Simulation of Hexagon Spatial Image Datasets for Free Motion in a Simulator for Smart City Bidirectional Navigation Purposes

Lepekola I. Lenkoe
Department of Electrical, Electronic and Computer Engineering
Central University of Technology, Free State
Bloemfontein, South Africa
email: pexonl3@gmail.com

Ben Kotze
Department of Electrical, Electronic and Computer Engineering
Central University of Technology, Free State
Bloemfontein, South Africa
email: bkotze@cut.ac.za

Pieter Veldtsman
Department of Electrical, Electronic and Computer Engineering
Central University of Technology, Free State
Bloemfontein, South Africa
email: pveldtsm@cut.ac.za

***Abstract*—It has been noted that Google Street Views serves millions of users with panoramic imagery across the globe. However, the configuration model such as the use of a 360^0 omnidirectional camera is utilised. However, an optimal model for capturing, calibrating, and compressing the captured images differs. It is with such reasons that a hexagon camera configuration model is investigated including but not limited to imagery capture and simulation of results to allow for bidirectional navigation within a simulator with the ability to view the initially uncaptured scenery. On the configuration model, cameras were placed at 60^o angle apart to obtain the full 360^o scenery view. It is seen that this model can optimally produce a full 360^o panoramic view with the capabilities for a bidirectional view and movement of the initially uncaptured scene/ view through the application of the image rendering technique.**

Keywords-Simulation; Image-Based Rendering; Spatial datasets; Google Street Views; Smart Cities.

I. Introduction

The smart city concept has the potential to capture real-time data that communicates with stakeholders for optimising decision-making utilising artificial intelligence and a low latency response rate. In addition, the modelling and visualisation of complex processes and data are significant. Hence, the hype in the furthering the concept of Google Street Views (GSVs) for mapping and documenting of rendered spatial built environment [1].

Furthermore, GSV is deemed as a technology implemented in several Google services to provide the user interested in viewing a particular location on the map with panoramic images [2]. In addition, the GSV implementation in most cases is achieved utilising the Image-Based Rendering (IBR) technique. Despite the selection of the IBR technique in this paper, several modelling techniques can be utilised to achieve the same results such as the Model-Based Rendering (MBR) technique. However, the construction and implementation of such techniques rely on techniques such as Structure of Motion, which is used to build 3D models from both structured and unstructured image collection depending on the dataset model [3].

A. Problem statement

In most GSV-based applications, a 360^o omnidirectional camera is utilised for image capture. However, as a result of having a single dataset, the application becomes limited and reduces the application of a full 3D panoramic view processing power. It is for such reasons that a new image capture technique utilising six (6) camera configurations at 60^o angle needs to be tested and investigated and modelled for a full 3D panoramic view to observe the feasibility for free motion in a simulator for bidirectional imagery viewing of the initially uncaptured scene utilising an alternative model configuration as opposed to 360^o omnidirectional camera.

B. The objective of this study is therefore to:

- Simulate a rendering technique for improvement of visual, spatial, and quality of the panoramic images for location identification.
- Present a framework that allows for omnidirectional virtual driving.
- Model image data collection technique utilising hexagon camera configuration model.

C. Original contributions of this research paper

This research paper has produced the following contributions in furthering the knowledge contribution in the field of computer vision as follows:

- Development of a rendered panoramic image model for the enhancement of virtual driving through incorporated image datasets for utilisation in a simulator.
- Image capturing technique for improvement of human interaction with the virtual world while traveling through a smart city.

This paper is organized as follows: Section II gives the background to the work conducted in this field also pointing out the shortcomings from the research conducted. Section III presents the configuration model for image capture and data processing techniques, while Section IV discusses the results obtained from the study. Section V draws the conclusion of the results obtained from the study.

II. LITERATURE REVIEW

The use of datasets in a simulator for constructing image rendering and camera content acquisition in a 3D content view is regarded as a significant approach in such a study.

It is with such reasons, that a study from Li et al. [4] outlines the significance of cities in the context of global warming and urbanisation that is also interpreted and analysed for further developments.

This is as a result of the view dependency, which means that the explicit geometric rendering much relies on the known approximate environment [5]. The use of the explicit rendering becomes complex in an informal environment/settlement due to the tiring exercise of data collection, which is skewed since residents can be built on any topography. However, the observation of the feature detection and matching technique utilising IBR technique for multiple image datasets is feasible to achieve the objectives of such a study [6].

