Extended Abstract
Real-Time Visualization of Wake-Vortex Simulation using Computational Steering and Beowulf Clusters

Anirudh Modi[1], Lyle N. Long[2], and
Paul E. Plassmann[3]

[1] Ph.D. Candidate, Department of Computer Science and Engineering,
Pennsylvania State University, University Park, PA 16802
anirudh@anirudh.net
http://www.anirudh.net/phd/
[2] Professor, Department of Aerospace Engineering,
Pennsylvania State University, University Park, PA 16802
lnl@psu.edu
http://www.personal.psu.edu/lnl/
[3] Associate Professor, Department of Computer Science and Engineering,
Pennsylvania State University, University Park, PA 16802
plassman@cse.psu.edu
http://www.cse.psu.edu/~plassman/

Parallel simulations are playing an increasingly important role in all areas of science and engineering. As the applications for these simulations expand, the demand for their flexibility and utility grows. Interactive computational steering is one way to increase the utility of these high-performance simulations, as they facilitate the process of scientific discovery by allowing the scientists to interact with their data. On yet another front, the rapidly increasing power of computers and hardware rendering systems has motivated the creation of visually rich and perceptually realistic virtual environment (VE) applications. The combination of the two provides one of the most realistic and powerful simulation tools available to the scientific community.

The computational steering system we have developed is very general in nature and is based on a simple client/server model. It is designed to be extremely lightweight, highly portable (runs on all Win32 and Unix platforms), fast and simple to use (using C++ classes, templates and polymorphism) making it very easy to augment any existing C/C++ simulation code in a matter of hours. It deals with byte-ordering and byte-alignment problems internally and also provides an easy way to handle user-defined classes and data structures. It is also multi-threaded in nature, supporting several clients simultaneously. It can also be incorporated with equal ease into parallel simulations running on Beowulf clusters. As a particular application of this computational steering system, this work aims at integrating a parallel wake-vortex simulation code with our Virtual Reality (VR) facility for visualization.

National Aeronautics and Space Administration (NASA) scientists predict that by the year 2022, three times as many people will travel by air as they do today [1]. To keep the number of new airports and runways to a minimum, there is an urgent need to increase their efficiency while reducing the aircraft accident rate. Moreover, according to the predictions by the United States Federal Aviation Administration (FAA), by the

year 2015, if the U.S. air transportation system does not change in any significant way, there could be a major aviation accident every seven to ten days. To greatly reduce the likelihood of this disturbing scenario, the FAA has declared that the existing Air-Traffic Control (ATC) system will shift to a new system soon, and is seriously evaluating the likelihood of having a VE based system to assist the ATC in its tasks.

One of the main problems daunting the ATC today is the "wake-vortex" hazard. Just as a moving boat or a ship leaves behind a wake in the water, an aircraft leaves behind a wake in the air (Fig 1). The aircraft wake is generated from the wings of the aircraft and consists of two counter-rotating swirling rolls of air which are termed as the "wake-vortex" or "wake-vortices". These wake-vortex pairs are invisible to naked eye and stretch for several miles behind the aircraft and may last for several minutes. The strength of these vortices depends on several factors, prominent amongst which are the weight, size and speed of the aircraft. The strength generally increases with the weight of the aircraft. The life of the vortex depends on the prevailing weather conditions. Typically, vortices last longer in calm air and shorter in the presence of atmospheric turbulence. Study of these vortices is very important for aircraft safety [2]. The rapid swirling of air in a vortex can have a potentially fatal effect on the stability of a following aircraft. Currently, the only way to deal with this problem is the use of extremely conservative empirical spacing between consecutive take-offs and landings from the same runway, which has been laid down by the International Civil Aviation Organization (ICAO). In instrument flying conditions, aircraft may follow no closer than three nautical miles, and a small aircraft must follow at least six nautical miles behind a heavy jet such as a Boeing 747. But, inspite of these spacings being extremely conservative, they are not always able to prevent accidents owing to the several unknowns involved, primarily the exact location and strength of the vortices. The infamous US Air 427 (Boeing 737) disaster which occurred on September 8, 1997 is attributed to this phenomenon, wherein the aircraft encountered the wake-vortices of a preceding Boeing 727 [3]. The more recent Airbus crash on November 12, 2001 is also believed to be, atleast partially, a result of wake-vortex encounter from a preceding Boeing 747.

