9

Making New Reality: Computers in Simulations and Image Processing

The computers discussed so far actually flew in space or worked in direct support of launches and missions. Yet NASA found numerous uses for computers in areas somewhat removed from flight operations. Chief among these are simulations and image processing, which made the training of crews, development of launchers and spacecraft, and analysis of image data possible.

Simulations are used in hundreds of ways in the space program. Simulation programs and hardware test the workings of vehicles and spacecraft, determine the accuracy of flight paths, train controllers, check out designs, and actively contribute to the software development process. Simulations help NASA find out whether its programs and projects will work as planned, lessening the risks for crews and equipment. Especially important are simulations used in crew training and simulations used to test hardware. Both provide models by which to judge the extent and efficacy of NASA's dependence on simulations and to demonstrate the dependency of such simulations on computers.

Image processing was developed to make the analysis of digital images transmitted by unmanned deep space craft more consistent and fruitful. At first largely driven by the needs of the Jet Propulsion Laboratory's (JPL) scientific community, imaging spread quickly with applications such as Landsat and the Shuttle's imaging radar. From spectacular images of distant worlds to detailed pictures of the neighbor's farm, imaging technology has contributed to the quality of life on earth. Without the use of high-speed computers, the analysis and use of the billions of bits of imaging data would be impossible.

CREW-TRAINING SIMULATORS

NASA's requirements for flight simulators far exceeded the state of the art when the first astronaut crews reported for duty in 1959. Feeling obligated to prepare the astronauts for every possible contingency, NASA required hundreds of training hours in high fidelity simulators. Each crewman in the Mercury, Gemini, and Apollo programs spent one third or more of his total training time in simulators. Lunar landing crews used simulators more than half the time[1].

Simulators must provide the astronaut trainee with as close an approximation of spaceflight as is possible on earth, without losing sight of the need to extensively practice procedures to respond to failures as well as nominal events. Requirements for realism increase the complexity of the simulation. For example, when an astronaut fires thrusters, the simulator must activate readouts and lights showing the thrusters firing, fuel reducing, velocity changes, and also show movement in the scene outside the cabin window. In a moving base simulator, such as a simulator in which a spacecraft cabin is suspended

on hydraulically moved pylons to enable it to tilt, physical motion must take place. Causing all these things to happen and coordinating them to happen simultaneously is the difficult task of the simulator designer[2].

A manned spaceflight program always had more than one type of simulator. Usually there was a pair of full-function simulators, one fixed base and one moving base, used for procedures training and extended simulations. Often, part task trainers were needed for more difficult mission phases as well. NASA built a simulator for the last 200 feet of the landing of the lunar module. One part-task trainer exists to train astronauts in using the Shuttle on-board computer System and its software.

Mission simulators today are so dependent on computers that it has become necessary for proper design to think of it as large data processing complex that incidentally is driving displays and performing other functions in a crew trainer[3]. For the Shuttle Mission Simulator several dozen mainframe, mini, and on-board computers interconnect to create window scenes, change displays, move indicators, and light event lights. Reaching this level of computer involvement resulted from a steady evolution since the beginnings of manned spaceflight.

Project Mercury Mission Simulators

When the time came to design the Mercury flight simulators, experience with aircraft simulators and with those built for the X-15 rocket plane were all that were available. There is one critical difference between training needs for test pilots of aircraft and those of astronauts. Although flying experimental aircraft is always dangerous they are rarely taken to their projected limits on the first flight[4]. Even the X-15 had a long series of buildup missions, first with a smaller engine, later incrementally increasing speed, then altitude, until a series of full out flights sent the plane to the edge of space. In rocket flight the spacecraft is pushed to the outer limits of stress and endurance from the instant of ignition. Its crews must be fully prepared for all contingencies before the first flight and continue to be prepared for every flight afterwards.

The primary simulator for the first manned spacecraft was the Mercury Procedures Simulator (MPS), of which two existed. One was at Langley Space Flight Center, and the other at the Mission Control Center at Cape Canaveral. Analog computers calculated the equations of motion for these simulators, providing signals for the cockpit displays[5]. In addition to this primary trainer, a centrifuge at the U.S. Naval Air Development Center in Johnsville, Pennsylvania, served as a moving-base simulator. A Mercury capsule mock-up mounted

at the end of the centrifuge arm provided ascent and entry training[6]. Additionally, Langley built a free-attitude trainer that simulated the attitude control capabilities of the spacecraft and two part-task trainers for retrofire and entry practice.

Analog computers commonly supported simulation in the 1950s and early 1960s. Having the advantage of great speed, the electronic analog computer fit well into the then analog world of the aircraft cockpit and its displays. By 1961, though, it became obvious that the simulation of a complete orbital mission would be impossible using only analog techniques[7]. The types and number of inputs and calculations stretched the capabilities of such machines so that when NASA defined requirements for Gemini simulators, digital computers dominated the design.

Computers in the Gemini Mission Simulators

Training crews for the more complicated Gemini spacecraft and its proportionately more complicated missions required the use of digital computers in the simulators. Aside from the tasks done during Mercury, such as ascent, attitude control, and entry, the Gemini project added rendezvous and controlled entries utilizing the spacecraft's greater maneuvering capabilities. At the Manned Spacecraft Center, NASA installed simulators to provide training for these maneuvers, including a moving-base simulator for formation flying and docking and a second moving-base simulator for launch, aborts, and entry. Besides these, two copies of the primary Gemini Mission Simulator, which had the same purpose as the Mercury Procedures Simulator, and the Johnsville centrifuge completed the list of Gemini trainers. One of the Mission Simulators was at Cape Canaveral; the other at Houston.

Gemini Mission Simulators used between 1963 and 1966 operated on a mix of analog and digital data and thus are a transition between the nearly all analog Mercury equipment and the nearly all digital Apollo and later equipment. Three DDP-224 digital computers dominated the data processing tasks in the Mission Simulator. Built by Computer Control Corporation, which was later absorbed by Honeywell Corporation, the three computers provided the simulator with display signals, a functional simulation of the activities of the onboard computer, and signals to control the scene generators[8].

Functional simulation of various components was made easier by the use of digital computers. In a functional simulation, the actual component is not actually located in the simulator, its activities and outputs being created by software within the computer. Thus, in the Gemini Simulator, the on-board computer was not installed, but the algorithms used in its programs were resident in the DDP computers,

and when executed, activated computer displays such as the incremental velocity indicator just as on the real spacecraft.

Scene depiction in the Gemini era still depended on the use of television cameras and fake "spacescapes", as in aircraft simulators. Models or large photographs of the earth from space provided scenes that were picked up by a television camera on a moving mount. Signals from the computers moved the camera, thus changing the scene visible from the spacecraft "windows," actually CRTs. A planetarium type of projection was also used on one of the moving-base simulators at Johnson Space Center to project stars, horizon, and target vehicles.

Gemini simulations often included the Mission Control Center and worldwide tracking network. No commercially available computer could keep up with the data flowing to and from the network during these integrated simulations, so NASA asked the General Precision group of the Link Division of Singer Corporation to construct a special-purpose computer as an interface[9]. Singer held the contract for the simulators under the direction of prime contractor McDonnell-Douglas, which supplied cabin and instrumentation mock-ups. Fully functional simulators came on line at the Cape and Houston during 1964.

Moving-base simulation came into its own during the Gemini program. The docking simulator was in a large rectangular cube that permitted great freedom of motion in training crews for station keeping and docking. The dynamic crew procedures simulator that replicated launch, abort, rendezvous, tethered (with the Agena upper stage), and entry maneuvers and procedures suggested the feeling of acceleration at lift-off by tilting the spacecraft at a rate equal to the g buildup during launch from about a 45-degree angle to nearly horizontal to the floor. This resulted in a push on the astronaut's back, which increased from $0.707g$ to $1g$. Engine cutoff and weightless flight could be suggested by returning the spacecraft to its original position, giving a feeling of maximum comfort to the crew[10]. Negative gs could be simulated by tilting the nose down, causing the astronauts to feel their weight on their shoulder harnesses.

Designing and using the Gemini simulators gave NASA a lot of experience in producing high fidelity simulations. Actual flight experiences from Mercury went into improving the Gemini simulators. Gemini rendezvous and maneuver experience helped make the Apollo simulations better. NASA adopted some of the Gemini equipment for Apollo. The use of Honeywell's DDP-224 computers continued, while moving-base simulators were adapted to Apollo use by changing the spacecraft mock-up and modifying existing techniques[11]. Still, the Apollo program requirements demanded a further increase in the amount of computer power.