It is with such reasons supporting the use of the advanced IBR method presented by Mao et al., [7] and Shi et al., [8] outlining the depth map containing associated pixels to the reference image in a 3D environment. It is noted according to the literature that depth estimation from a single image is an important issue in understanding a 3D scene.

In addition to the parametric methods for extracting depths, many non-parametric depth sampling approaches have also been proposed to automatically convert monocular images into stereoscopic images with good performances.

Unfortunately, vision techniques are not robust enough currently to recover accurate 3D models. In addition, it is difficult to capture visual effects such as highlights reflections and transparency making use of single texture-mapped model.

It is for such reasons, that the rendering techniques are associated with computer graphics to enable better processing of images [9].

The evolution of such an innovation is mainly based on the increased data resources, multiple spatial datasets, and tools for processing and computation [10] [11]. This paper focuses on the use of multiple cameras rather than the use of a single 360^o omnidirectional camera to allow for bidirectional imagery viewing of the initially uncaptured scene. It is noted against the literature that much of the work has been conducted in this field of study, however, the exploration of IBR technic utilising other configuration model apart from the use of a 360^o omnidirectional camera has not yet fully explored. Furthermore, the use of the 360^0 omnidirectional camera has still not yet proved the concept of imagery view of the uncaptured scene.

III. METHODOLOGY

Figure 1 depicts the camera configuration setup. However, this model can be altered depending on the testing site, and theoretically, the number of the cameras does not affect the results, but rather affects the resources required to process the output.

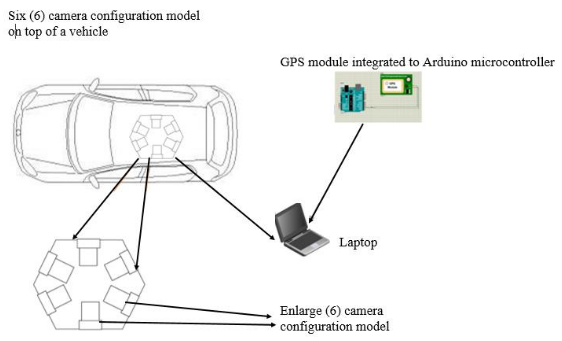
Six (6) camera configuration model on top of a vehicle
GPS module integrated to Arduino microcontroller
Laptop
Enlarge (6) camera configuration model

Figure 1. Architectural proposition for data collection and rendering using IBR technology.

The data collection was conducted utilising a vehicle that is mounted with six cameras at the top with the processing of the algorithms simulated in Blender3D for both image compression and image rendering. The camera configuration model consists of six cameras that are placed in a hexagon formation at a 60^o angle between the cameras.

All six cameras are connected to a laptop that is used for image storage, data processing, and data computation. Each image is Geotagged using a Global Positioning System (GPS) module that is attached to the Arduino microcontroller.

The process is started by allowing the cameras to capture individual images at a 3ms switched interval between the cameras. The switching duration was selected to allow for proper transition between the cameras taking into account the number and size of the images. The switching algorithm is processed from Arduino Microcontroller as follows:

$$T = \frac{A}{S} \qquad (1)$$

Where: A = data transfer speed (1.5 MB) raw data,
S = speed (480 MB/s) USB 2.0 speed

Following the image capture phase, the captured images are downloaded and manually placed into a single folder based on their timestamp.

The reason for obtaining GPS coordinates is to ensure that the images can be merged to obtain a panoramic view with approximately the same timestamp and location.

Other non-geometric methods that utilises computer vision functions, such as plenoptic function are utilised for allowing the intensity of light rays to pass through the camera centre at every location (L_x, L_y, L_z) and at every possible angle $(\theta, \emptyset)$ for every wavelength λ, at every time t were tested [12]. In this paper, two algorithms were tested namely; Lossy and Lossless algorithm. However, It was noted that the Lossy compression algorithm was the best algorithm for utilisation in this paper based on the reduced image size and the less reduction of the image quality as compared to losslessly compression that can still do the same but with the image sizes being a concern.