To tackle this problem of reduced airport capacity which is a direct fallout of these overly conservative ICAO spacing regulations, and to address the concerns of the aircraft in circumstances when these regulations fail to meet safety requirements, the NASA researchers have designed a system to predict aircraft wake-vortices on final approach, so that the aircraft can be spaced more safely and efficiently. This technology is termed AVOSS or Aircraft VOrtex Spacing System (AVOSS) [4]. AVOSS, in spite of carrying out a rigorous simulation of the wake-vortices, does not implement any system for their visualization. It only provides the ATC with the aircraft spacing time for each aircraft which is all the current ATC systems can handle. Thus, at present, it is unable to provide alternate trajectories for the take-off and landing of aircraft.

This work attempts to fill in the gaps left by AVOSS by creating a wake-vortex hazard avoidance system by realistically simulating an airport with real-time 3D visualization of the predicted wake-vortices generated by the moving aircraft. Such a system has the potential to greatly increase the utilization of airports while reducing the risks of possible accidents. Aircraft will be able to adjust their flight trajectory based on the

Fig. 1. B-727 vortex study photo (Courtesy: NASA Dryden Flight Research Center)

information obtained from this VE system to avoid the wake-vortices and operate more safely and efficiently.

Since the wake-vortex prediction for an entire fleet of aircraft taking-off and landing at a busy airport is a computationally intensive problem, a parallel solution for the same is employed. A typical metropolitan airport in the US is extremely busy with several take-offs and landings occurring every few minutes. Dallas/Fort Worth, the country's third busiest airport, has seven runways that handle nearly 2300 take-offs and landings every day. For the wake-vortex code to track the vortices shed by an aircraft for 5 miles after take-off, assuming that a vortex core is stored every 5 meters, $5 \times 1600/5 \times 2 = 3200$ vortex filaments have to be tracked. For 2300 take-offs and landings every day, it implies that $3200 \times 2300/24 = 306667$ vortex filaments have to be tracked every hour. Since, the average vortex does not decay before 15 minutes, vortices due to typically half the take-offs and landings every hour need to be tracked at any given time. This amounts to roughly 153333 vortex filaments. While this may not seem to be a very large number on its own, the problem gets complicated by the presence of a double loop for the calculation of the induced velocity of every vortex on every other vortex. Even if the induced velocity effect due to vortices from the other aircraft are ignored, this still amounts to as much as $3200 \times 3200 = 10.24$ million computations for each airplane at every timestep. For 2300 planes/day, this comes out to $10.24 \times 2300/24/2 = 490.7$ million calculations per timestep for the induced velocity, a very large number indeed for a conventional uniprocessor system. And with each timestep being, say 0.1 seconds, this amounts to 4.9 billion calculations per second. Although this number can be reduced by as much as a factor of 100 by making simplifying assumptions for the induced velocity calculations (wherein, we say that any vortex element is only affected by a fixed number of neighboring elements, say k, rather than all the other elements), this still amounts to a large computation considering that each

induced velocity calculation consists of 200-300 floating point operations. This takes our net computational requirement to approximately 10-15 Gigaflops, necessitating the need of a parallel computer. Hence, our wake-vortex prediction code, which is based on the potential flow theory [5], is written in C++ with Message Passing Interface (MPI) for parallelization. The code has been parallelized to track vortex elements from each aircraft on a different processor in such a way that we get an almost real-time solution to this problem with tolerable lag. The output from the parallel code is then sent to the VR system via a computational steering framework.

For the computational steering, an approach similar to that used in *Falcon* [6] (an on-line monitoring and steering toolkit developed at Georgia Tech) and ALICE memory snooper [7] (an application programming interface designed to help in writing computational steering, monitoring and debugging tools developed at Argonne National Lab) has been used. It consists of a steering server on the target machine that performs steering, and a steering client that provides the user interface and control facilities remotely. The steering server is created as a separate execution thread of the application to which local monitors forward only those "registered" data that are of interest to steering activities. The steering client receives the application run-time information from the application, displays the information to the user, accepts steering commands from the user, and enacts changes that affect the application's execution. Communication between the steering client and server are done using UNIX sockets. To make things simpler, the computational steering library has been written in C++, using several of C++'s advanced object-oriented features, making it very easy and powerful to use while hiding most of the complexities from the user. The library allows a simulation running on any parallel or serial computer to be monitored and steered remotely from any machine on the network using a simple cross-platform client utility.