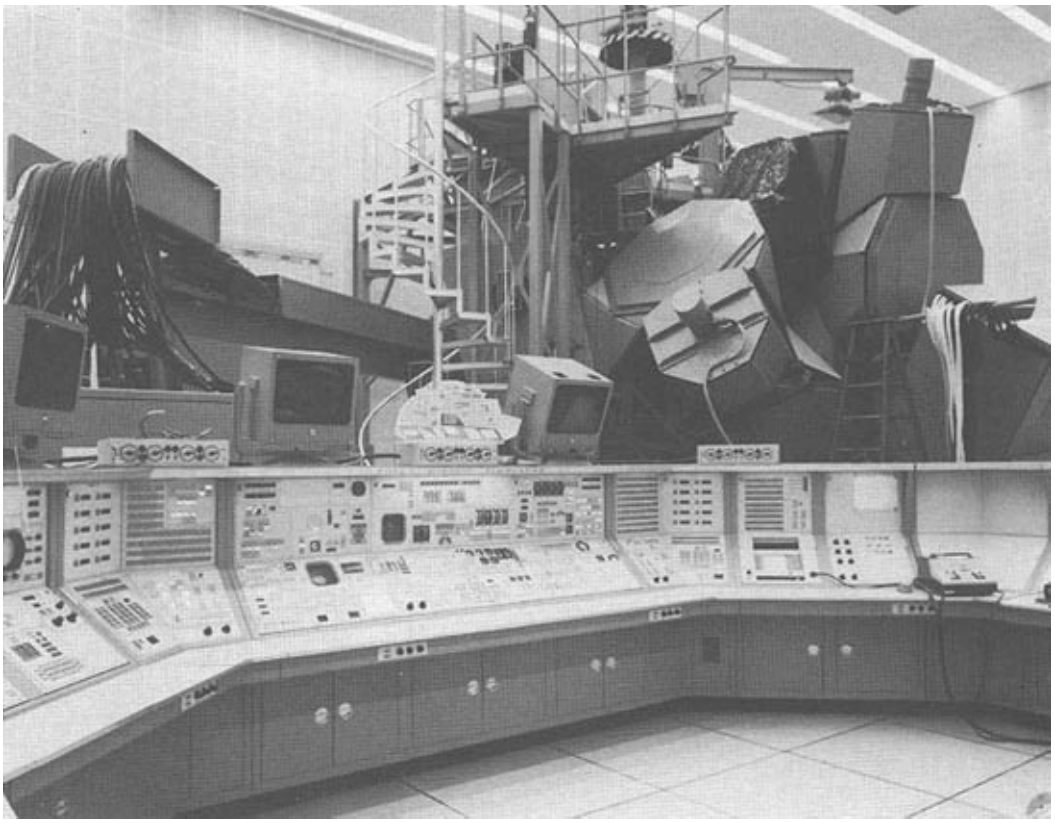


Figure 9–1A. The Apollo Command Module Mission Simulator. (NASA photo 108-KSC-67PC-178)

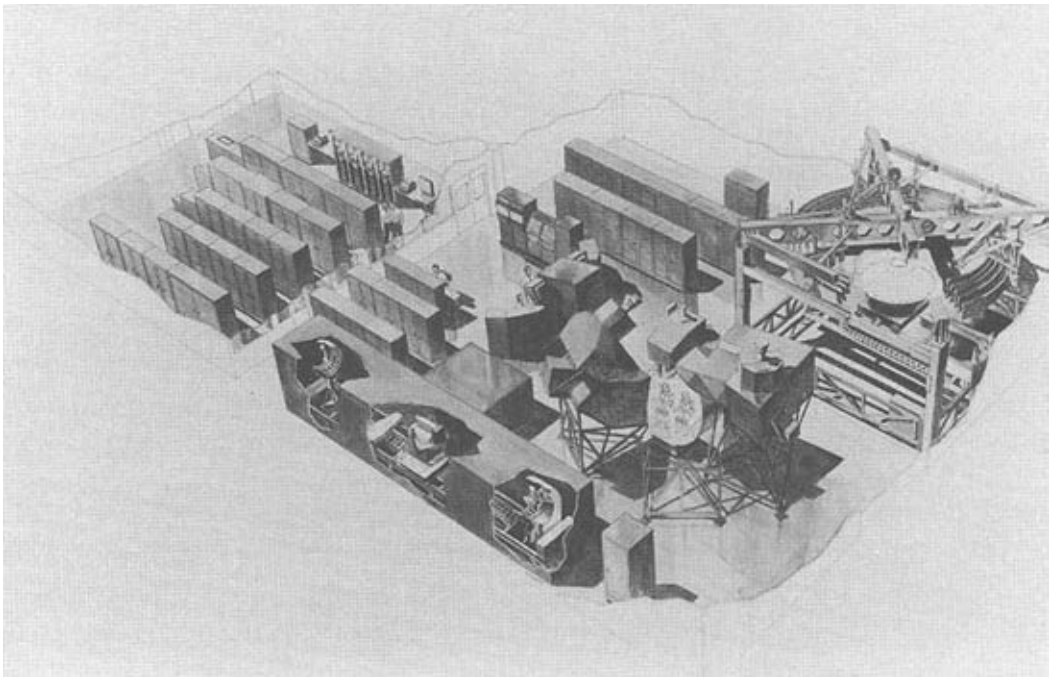


Figure 9–1B. An artist's conception of the Apollo Lunar Mission Simulator.

Computers in the Apollo Mission Simulators

No less than 15 simulators trained crews during the Apollo Program. Three were the primary Command Module Simulators, with one at Houston and a pair at the Cape. Two were the primary Lunar Module Simulators, one at each site. At Houston, a Command Module Procedures Simulator trained crews just to rendezvous with the command module, as there was a Lunar Module Procedures Simulator for lunar module rendezvous and landing training. Gemini's Dynamic Crew Procedures Simulator became the same for Apollo. Additional moving-base simulators at the Manned Spacecraft Center were for lunar module formation flying and docking, and a centrifuge (to avoid trips to Johnsville). Langley Space Flight Center pioneered the research into the final 200 feet of lunar landing by suspending five sixths of a simulator's weight to give astronauts practice in controlling the lander in the gravity of the moon[12]. Another lunar landing simulator used a jet engine to support five sixths of its weight and permit free-flight landing training. That simulator required a simulator of its own to keep the crews from crashing it. Finally, a pair of partial-gravity simulators gave the astronauts the chance to walk in space suits while having five-sixths of their weight supported. Later in the program, Marshall Space Flight Center built a simulator for the lunar rover vehicle.

Among the plethora of simulators, use of the Command Module Simulators and Lunar Module Simulators nonetheless occupied 80% of the Apollo training time of 29,967 hours[13]. These simulators and their associated computer systems were crucial to the success of the program. The Apollo 13 emergency in April 1970, when there was an explosion in the service module on the way to the moon, demonstrated the high fidelity and flexibility of the simulators as all lunar module engine burns, separations, and maneuvers could be tested and ad hoc procedures developed as the crippled mission progressed.

In contrast to the procedures simulators, all of which were driven by a single mainframe computer, the Mission Simulators used networks of several computers[14]. Honeywell won a $4.2 million contract on July 21, 1966 to supply DDP-224 computers for the complexes[15]. Singer–Link was again the contractor for the simulators. Singer allocated three computers for the Command Module Mission Simulator and two for the Lunar Module Simulator. The sets of computers could communicate among themselves by using 8K words of common memory, where information needed throughout the simulation could be stored[16]. Later, a third and fourth computer were added, respectively, to the Lunar and Command Module Simulators. These computers

simulated the on-board computers. By the Apollo 10 flight a fifth computer, simulating the launch vehicle, completed the Command Module Simulator computer complex[17]. The two types of simulators and the Mission Control Center could do integrated simulations, thus requiring up to 10 digital computers to be working on one large problem simultaneously[18].

Software became as important to the simulated world of Apollo as it was in the real world. Software development for the Apollo Mission Simulators required the efforts of 175 programmers at the peak, compared to 200 hardware persons[19]. Over 350,000 words of programs and data eventually ran in the two simulators. Using digital computers, trainers could return the crews to a certain point in a simulation and try again by simply recording the status of the computers and data on magnetic tape and reloading memory to match the state of the software at the time desired. This sort of flexibility made the training task much easier.

Early in the development of the Apollo simulators, a problem arose that would have had critical consequences if not solved. The importance of the on-board computer to the guidance and navigation of a moon-bound spacecraft was obvious. Crews interacted with the computer thousands of times in a typical mission; its keyboards contained the most used switches in the spacecraft. Initially, the Apollo Guidance Computer (AGC) for both the command module and the lunar module were simulated functionally, just like the rest of the spacecraft hardware[20]. This meant that the major components of the Apollo modules existed as software in a DDP-224 rather than in their physical form in the simulator.

Even so, functionally simulating the on-board computer soon proved to be nearly impossible. Mathematical models and algorithms for specific Apollo missions had to be sent to the simulator programmers from the Instrumentation Laboratory at MIT. Although Singer contracted for over 20 experienced IBM programmers, the development of functional simulations lagged[21]. The programmers had to take logic and create software for the DDP-224s that executed just like the software on the AGC. Essentially they coded programs already being coded for the real computer but in a different machine language. Warren J. North of the Computational Analysis Division at the Manned Spacecraft Center studied the process of creating the new software and found it took about 4 months to write the functional simulation. Since crews needed the software for training at least 6 months before the mission, and some buffer had to be allowed for last-minute glitches and their solutions, software designs for the AGC, developed at MIT, had to be available a full year before a flight, a very difficult schedule to meet at the time[22]. As a result of this study and the continued concern of the Apollo Spacecraft Project Office, W. B. Goeckler of the Systems Engineering Division of the program asked James L. Raney of

Computational Analysis to do a feasibility study of using a DDP-224 to simulate the AGC[23]. Goeckler thought it might be possible to make the Honeywell computer think it was the MIT computer and execute the MIT code, thus eliminating the need for rewriting the programs and solving the time problem.

When Raney joined Apollo in February of 1966, he faced a rather interesting question: Could a floating-point arithmetic, two's complement representation, 24-bit computer with accumulators and index registers run programs written for a fixed-point, one's complement representation, 16-bit machine that buried its registers in memory? Hardly likely, just about everybody thought—except Raney.