The plenoptic function is expressed as follows:

$$P_7 = P(L_x, L_y, L_z, \emptyset, \theta, \lambda, t) \qquad (2)$$

The images are projected from the laptop, and keyboard inputs are being used to navigate through the rendered scene. Furthermore, the GPS information such as (longitude, latitude, and elevation) from the scene is written to a text file, which is stored in a specified directory. The cylindrical panorama was also utilised due to its ease to build method for the single unit camera.

A. *System apparatus setup*

The following components were selected as indicated in Figure 2 as the primary apparatus for this research. It is important to note that the development of the physical model was constructed before any process can take place.

This task needed to take place for the system component's functionality to be tested in a stationary environment before they can be placed on the test vehicle.

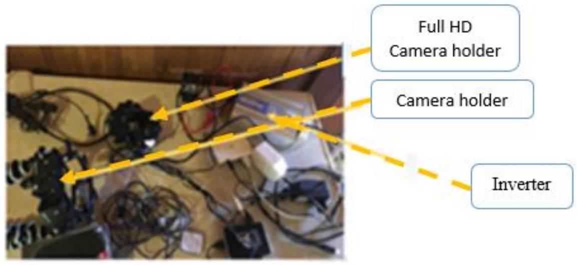

Figure 2. System components setup.

The six Full HD Action cameras were selected and utilised to capture quality images that do not require any customisation which might delete some important data from the image such as image descriptor/GPS coordinates.

Blender3D was selected and utilised due to its ability to convert and remodel files other than the .bli files and also because it has the capabilities for code re-use. Furthermore, Blender3D was compared against Cinema4D as indicated in Table I as follows:

TABLE I. CINEMA4D COMPARISON TO BLENDER3D.

	Cinema 4D	Blender3D
Availability	Paid, R700 – 1200 per month	Free
Source	Closed	Open
Applications	Animation Rendering Texturing	Animation Rendering Texturing 3D printing
Learning curve	Easy to learn	A hard learning curve at the beginning
User interface	User-friendly	Not such intuitive

Table I depicts the reasons for the selection of Blender3D over Cinema4D. There are other softwares available, however, for this paper, it was deemed necessary to evaluate only these two softwares due to resource constraints.

Furthermore, the system design model is based on the initial model designed on Microsoft® Visio Professional 2016, as indicated in Figure 3.

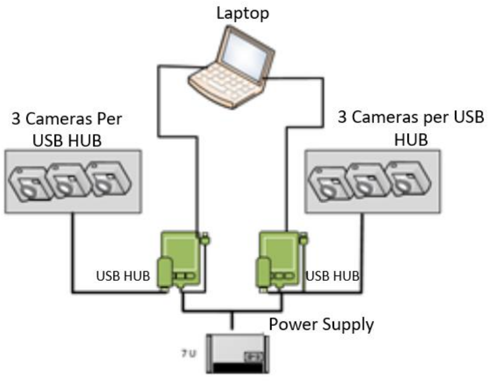

Figure 3. Camera configuration setup.

The camera configuration setup outlined in Figure 3 depicts the system connections of the 2 hubs, which are connected to an ACER laptop that processes the incoming data from the cameras. There are 6 cameras, which are divided into 2 sectors/groups of 3 cameras per hub.

Due to the much-required processing power, the laptop specifications had to be modified for rendering and processing to take place without any interruptions. Additionally, an external battery with a 1200mAh capacity was purchased and used to power the two hubs and act as an extra power supply to the laptop due to the high amount of power needed to keep the system active.

Furthermore, the system test duration was calculated as indicated in (3) as follows:

$$B_d(A) = \frac{C_c * C_{bv} * N_{b_s}}{L_c} = \frac{0.9Ah * 3.8v * 3}{11.4w} = 0.9\text{hr} \quad (3)$$

$$\underline{= 1 \text{ hrs testing}}$$

<u>Where:</u>

$B_d(A/B)$ = Battery duration for Hub (A or B); C_c = Camera battery capacity; C_{bv} = Camera battery voltage capacity; N_{b_s} = number of batteries connected in series; L_c = Load connected in Watts

B. *Image compression*

The Lossless compression algorithm is performed for the redundant processing of image information. Additionally, the image dataset is simulated without noise for better performance verification.

