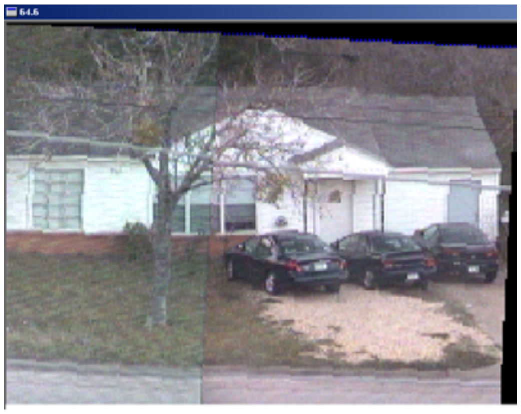
Figure 9: Stitching and a Roll of .05 Radians

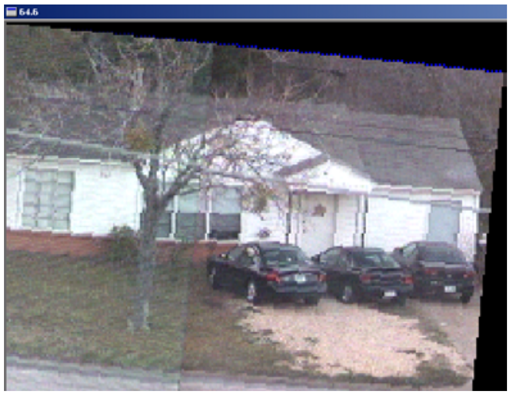
Figure 10: Stitching and a Roll of 1.5 Radians

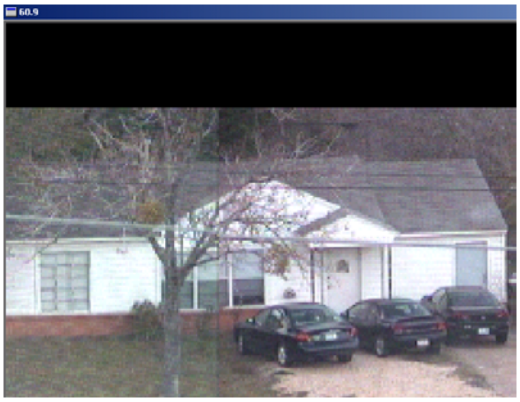
Figure 11: Stitching and Pitch

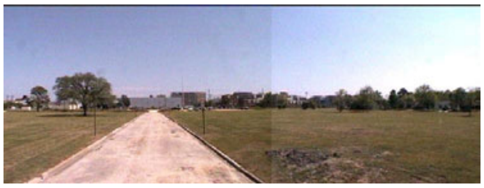
Figure 12: Panoramic Stitching

REFERENCES

Benosman, R., and S. B. Kang., eds. 2001. *Panoramic Vision: Sensors, Theory and Applications*. 1st ed. New York, NY: Springer-Verlag.

Forsyth, D. A., and J. Ponce. 2003. *Computer Vision: A Modern Approach.* 1st ed. Upper Saddle River, NJ: Prentice Hall. Gluckman, J., S. K. Nayar, and K. J. Thoresz. 1998. Real-time omnidirectional and panoramic stereo. In *Proceedings of DARPA Image Understanding Workshop*, Monterey, CA. Available online via`cis.poly.edu/~gluckman/papers/iuw98a.pdf`> [accessed August 5, 2004].

Press, W. H., B. P. Flannery, S. A. Tuokolskym, and W. T. Vetterling. 1992. *Numerical Recipes in C: The Art of Scientific Computing.* 2nd ed. Cambridge, England: Cambridge University Press.

Szeliski, R. 1996. Video Mosaics for Virtual Environments. *IEEE Transactions on Computer Graphics and Applications* 16 (2):22-30.

ACKNOWLEDGMENTS

This material is based upon work supported by NASA under Grant No. NAG9-1442. The authors would like to acknowledge the assistance of Lt. Col. Eric Boe, Jeffrey Fox, James Secor, and Patrick Laport at the Johnson Space Center in Houston, TX for their help and assistance in integrating the system in the Advanced Cockpit Evaluation System (ACES).

AUTHOR BIOGRAPHIES

AMARNATH BANERJEE is an Assistant Professor in the Department of Industrial Engineering at Texas A&M University. He received his PhD in Industrial Engineering and Operations Research from the University of Illinois in Chicago in 1999, and BS in Computer Science from the Birla Institute of Technology and Science, Pilani, India in 1989. His research interests are in virtual manufacturing, simulation, image processing, real-time video processing, augmented reality and human behavior modeling. He is the Director of Advanced Virtual Manufacturing and Augmented Reality Laboratory at Texas A&M University. He is a member of IIE, IEEE and INFORMS. His e-mail address is `<banerjee@tamu.edu>`, and his web address is `<ie.tamu.edu/people/faculty/banerjee>`.

SUMANTRA DASGUPTA is a PhD student in the Department of Industrial Engineering at Texas A&M University. He received his MS in Electrical Engineering from the same university in 2003, and BE in Electrical Engineering from Birla Institute of Technology, MESRA, India in 2000. His research interests are in computer vision, image processing, video processing and signal processing. His e-mail address is `<sumantra-dasgupta@neo.tamu.edu>`.