Instead of a functional simulation, a computer running another computer's code uses interpretive simulation techniques. It takes a single instruction from the other's program, executes it using as many instructions as necessary from its own repertoire, and then goes on to the next. Since the AGC had a unique interrupt structure and limited arithmetic capabilities (limited compared with the DDP-224), many Apollo instructions took multiple Honeywell instructions to get around the differences.

Raney suggested both hardware and software modifications to the DDP-224. He specified a switch to disable the machine's floating-point capability. Instructions were added to enable more efficient table searching and other operations that the AGC did well. To handle the different word sizes, Raney let the right-most 14 bits of the DDP word be the value of a corresponding AGC word. The left-most bit was always set to zero to indicate that it was an Apollo word, and the intervening bits matched the sign bit of the original word. Words that could not be translated (i.e., executed one for one), had to be executed by interpretive subroutines written for the purpose and stored in the lower part of the Honeywell memory. Raney figured that since the DDP had a 10-to-1 advantage in execution speed over the AGC, several instructions could be used to do one Apollo instruction without slowing down the program. He used the index registers in the Honeywell DDP to act as the Fixed Bank Register, which kept track of which core rope memory module the AGC was currently using, as well as the address of the next instruction. Finally, to store the AGC code, the flight program was put in the upper half of the 64K words of core, with the interpreter used in the AGC to execute its own instructions in an area in lower core. The contents of the AGC's 2K erasable memory and the 8K of common core addressable by all the simulator computers also was in lower core, along with Raney's interpretive subroutines[24].

Despite Raney's careful evaluation of the situation and proposed solution, many Apollo project personnel opposed it, simply feeling it was unworkable[25]. In desperation, NASA approved the attempt at an interpretive simulator and bought the modified computers. In the end, the simulation within a simulation was spectacularly successful.

Even though Raney and his team took care to time the subroutines so that they matched execution of the actual Apollo code, the simulated computer was faster than the real article. Following the Apollo 9 earth-orbiting mission that tested the command module and lunar module rendezvous techniques, pilot Dave Scott complained that he had up to 12 seconds less time to react when the computer signaled for a maneuver to begin. This was adjusted for later flights.

Developing the interpretively simulated AGC had several impacts on the program. MIT could use the simulator as a field test of its code before flight. Since MIT used tape rather than core rope to send the programs to Houston and the Cape, errors discovered could be corrected and then the corrections tested in a "real" situation. Crews could react to the way the software worked with them. Also, the simulator cost just $4.6 million, compared to an estimated $18 million for functionally simulating the programs.

Actually, the Apollo Mission Simulators were the last of their type in that the analog environment of the spacecraft that dictated hybrid and functional simulations changed to a digital environment that lent itself to full digital simulations for the Shuttle program. Evolution to full digital simulation, including digital imaging of window scenes, meant even more dependence on digital computers. Making the Shuttle a more autonomous and thus more complex spacecraft contributed to a massive increase in the size of the computer systems needed to support simulations.

Full Digital Reality: Computers in the Shuttle Mission Simulators

The difficulty of producing a fully digital simulation of the Shuttle may be appreciated by considering the fact that when NASA issued the first request for proposals for the Mission Simulators, there was no response[26]. Singer, which by then had converted Precision Link to the Simulator Products Division, eventually responded with a plan for a detailed analysis of the simulation problems of the Shuttle. NASA had already decided that the extreme cost of developing Shuttle simulators would be moderated by acquiring fewer of them[27]. Shuttle program director Robert F. Thompson formed a committee in 1970 to monitor development of the simulators and involve the projected users, the Flight Crew Operations Division, and the Flight Operations Division, in its design[28]. Singer considered the requirements and suggested a large complex of mainframe computers functioning through limited task minicomputers to drive the simulator.

All Shuttle simulators are located at the Johnson Space Center. The fixed-base simulator replicates the four crew stations on the flight

Figure 9–2. The fixed base Shuttle Simulator (upper center) with some of its electronics. (NASA photo S-81-27526)

deck of the orbiter. It has window views through both aft windows and the overhead windows. Hosted by four Sperry Corporation UNIVAC 1100/40 mainframe computers, 15 Perkin-Elmer minicomputers (mostly 8/32s) provide digital images for the windows, interface with the on-board computers, and perform other functions, acting as fancy channel directors for the mainframes[29]. A motion-base simulator recreates the two forward crew stations, all forward window views, and the heads-up display used in landing. Also hosted by four 1100s, it has 11 minicomputers due to the lesser digital image requirements. The fixed-base simulator not only has to display proper images of the earth and the cargo bay but it also must image the remote manipulator arm and any payloads, thus requiring the power of five of the 8/32s. Supplementing the two primary Mission Simulators is the Shuttle Procedures Simulator. Also called the "Spare Parts Simulator," it was often cannibalized to keep the more critical Mission Simulators running[30]. In the early 1980s it was scrapped, and a Guidance and Navigation Simulator was built out of its remaining parts. It is used for some part-task training.

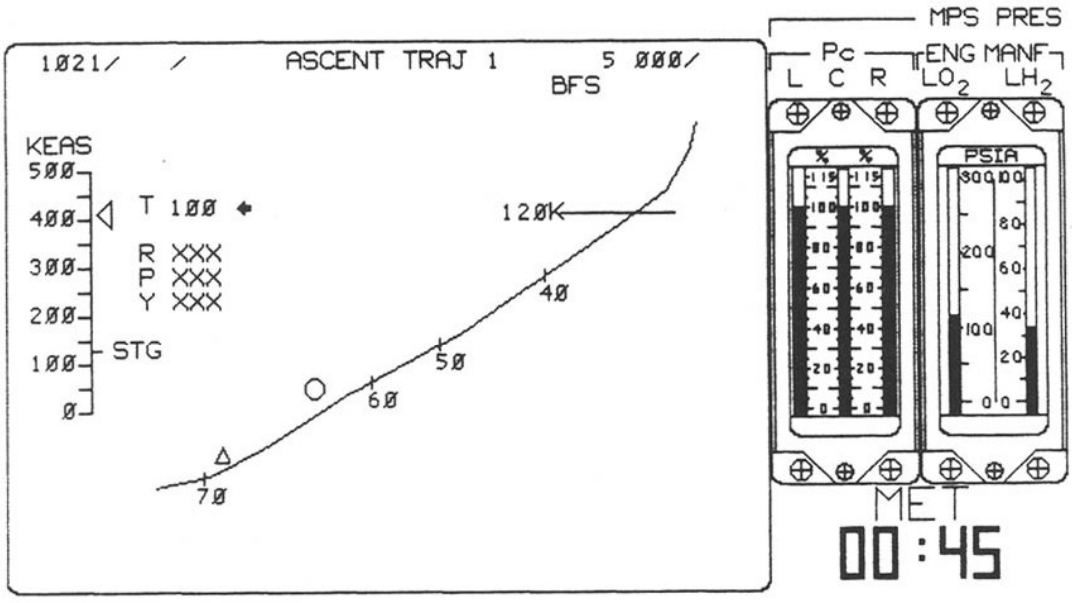
Singer quickly decided that the Shuttle's on-board computers could not be interpretively simulated, as the AGC was[31]. IBM's AP-101 machines used on the spacecraft were roughly as fast as the UNIVAC

computers, eliminating the time advantage the DDP machines had over the AGC[32]. Functional simulation of five computers working in concert was also out of the question. Therefore, each simulator had five computers, just as in the real spacecraft, and NASA bought two more as spares. During the course of the program, however, the computers began failing. With training schedules calling for simulation runs of 16 hours a day plus maintenance and reloading, several computers reached 30,000 hours of operation, far greater than the operational life of the flight version. Roughly 12 or 13 are actually available at any one time, with the two primary simulators always kept at a full complement of five, and the Spare Parts Simulator using the rest[33]. The mass memory unit (MMU) of the on-board computer system, the magnetic tape drive that stores the software, is functionally simulated. It proved impossible to keep the actual mass memories running long enough to be cost effective. Designed for only a few minutes of use in each flight, they fell apart under the demands of the simulators. A disk drive controlled by an IBM Series/1 processor replaced the MMU, with delays built in to make it load as slowly as a tape would.

Software for the Shuttle Mission Simulators is based on a 20-millisecond cycle controlled by a special real-time clock that sends a signal to all participating computer systems[34]. This is about the only way the large number of computers can be kept in step. The operating system for the UNIVAC machines is a commercial version that is no longer supported by Sperry, so NASA has had to specifically contract for maintenance on the system to avoid having to change the rest of the software to match a new one[35]. Singer wrote the real-time operating system used on the Perkin-Elmer machines[36]. Despite the large number of programmers on Singer's Shuttle Simulator payroll (200+ of 611 people), it subcontracts with Perkin-Elmer for some software, creating a situation where the developers are removed from NASA managers by another layer of management, which has resulted in unsatisfactory products[37]. In 1980, NASA's Robert Ernull, with years of experience in the on-board software division, was named head of the simulator division to help clear up problems with the complex simulators. He tried to reduce the throughput required of the computers to 70% of the total capability to allow for changes. This did not help what he thought was a second major problem—lack of memory. Memories were so full any modifications caused a crisis[38].