C. *Footnotes*

Other camera information including but not limited to intrinsic and extrinsic camera properties are important during the camera calibration, however, they are deemed to be camera-specific.

Moreover, the information about the focal length (f_x, f_y) and optical center (C_x, C_y) is critical. The focal length and optical center are then used to create the camera matrix, which is used to remove the distortion due to the lenses, which are camera-specific.

The camera matrix is then calculated utilising the focal length and the optical centers for the reused of images that are captured utilising a specific camera model as follows:

$$C_m = \begin{matrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{matrix} \quad (5)$$

Where: C_m = Camera matrix, (C_x, C_y) = optical centres, (f_x, f_y) = focal length

The system setup approach makes use of the chess pattern for the camera calibration setup. Some calibration methods in the literature rely on 3D objects.

The camera calibration process is as follows:

- Capture 20 chessboard images from different poses;
- Find the chessboard corners;
- Find the intrinsic matrix, distortion coefficients, rotation vectors, and the translation vector;
- Store the .xml file.

Following the process completion, OpenCV for the python library is utilised to compute the results stored in the .xml file.

The black and white squared pattern match-finding is outlined as indicated in Figure 4.

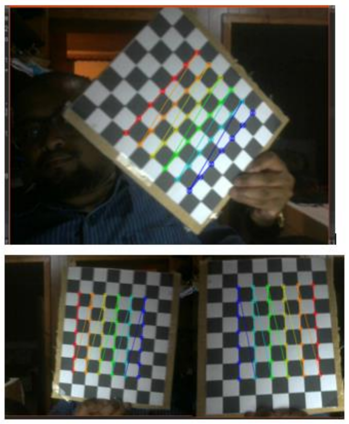
Figure 4. Black and white test match on a chessboard.

Figure 4 depicts the black and white match-finding test on a chessboard utilising an intrinsic matrix for the reduction of image biasing. This model was deemed less complicated in matching the images utilising the K d-tree approach. However, it was noted that CasHash based approach can have a fast computational time as compared to Kd-tree-based image matching depending on the number and size of the datasets. However, the Kd-tree-based approach was used since this project was based on few datasets.

D. *Application of structure of motion*

The basic operation of the structure of motion in this research follows the following pipeline:

- Detection of 2D features on every image;

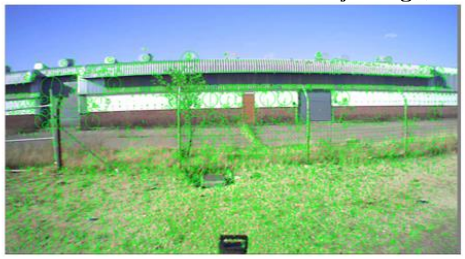
Figure 5. Detecting the image feature using SIFT algorithm.

Figure 5 depicts the captured image scene with the application of feature detection utilising the SIFT feature detection algorithms. This is obtained by creating an orientation histogram with 36 bins that cover 360° of the captured image. As a result, this creates the key points with the same location and scale but with different directions.

Table II depicts the SIFT feature detection algorithm that is utilised for the detection of the image scene outlined in Figure 5. These detected image features are detected from a .JPEG image format.

TABLE II. SIFT ALGORITHM FOR FEATURE DETECTION.

***Algorithm**: sift.detect()*
Function**: Begin()* *Import cv2* *Import numpy as np* *img = cv2.imread('cam1Image.JPG')* *gray=cv2.cvtColor(img,cv2.COLOR_BRG2GRAY)* *sift = cv2.SIFT()* *kP = sift.detect(gray, NONE)* *img = cv2.drawKeypoints(gray,kP)* *cv2.imwrite('sift_keypoints.jpg', img)* ***END

The use of the sift.detect() function is to find the keypoints, which are outlined in Figure 6. Furthermore, The cv2.drawKeyPoints() function is utilised to draw the small circles concurrently with the sift.detect() function . During this process, it was noted that the keypoint structure has many features for example the (X, Y) coordinates, size of neighbors, and keypoints strength.

- Matching of the 2D points between images;

Upon feature identification, the key points are matches by applying the Fast Library Approximate Nearest Neighbour (FLANN) as indicated in Figure 6.