For the real-time simulation, the parallel wake-vortex code has been augmented with the computational steering library, so that it can remotely run on our in-house 40 processor PIII-800 Mhz Beowulf Cluster, the COst-effective COmputing Array 2 (COCOA-2) [8]. For the steering client, a visualization tool has been written in C++ using the OpenGL application programming interface (API) for graphics and CAVELib [9] API for stereographics and user interaction. A screenshot of the program depicting several aircraft and their generated wakes is shown in Figure 2. The colors represent the relative strength of the vortices with red being maximum and blue being minimum. The Reconfigurable Automatic Virtual Environment (RAVE) from FakeSpace Systems driven by an HP Visualize J-class workstation is then used as the display device.

The coupling of computational steering of our parallel simulation with VR setup provides us with near real-time visualization of the wake-vortex data in stereoscopic mode. It opens a new way for the ATC to effectively deal with the wake-vortex hazard problem and to improve the capacity and safety of large airports. At a more basic level, this ability to get "immersed" in the complex solution as it unfolds using the depth cue of the stereoscopic display and the real-time nature of the computational steering system opens a whole new dimension to the scientists for interacting with their simulations.

References

1. Hypotenuse Research Triangle Institute. Wake Vortex Detection System: Engineered for Ef-

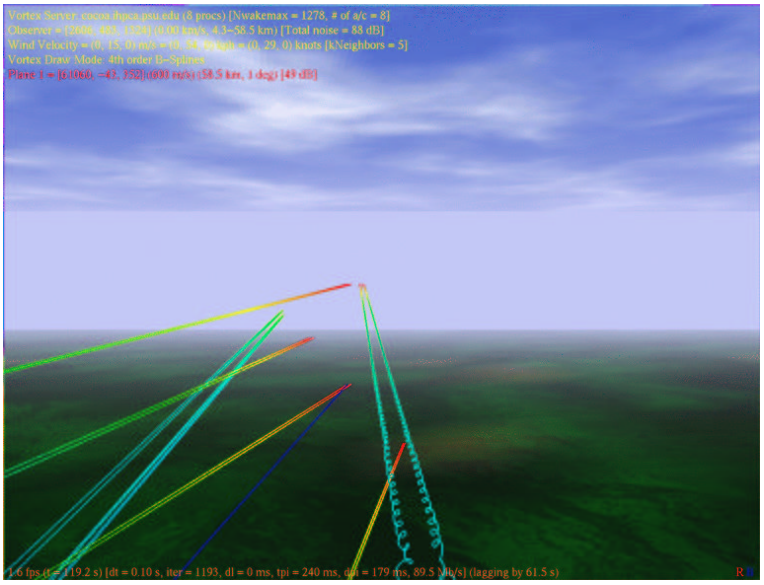

Fig. 2. Screenshot of the current Wake-Vortex simulation

ficiency and Safety. http://www.rti.org/hypo_etc/winter00/vortex.cfm, 2001.
2. Barnes W. McCormick. Aircraft Wakes: A Survey of the Problem. *Keystone Presentation at FAA Symposium on Aviation Turbulence*, March 1971.
3. Airdisaster.com. Investigation: USAir Flight 427. http://www.airdisaster.com/investigations/us427/usair427.shtml, 1997.
4. D. A. Hinton. Aircraft Vortex Spacing System (AVOSS) Conceptual Design. *NASA TM-110184*, August 1995.
5. Fred H. Proctor and George F. Switzer. Numerical Simulation of Aircraft Trailing Vortices. *Ninth Conference on Aviation, Range and Aerospace Meteorology*, September 2000.
6. W. Gu, G. Eisenhauer, E. Kraemer, K. Schwan, J. Stasko, J. Vetter, and N. Mallavarupu. Falcon: On-line Monitoring and Steering of Large-Scale Parallel Programs. *Proceedings of the Fifth Symposium on the Frontiers of Massively Parallel Computation*, pages 433–429, February 1995.
7. I. Ba, C. Malon, and B. Smith. Design of the ALICE Memory Snooper. http://www.mcs.anl.gov/ams, 1999.
8. Anirudh Modi. COst effective COmputing Array-2. http://cocoa2.ihpca.psu.edu, 2001.
9. VRCO. CAVELib Users Manual. http://www.vrco.com/CAVE_USER/caveuser_program.html, 2001.