Aside from the more traditional Mission Simulators, NASA is beginning to use microcomputers to replace the expensive part-task trainers of the past. A system called Regency provides a programmable 64 by 64 spot touch screen. Detailed graphics of switches and indicators can be displayed, and also component schematics, so that trainees can communicate with the teaching software by touching the screen in the appropriate place. The teaching software is based on techniques developed for the PLATO system at the University of

Illinois. Increased use of microcomputers and other small computers for more generalized training will come as the space program enters the Space Station era. Simulating large spacecraft will be financially impossible, but simulation of critical crew stations using software and graphics for flexibility will be possible. Given the present direction, it appears that some sort of generic trainer with its characteristics controlled by software will be the mainstay of the training program, replacing the large computer complexes of the past and present.



THRUST BUCKET

Thrust bucket is a term used to describe a temporary reduction in thrust for aerodynamic loading. The thrust level changes are controlled by engine valves responding to GPC guidance programs. Orbiter flight requirements will determine the time, duration, and thrust levels.

Percent of thrust (T) is displayed on the BFS TRAJ1 and on the 3 P_c (chamber pressure of each engine) meters on panel F7.

TOUCH SCREEN FOR ANIMATION AND CHECK THRUST LEVELS

Figure 9–3. One of the instructional screens of the Regency system used in training.

ENGINEERING SIMULATORS

While flight simulators are the glory of the simulation business, engineering simulations help make spaceflight possible. Many times highly innovative systems proved themselves in extremely accurate simulations. One example is the control moment gyro system used in attitude control of Skylab. A large simulator constructed at the Marshall Space Flight Center gave engineers valuable data about the behavior and feasibility of the system, which was understood by few aside from its inventor. Also at Marshall was a simulation of the Shuttle's main engines. These first computer-controlled rocket motors run much hotter and closer to destruction than any predecessors. Software for the engine controllers can be tested and certified in the simulator. At Johnson Space Center and the Rockwell plant in Downey, California are full-scale engineering simulators of the entire Shuttle orbiter. Early in the program, engineers led by Kenneth Mansfield at Johnson used these simulators to work out preliminary concepts, flight techniques, and procedures development using functional simulations (no flight hardware). After the installation of the actual hardware, changes to the individual hardware and software components could be checked for integration with the remainder of the spacecraft in those simulators. Thus, engineering simulators provide engineers with help in requirements analysis, prototyping, verification of concepts, and integration testing.

Simulation of components involved in rocket flight began in the late 1930s with the German development group at Peenemunde, where attitude control systems were simulated. In 1939, a one-axis mechanical simulator of the A-4 rocket's motion about its center of gravity provided valuable control data without the expenditure of test vehicles[39]. That device led conceptually to a more robust electronic analog simulation of the control system designed by Helmut Hoelzer and built under his direction by Otto Hirschler. Included in that simulator was an analog device to correct for the vehicle's lateral drift while in flight. Completed in 1941, the simulator was the most advanced analog computer built to that time[40].

Following World War II, the Peenemunde group brought the concepts of simulation to the United States. Hoelzer became head of the Computation Laboratory at the Army Ballistic Missile Agency research site in Huntsville, Alabama. When NASA absorbed the Agency's Huntsville facilities in 1959, Hoelzer continued his work and gathered a powerful set of digital and analog computation devices at the Marshall Space Flight Center. So much simulation work needed to be done that Hoelzer developed a simulation system consisting of a set of general-purpose digital, analog, and hybrid computers that several projects could use. Usually consisting of a large analog device and supporting digital minicomputers, the hybrids modeled booster flight characteristics and tasks such as payload loading, space

telescope pointing, attitude control problems, circuit design, and mission support[41]. One system, the SMK-23, modeled moving vehicles such as the lunar rover, providing television window views inside a closed control cockpit[42]. Besides this central facility, Marshall Space Flight Center also developed two large stand-alone simulators for special complex problems: the Skylab Attitude and Pointing Control Simulator and the Hardware Simulation Laboratory used to model the space Shuttle main engines.

Skylab Simulators

Prior to Skylab, the primary method of attitude control in a spacecraft was the use of reaction control system jets that burned liquid fuels. With a mission profile of up to a year's worth of occupancy and experimentation, Skylab could hardly carry enough fuel to maneuver its bulk for that length of time. An alternative solution was a control moment gyro (CMG) system that was very innovative and complex. A redundant digital computer system provided commands to the system in orbit. To study the operation of the complete system, including the computer and its attendant software, required the construction of a complete laboratory dedicated to the task.

Three rooms of the Astrionics Laboratory at Marshall were set aside for the simulator. In the Black Room sat the hardware that simulated the motion of the space station. The Green Room held the control devices and some of the computing equipment, with the remainder in the adjoining computer room. Primary computer for the simulator was a hybrid consisting of an XDS Sigma V digital computer and Comcor Ci5000 and Ci550 analog computers. These drove the simulation of the orbital workshop and interfaced with the ATMDC which flew on the actual spacecraft. A SEL 840 digital computer sent digital commands to the on-board computer[43].

Originally, the use of the simulator concentrated on mission planning and hardware and software verification tasks. Engineers expected to operate it less than half the working hours of a normal week. However, due to the severe hardware failures on the actual mission, the simulator reverted to 24 hours a day, 7 days a week operation. First the micrometeoroid shield and solar panels were damaged during ascent. This meant that the workshop had to be oriented in ways not set out in the requirements. For nearly 2 weeks, while Marshall prepared tools and techniques to effect repairs with the first crew aloft, the simulator tested attitude control maneuvers that would keep the workshop from excessive internal heating. Later failures, especially the loss of a CMG, were successfully modeled and solutions devised. As in the Apollo 13 flight, ground simulation of actual flight damage led to safe alternatives.

Space Shuttle Main Engine Simulator

The main engine of the space Shuttle is another complicated device that needs its own simulator. The Hardware Simulation Laboratory is the primary site for verifying the design of the main engines, testing the engine controller software, preparing for hardware changes such as new controllers, and modeling failures such as faulty valves and sensors that caused engine shutdowns on the pad and in flight during the Shuttle program. Begun in the early 1970s, by 1975 the engine simulator became operational. At the heart of the first version of the Laboratory were two Ci5000 analog computers and a SEL 840 MP digital computer. The engine, actuators, and sensors are simulated with the hybrid stem. Actual engine control computers are mounted in the simulator[44].

Figure 9–4. A collage depicting the Hardware Simulation Laboratory at the Marshall Space Center used for testing the Shuttle Main Engine Controllers. (NASA photo 331594)

Since Marshall was responsible for all the booster components of the Shuttle, it developed other devices that modeled those components in the largest of the active engineering simulators, the Shuttle Avionics Integration Laboratory.

SAIL: Fully Operational Shuttle Skeleton

The Shuttle Avionics Integration Laboratory, or SAIL, one of the largest engineering simulators ever built, sits in a big bay at the Johnson Space Center. A fully functioning skeleton of the Shuttle orbiter, it contains all avionics components used on the real orbiter, totaling nearly 1,750 black boxes weighing 6,000 pounds[45]. In fact, they are placed in exactly the same positions as in the actual spacecraft so that components can be certified and any changes made to the avionics can be tested. Also, software for a particular flight can be run to check for errors. Through the first six flights of the Shuttle program, the SAIL accounted for 241 errors found in the primary software and 196 errors in the backup software. For the first flight, SAIL operated for 644 shifts and since then has averaged 80 shifts in support of a mission. Just short of 350 contractors and NASA personnel manned the Lab in its operational phase.

Planning and construction of the SAIL began in 1968, when the Shuttle Engineering Simulator first began operations. This simulator, still functioning after many modifications 15 years later, replicates a cockpit. Scene generators for one forward window and both rear and overhead windows, as well as four SEL minicomputers and a Control Data Corporation Cyber 74 mainframe, drive the simulator. Preliminary work on this simulator gave experience that contributed to the SAIL, which started in 1972[46].

Until January of 1983, the SAIL itself consisted of a guidance and navigation test station; the Shuttle Test Station, which is the skeletal orbiter; the Marshall Mated Element System, which simulates the propulsion system; a ground standard interface unit, which sends commands and acquires data from the SAIL for display; and a subset of the Launch Processing System. Since the avionics system is the only real hardware in the orbiter mock-up, the orbital maneuvering engines, reaction control system, main propulsion, and other non-avionics boxes must be simulated by computer software or analog devices. To preserve the exact signal timing, these simulators must, in some cases, be located farther from the spacecraft skeleton than the real equipment. The forward reaction control jets simulation boxes, for example, are over 10 meters from the spacecraft nose. Since Marshall contributed 55 racks of electronics, and Kennedy Space Center sent the Launch Processing System subset, each center can use the SAIL to verify software written for equipment under their development control,

Figure 9–5. Astronaut John O. Creighton in the cockpit of the Shuttle Avionics Integration Laboratory. (NASA photo S-79-39162)

such as the engine controllers from Marshall and the interfaces and selected test software from Kennedy[47]. SAIL operators can monitor tests from display control modules connected to the interface unit. The consoles have color monitors and individual processors used for fault detection. Aside from validating engineering changes and software, the SAIL is used for validating tests to be carried out later on the spacecraft while it is being prepared for flight[48].