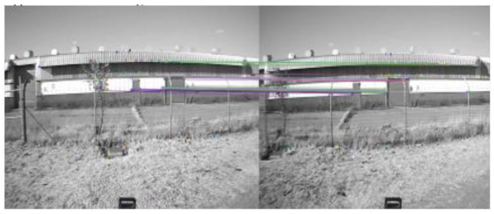
Figure 6. Matching of 2D points using the FLANN library.

During the image matching process, the flag is passed using the match flag function. The image match function in the context of this application was used with reference to the chessboard intrinsic matrix and was computed after the track construction was obtained as follows:

detIMG=cv2.drawKeyPoints(gray,kP,flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

cv2.imwrite (‘sift_keypoints.jpg’, detIMG)

In a normal circumstance mainly utilising exhaustive matching, the computation is usually time-consuming and complex. Hence, the need to utilise FLANN algorithm in this application. The FLANN algorithm utilised a tree-based approach by storing the image datasets within efficient data structures and utilising the Kd-tree approach. The Kd-tree approach was selected based on [13] [14].

- Construction of the tracks for the matches;

In track construction, the essential and fundamental matrices of a camera are computed. These matrices specify the camera motion in terms of rotational and translational components. This function is called in OpenCV to find the fundamental camera matrix as indicated:

cv2.findFundamentalMat(self.match_pts1,self.match_pts2, cv2.FM_RANSAC, 0.1, 0.99).

For essential matrix, the solve for structure library is called.

- Solving of the structure from motion from 2D tracks.

At this stage, the triangulation is performed for determining the point in a 3D space given its projection onto two or more images [15]. In a direct linear transformation, P3P, the algorithm is used for triangulation.

- Refine the SFM model using bundle adjustment algorithm;

At this stage, the Bundle Adjustment (BA) is performed. However, the problem relating to the BA is the simultaneous refining of the 3D coordinates describing the scene geometry. The parameters of the relative motion and the optical characteristics of the camera (s) are employed to attain the images. However, in this paper, the CERES algorithm is utilised.

IV. RESULTS

The technique of combining Street Views imagery with other known image datasets makes the experience of using panoramic view mainly in the virtual world even richer. The use of Street Views is essential as it is used as a tool to ease the livelihood with the ability to convert it into an application tool for directions while driving. In the process of acquiring such results, the feature detection and matching utilising the Structure From Motion (SFM) technique are important.

Another element outlined in Figure 7 depicts the point of cloud where all the feature tracks are visible in the image. The incremental SFM points are then used in the reconstruction of the scene and result in the formulation of the points of cloud. The output results are derived based on the camera position.

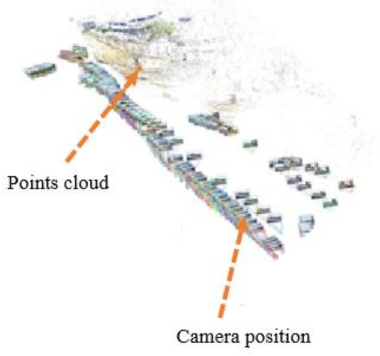

Figure 7. Point cloud and camera position reconstruction.

Figure 8 depicts the dense geometry reconstruction of the scene. This is achieved by performing the multi-view stereo algorithm. The application of the multi-view stereo produces the depth maps and dense point cloud and with this, the algorithm can perform the map recovery. Once this process is complete, a mesh of a scene is reconstructed by fusing depth maps and dense cloud as indicated in Figure 8.

Figure 8. Denser point of cloud with multiview stereo.

Following the creation of the mesh output, the texture was then generated by taking models, images, and the camera position (this was achieved using the GPS coordinates). Furthermore, meshroom function selecting 8192 texture slides was unwrapped.

The depth map restoration and colour images were paired up to create the application of texture to the 3D mesh model and warp to the new viewpoint. Furthermore, the extraction of the depth map model from the 3D mesh, and the shortcomings were observed to be the delay by which the mesh update duration is long and as a result, affects the depth map extraction.

The final output that is represented in Figure 9 depicts the rendered view of the panoramic street views. The depicted images are shown in a rendered street view from a top view perspective. This is achieved and projected from a horizontal street view in an omnidirectional manner.

These results are outlined to provide the freedom of movement and the views that were not captured by the camera but through rendering the uncaptured scene can be viewed as indicated in Figure 9.