An extensive hybrid computing center drove the SAIL and its attendant simulators during its first decade. A pair of EAI 8800 analog computers simulated the landing gear, runway, and braking. A pair of 7800s represented the aerosurfaces and rate gyros. These analog computers were replaced with a pair of Gould SEL 32/8780 digital minicomputers in 1983. Other SELs provide a digital autopilot simulation, equations of motion, radar altimeter, and other nonavionics functions[49]. Separate computers generate the window scenes. These are so much better than those done in the Shuttle Mission Simulator, especially in regard to the Remote Manipulator System, that crews prefer to use the Shuttle Engineering simulator in the SAIL for training when a mission requiring use of the arm is coming up[50].

Since the Shuttle was the first manned spacecraft to fly without unmanned development flights, the SAIL's importance cannot be minimized. By essentially replicating the entire spacecraft and its operations exactly as the spacecraft currently exists, the SAIL provides NASA and the astronaut crews confidence in the hardware and software for each mission. In its role, SAIL is the ultimate engineering simulator.

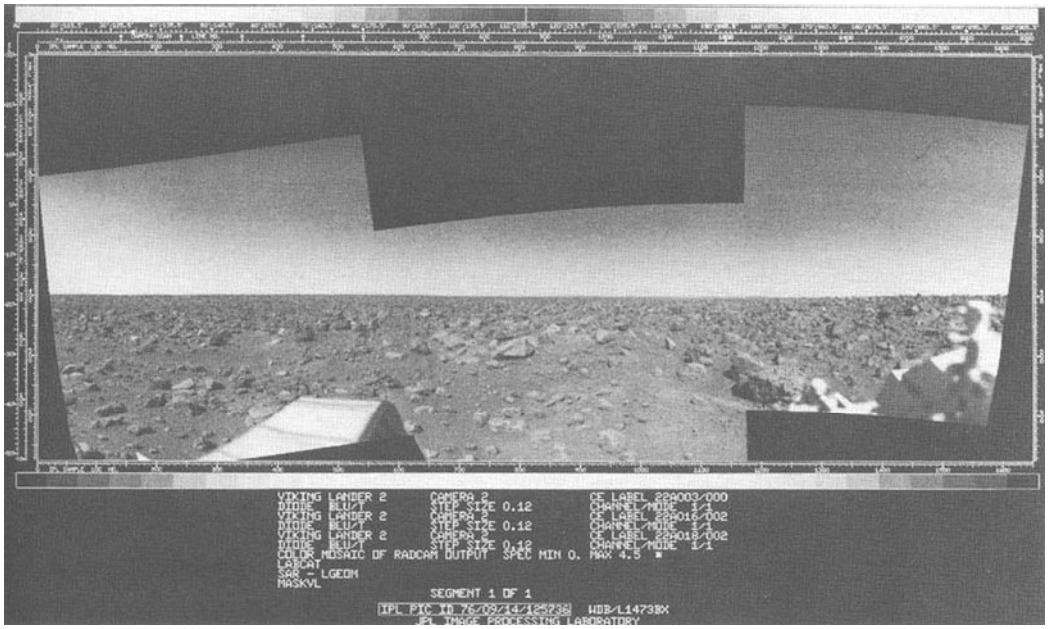


Figure 9–6. Image processing makes possible scenes from alien worlds such as this panorama of the Martian surface. (JPL P-17982)

IMAGE PROCESSING

Image processing is one area in which NASA, primarily through work done at JPL, clearly leads the field. Ironically, even though the production of high-quality images from space probes and Landsat earth orbiters has great scientific and public relations value, the concept of digital image processing was not incorporated in the original planning of a number of early missions. Instead, it had to gain acceptance as a "tack-on" to the Ranger and Surveyor programs[51]. Robert Nathan led the development of digital image processing in its early stages, and with the technical help of other JPL scientists, won for it a featured place on the planetary missions of the late 1960s and beyond. Of the early resistance, he later said that he "had to prove to [project management] each time what they needed" to get the most out of the first American pictures coming from space.

Nathan came to the California Institute of Technology as a graduate student in 1952. He earned a Ph.D. in crystallography in 1955 and soon found himself running CalTech's fledgling computer center, where he received a good grounding in the potential of digital computers. In 1959, he went to JPL to help develop imaging equipment to map the moon. When he saw the Russian pictures of the far side of the moon, he thought he could do better and began developing digital techniques for image enhancement using a small NCR 102D computer. Nathan reasoned that analog equipment, such as television cameras, could only be controlled by hardware changes, just like an analog computer can only have its internal program changed by rewiring or switching components. However, digital processing allows changes to be made with software, allowing a wider variety of enhancements[52].

Before an image can be processed, it must be put into digital form. Frederick Billingsley and Roger Brandt of JPL devised a Video Film Converter (VFC) that could transform analog video signals, such as those sent back by Ranger spacecraft, into digital data. While they supervised the construction of the device, John Morecroft of JPL used the NCR computer to begin programming processing algorithms. These events took place in 1963, and by the next year Howard Frieden had programmed the Laboratory's institutional IBM 7094 computer to process Ranger data. Success with Ranger images led the Surveyor project to use Nathan's techniques, as well as Mariner Mars 1964. By the Mariner Mars 1969 missions, the concept of digital image processing was fully accepted.

Why is image processing needed? Due to the resolution and design of the video cameras used to make the images, they must be processed in order to return the most information possible. The surface of Mars is such a low-contrast object that without enhancements, features would be lost in the wash of monocolor[53]. Also, because the human eye cannot adjust to differences in illumination across a field of view, illumination must be normalized[54]. The cameras operate by taking an instantaneous view of the scene; the values of the light impressed on the vidicon tube are then made into digital data. Since images are taken one after the other, very close together in time, residual images from prior "snapshots" affect the current view[55]. These residual images must be removed, a technique that took several missions to perfect. Finally, noise from transmitting a signal over planetary distances must be accounted for.

To see how such processing is done, the real-time display system used for the Mariner Mars 1971 orbital mapping mission provides a useful example. A UNIVAC MTC 1230 computer extracted 9-bit pixel data from the telemetry stream. A pixel is a single picture element, or dot. The spacecraft had a camera capable of recording frames of 700 lines by 832 pixels, or 580,000 individual dots. Such large numbers of

pixels were only practical as interplanetary communication advanced. Mariner Mars 1964's 200 by 200 pixel imaging equipment transmitted at the rate of 8 1/3rd bits per second. Thus, it took nearly one entire shift at a Deep Space Network station to record a single frame. At that data rate it would take over 1 week for a Mariner Mars 1971 frame! But by 1971, the data rate increased to 16,200 bits per second, giving a complete picture in 5 minutes and 40 seconds. Even these rates increased by over seven times in the next few years.

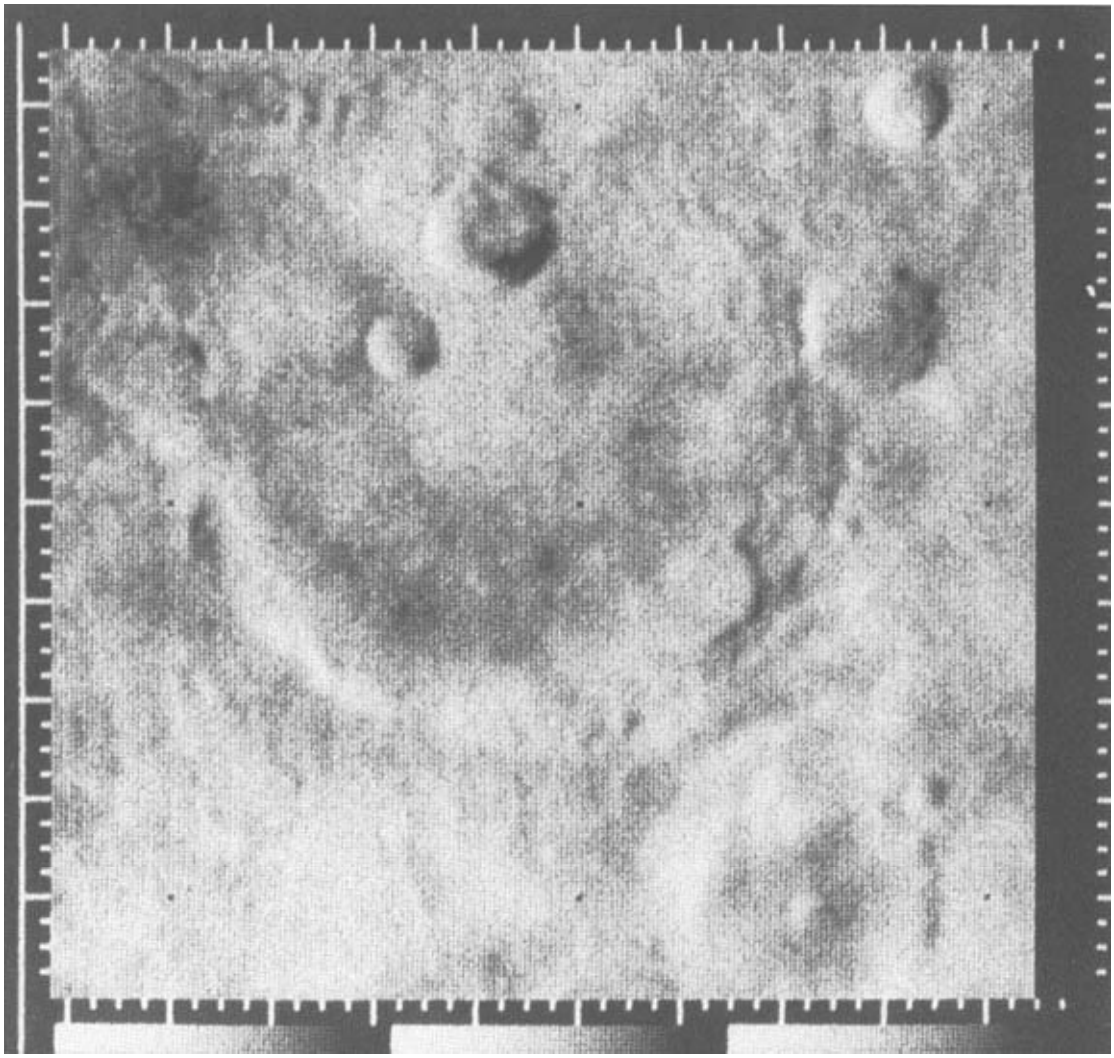


Figure 9–7A. Image processing's decade of progress: Mariner Mars 1964 returns the first closeups of Mars. . . .

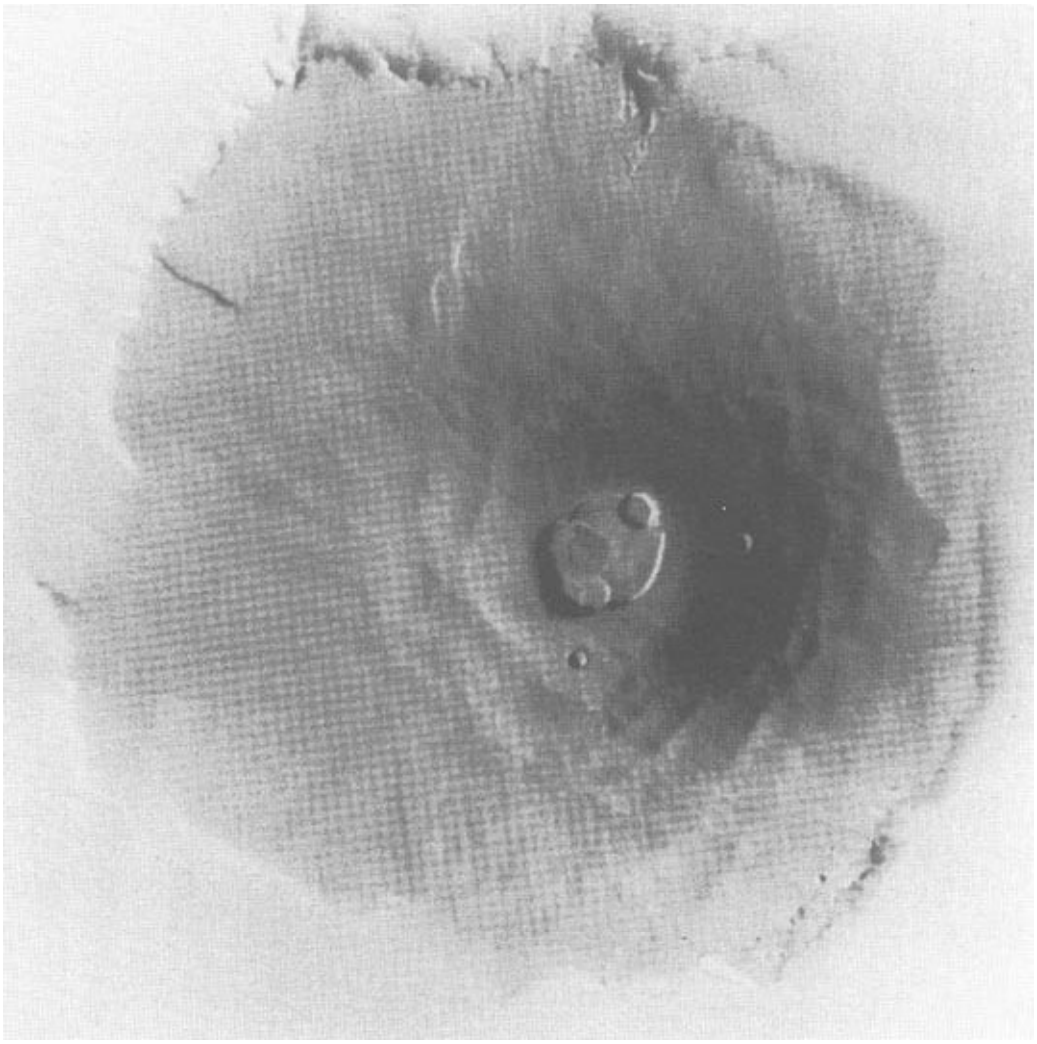


Figure 9–7B. . . . .As the planetwide dust storm clears, Mariner Mars 1971 scans Nix Olympica in January, 1972. . . .

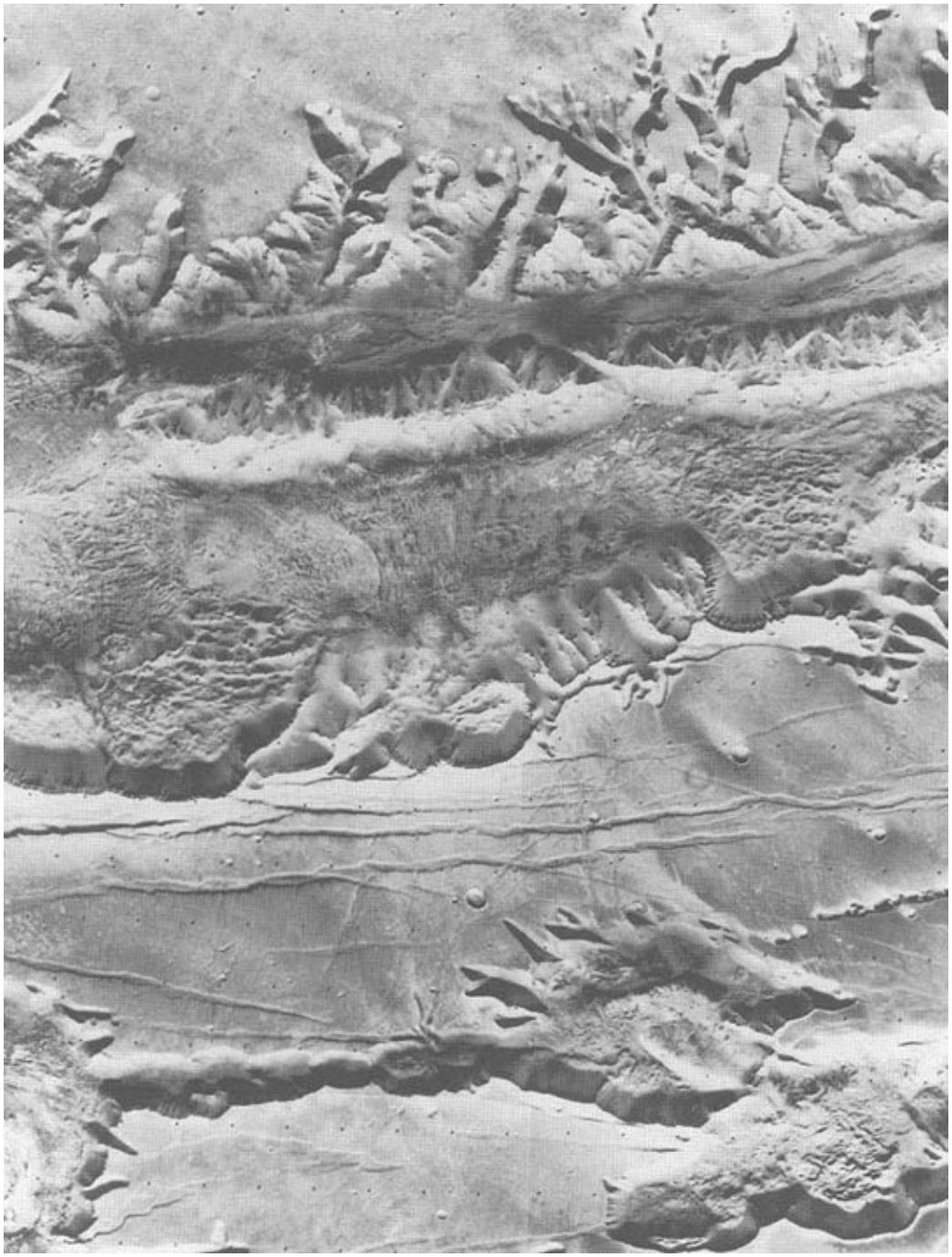


Figure 9–7C. . . . .Details from the Valles Marineris canyon taken by the Viking Orbiter in 1976. (JPL photos P-7875A; P-13074; P-17872)

Several techniques could be applied to the data by the computer. Contrast stretching helped increase the contrast of the single color Martian surface. Original values of the pixels ranged from 0 (black) to 255 (white). The computer truncated these to 6 bits, which yielded 64 levels. Since humans can only discern about 25 levels of grayness, this was more than enough. By increasing the brighter grays toward the white end of the scale and decreasing the darker grays toward the black end, the contrast was increased[56]. Illumination could also be normalized using the computers. A "high pass filter" corrected the value of the pixels by averaging the immediately surrounding 125 pixels and then subtracting the running average from the value of the pixel[57]. Another process compensated for geometric distortion. Simply because of the way the cameras were made, there was distortion in the image frames. Reference points marked on the image served to help distortion elimination algorithms properly square off the image. These techniques were also applicable to developing mosaic maps by taking images shot at oblique angles and flattening them out in any one of several projections[58]. Noise elimination could be done by assuming that any pixel exceeding a difference of 32 levels of brightness from its neighbors was a spike and then changing the value of the spike to the average of its two immediate neighbors. From 20 to 10,000 spikes could be found on a single raw image, so without removal the image would be noticeably damaged[59].