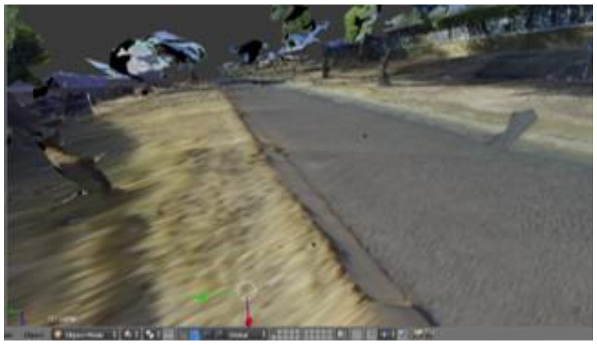
Figure 9. Rendered bidirectional 3D scenery view.

The panoramic street view can be viewed in multiple viewing angles as depicted in Figure 9. Furthermore, the depicted image outlines a rendered street view from a top view perspective with the bidirectional capability of scenes that we originally not captured.

V. Conclusion

The result and outcome of this paper demonstrate the application of the use of Street Views and image rendering in a real-live environment based on the hexagon camera configuration. Furthermore, it was seen that the IBR technique could easily allow for faster processing of multiple datasets based on the Kd-tree approach. Additionally, these techniques can allow for the movement and view of the uncaptured scene without the need for any extra application or tools.

References

[1] W. Wahbeh, S. Nebiker and G. Fangi, "Combining public domain and professional imagery for the accurate and dense 3D reconstruction of the destroyed bel temple in Palmyra," *ISPRS Annals of the Pthotogrammery. Remote Sensing and Spatial Information Sciences*, vol. 88, no. 81, p. 5, 2016.

[2] N. Bruno and R. Roncella, "Accuracy assessment of 3D models generated from Google Street View imagery," in *8th Intl. Workshop 3D-ARCH "3D Virtual Reconstruction and Visualization of Complex Architectures"*, Bergano, Italy, 2019.

[3] J. M. Frahm et al., "Building Rome on a cloudless day," pp. 368-381, 2010.

[4] X. Li and C. Ratti, "Mapping the spatial distribution of shade provision of street tree in Boston using Google Street View panoramas," pp. 109-119, 25 February 2018.

[5] P. E. Debevec, C. J. Taylor and J. Malik, "Modeling and Rendering architecture from photos: A Hybrid geometry- and image-based approach," *In ACM SIGGRAPH*, pp. 11-20, 1996.

[6] I. L. Lenkoe and B. Kotze, "Enhancing Spatial Image Datasets For Utilisation in A Simulation for Smart City Transport Navigation," in *Smart 2021: The Tenth International Conference on Smart Cities, Systems, Device and Technologies*, Valencia, Spain, 2021.

[7] Y. Mao, G. Cheung, A. Ortega and Y. Ji, "Expansion hole filling in depth-image-based rendering using graph-based interpolation," pp. 1859-1863, 2013.

[8] S. Shi, K. Nahrstedt and R. Campbell, "A real-time remote rendering system for interactive mobile graphics," *ACM Transactions on Multimedia Computing, Communications and Applications (TOMCCAP)*, vol. 8, no. 3, pp. 1-20, 2012.

[9] A. I. Audu, "Camera positioning for 3D panoramic image rendering," London, 2015.

[10] R. E. Klosterman, "The What if? Collaborative planning support system.," *Environment and Planning B: planning and Design*, pp. 393-408, 1999.

[11] D. Z. Sui, "GIS-based urban modeling: practices, problem and prospects.," *International Journal of Geographical information Science*, no. 12, pp. 651-671, 1998.

[12] E. H. Adelson and J. Bergen, "The plenoptic function and elements of early vision. In computational Model of Visual Processing," in *MIT Press*, Cambridge, MA, 1991.

[13] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz and R. Szeliski, "Building Rome in a day," vol. 54, no. 10, pp. 1-8, October 2009.

[14] D. Crandall, A. Owens, N. Snavely and D. P. Huttenlocher, "Discrete continuous optimisaton for large-scale structure from motion," pp. 3001-3008, July 2011.

[15] T. Moons, L. Van Gool and M. Vergauwen, "3D Reconstruction from multpile images: Part 1," vol. 4, no. 4, pp. 287-404, 2008.