Aside from the near-real-time imaging provided by the UNIVAC and other computers on later missions, long-term processing with a number of techniques is done in the Image Processing Laboratory at JPL. First established in 1965 with a new IBM 360/44 computer that lasted 10 years, the Processing Lab pioneered new imaging techniques and developed support software to implement them. Central to the success of image processing was the Video Information Communication and Retrieval language, or VICAR. Written in 1966 after a design by Stan Bressler and Howard Frieden, VICAR enabled users to define a pipeline of processes without having to use cumbersome job control language. For instance, VICAR could define an image file to be processed and then specify the type of processing to be performed on it in a sequential manner. Output from the stretching program could thus be directed to the input to the geometric transformation program. The existence of this language significantly increased the value of the imaging[60].

By 1975, when a 360/65 replaced the older computer, the Image Lab did roughly half of its work on planetary imaging and half on earth resources work using Landsat images[61]. Also, by that time numerous spinoffs from the program began to turn up in other fields, chief among them astronomy and medicine. Astronomers now use digital techniques to enhance their photographs of celestial objects in the same way spacecraft images are processed. Nathan left the planetary imaging

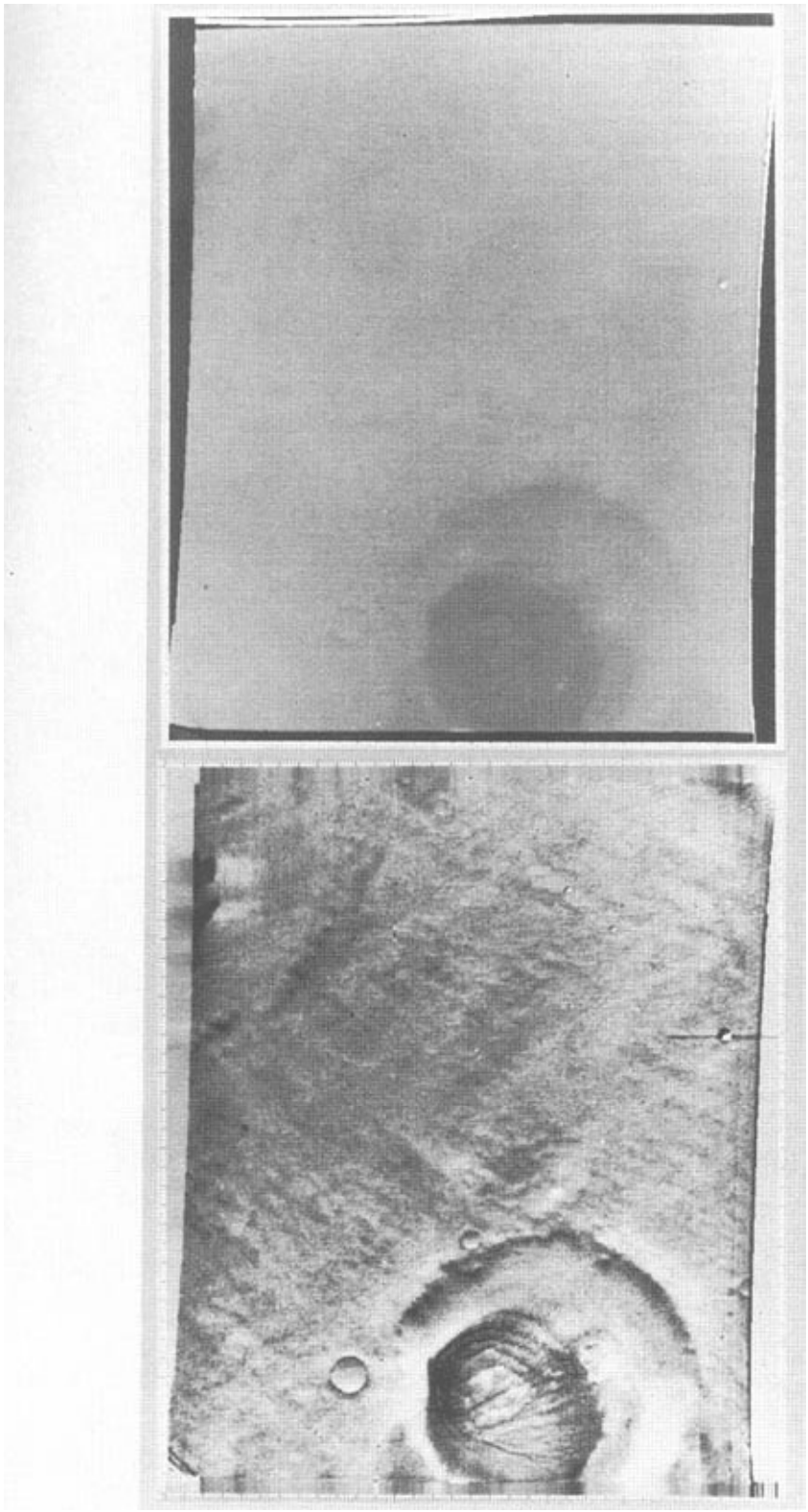


Figure 9–8. Increasing contrast enhances a Mars image. (JPL 511-4353)

to his colleagues in 1968, when he turned his attention to a series of grants from the National Institutes of Health to study applications of digital image processing to microscopy and medical diagnosis. Robert Selzer of JPL had applied the techniques to x-ray enhancement. For Nathan, with a background in x-ray crystallography, this was a natural step. Unfortunately, by 1973 the government canceled all fundamental research grants in the field and Nathan found himself without support and nearly without a JPL position[62].

Nathan managed to hold on for a few more years at JPL on other projects until, in the late 1970s, he thought of a way to increase the speed of the then computer-time-hungry image-processing programs. With Mariner Mars 1971 it became possible to send images faster than they could be processed. Since then, the ratio between transmission time and processing time has gone way up in favor of transmit time. In general, it does not really matter, since instant images are not now a requirement, but for users of image processing other than planetary scientists, additional speed is attractive. Also, as the number of images has skyrocketed from Mariner Mars 1964's 22 to literally tens of thousands in the Voyager and Galileo projects, time to process the images is of interest even to the most patient. The problem is that as the number of pixels has increased, the number of individual computations also increases. A 1,000 by 1,000 pixel image weighted 35 by 35 times requires 1.225 billion multiplications[63]! If these are done in sequence, the amount of processing time would be formidable.

To solve this problem, Nathan suggested putting 35 sets of 35 multipliers in parallel on very large-scale integration (VLSI) chips. By doing that, the amount of calculations is reduced by 1,225 to 1. Recently, he has begun design of a set of VLSI chips that will speed up the geometry or reprojection operations[64]. Basically, the weighting algorithm is encapsulated in a single chip as a unit of hardware, rather than as software. Logic in hardware executes faster than logic in software because all 1,225 multipliers are operating simultaneously in parallel rather than one at a time serially as in a central processor. Nathan's chips have been plugged into Digital Equipment Corporation VAX 11/780 computers. When the computer is executing an image-processing program and reaches the point where it wants to do the algorithm on the chip, the computer "calls" the chip just as though it were calling a software subroutine.

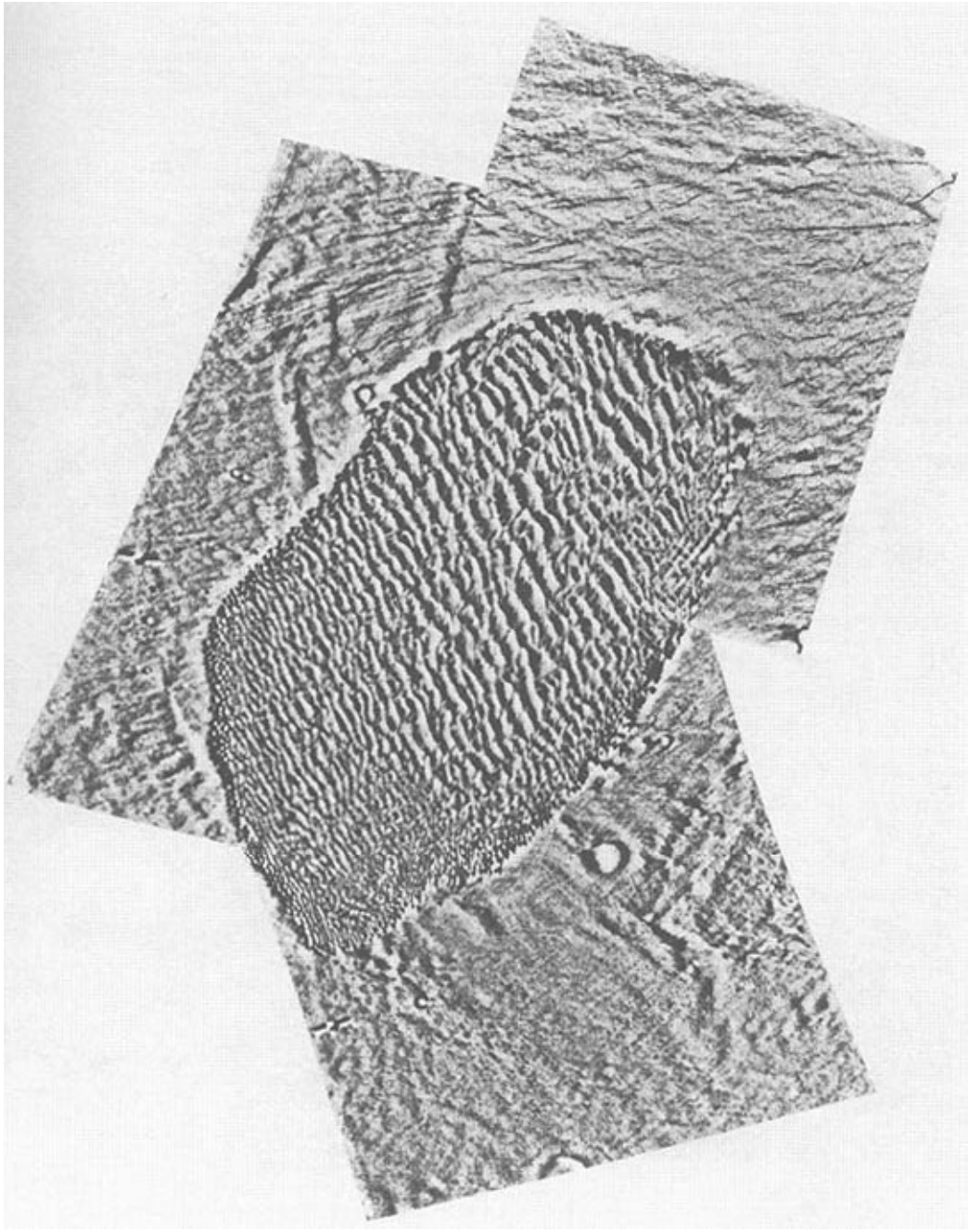


Figure 9–9A. Mosaics combine detailed images into detailed maps: a Martian desert. . . .

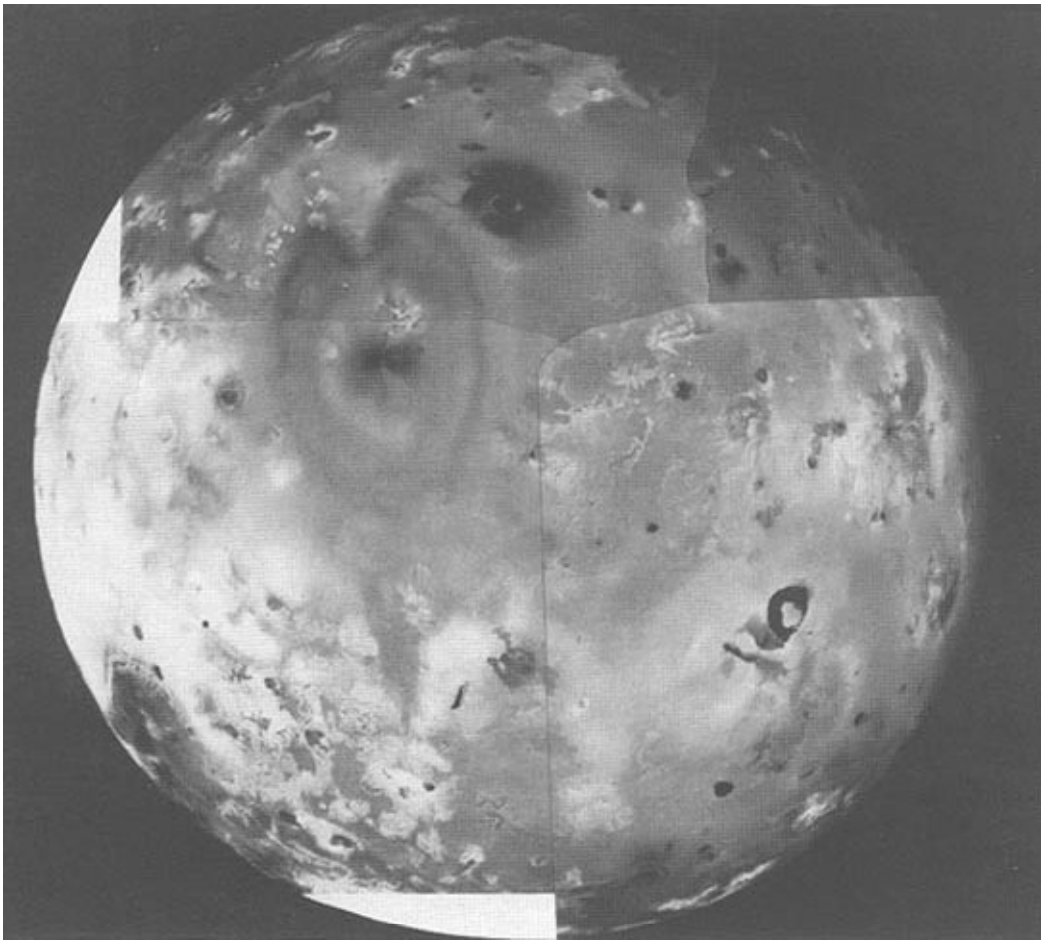


Figure 9–9B. . . . .Volcanic Io. . . .

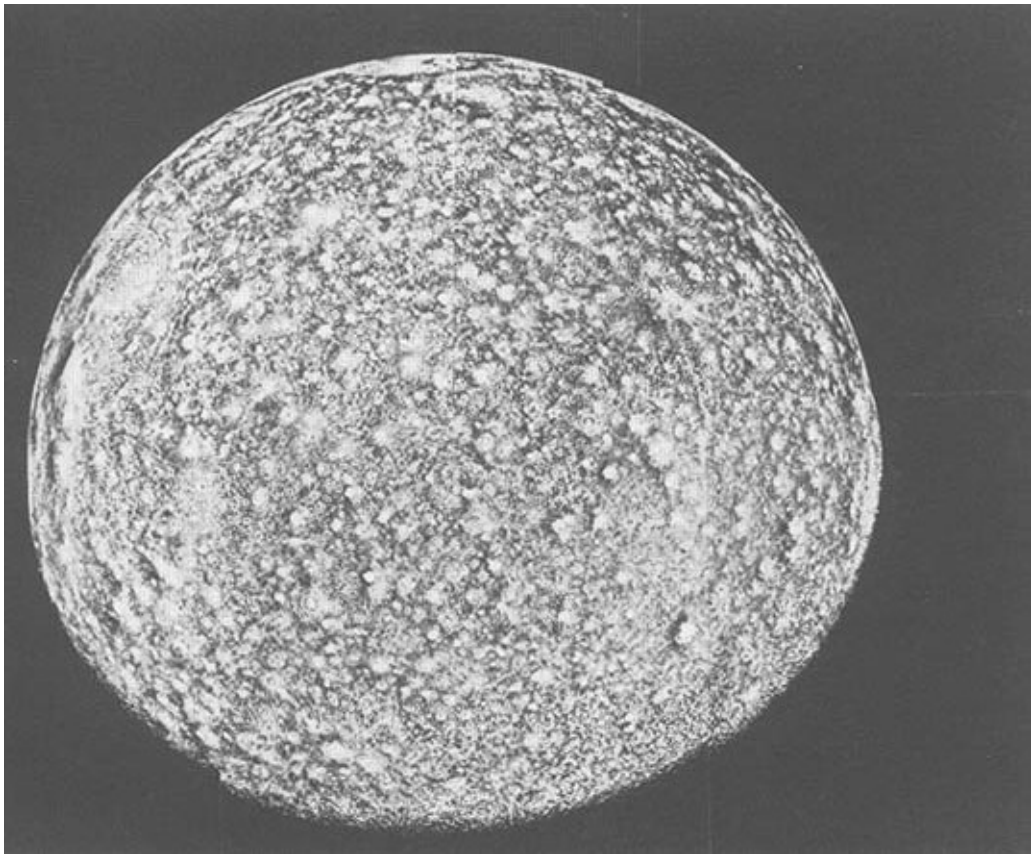


Figure 9–9C. . . . .Heavily cratered Callisto. (JPL photos 211-4704; P-21278; P-21746)

Nathan sees his invention not only as the solution to a problem in image processing but also as the beginning of a new future in computing. Using this technique, special-purpose computers with a lot of logic embodied in hardware could easily outstrip the existing systems in speed and accuracy. In some ways, it would be like electronic analog computers, but better in that the rearrangement of components would be simpler.

It is fitting to end on this note, as Nathan's application of computers to fulfill a need in space exploration mirrors the entire story of NASA's use of computers. He approached his tasks in the late 1950s and early 1960s as a pragmatist. He had some computing background, as well as grounding in other fields, so he could see the possibilities of applications. He used equipment usually behind the state of the art but got beyond the state-of-the-art results with it. And, finally, he repays computing by finding one way to improve it on the path to solving yet another problem. Nathan himself said that "NASA is not to be given credit for initiating advances in image-processing technology, but NASA has supported the grass roots initiatives." In general, that is true. NASA never asked for anything that could not be done with the current technology. But in response, the computer industry sometimes pushed

itself just a little in a number of areas. Just a little better software development practices made on-board software safe, just a little better networking made the Launch Processing System more efficient, just a little better operating system made mission control easier, and just a little better chip makes image processing faster. NASA did not push the state of the art, but nudged it enough times to make a difference.