Teaching Editorial

Nuclear Medicine Computers—Software

Although the aphorism that "any technology, sufficiently advanced, is indistinguishable from magic" is clearly true, the computer, no matter how complex it may seem in any particular configuration, is not in this category of technology. Any technologist or physician who pretends to be an intelligent computer user must have at least a rudimentary understanding of the basic concepts of nuclear medicine software.

These concepts are illustrated in this, the second of two articles presenting a review and overview of computer systems used in nuclear medicine. A previous article focused upon hardware (*1*). In keeping with the format of the first article, this discussion will: 1) give an overview of the general concepts of software; 2) define the steps involved in software production; 3) examine how the various pieces are tied together in the system; and 4) consider a number of specific clinical applications in image processing and data analysis.

Software is simply the set of instructions that provides control over the calculations and operations of the computer. Table 1 depicts different types of software used in any given computer system (*2,3*). Clinical users should be familiar with clinical analysis software, but it is necessary for any competent computer user to have at least a nodding acquaintance with the rest of the system. There may appear to be a fundamental difference between the program used by the technologist to collect data from a rotating SPECT camera and that which is used by the computer to control the data transfer between memory and disk. The difference, however, is one of form of implementation rather than anything else. Both programs are based on a set of rules laid down by the designer before the program has been written; they respond to stimuli from the outside world and transmit control commands to outside devices.

PROGRAM PRODUCTION

Prior to a detailed discussion of various components of the software system, it is helpful to consider a general overview of program production—from conception through operation.

The first step in the production process is the conversion of a concept into a detailed plan for the program (Fig. 1). This plan, known as an algorithm, contains specifications for each step to be taken in calculating, and the various options allowed at logical decision points in the process. The design of the algorithm is often accomplished by generation of a flow chart. The algorithm is converted to a program through the use of a programming language that allows the programmer to instruct the computer in "human language." The code generated in this step is entered into the computer through a text editor, very similar to an office word processor. Once the code is free of typing errors, it is processed by a language translator, known as a compiler, which converts the codes to a form more acceptable to the computer hardware. At this stage, the program is examined for executional inconsistencies and errors in coding which the compiler may be able to recognize. The last step in program production is combining the user program with supporting software supplied by the system. The programmer may tell the computer to add two numbers, but it is ultimately necessary to provide the commands that tell the computer exactly how to do it. These detailed instruction sets, and other software, which control the various physical devices, are stored in libraries from which they can be called. The linking program examines the user software, decides which supporting routines are required, and ties all of the software into a package called the load module. It is this load module that is finally executed by the clinical user.

OPERATING SYSTEM

The core of the software system for any computer is the operating system (*2,3*) which provides the basic intelligence for the computer (Fig. 2). It forms the link between the user and the program, the program and the data stored on disk, and the disk hardware or other peripherals and the memory.

Every device connected to the computer must operate according to a set of rules controlled by the computer. Although the actual passage of data between the main computer and any peripheral usually takes place without software intervention, the starting and appropriate closing down of data transfer requires a class of software known as device controllers or device handlers. Device handlers are written according to very strict protocols that specify all aspects of the interactions between the handler and the operating system.

On the other side of the operating system are the application programs which calculate ejection fraction, reconstruct

For reprints contact: Jon J. Erickson, Dept. of Radiology, Veterans Administration Hospital, Nashville, TN.

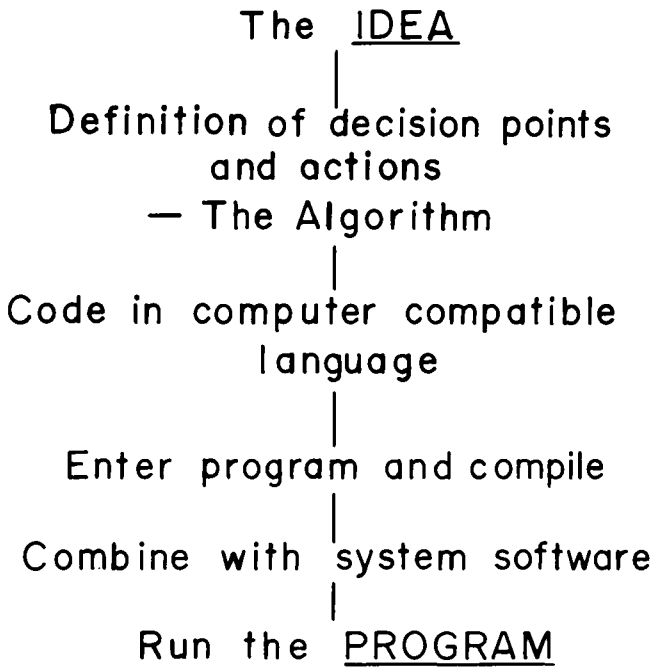

FIG. 1. Steps taken for conversion of an idea for a processing procedure into a working clinical computer program.

SPECT images, and type patient reports. These programs all obtain and return data through the operating system. Any time a program requires data for a calculation it requests the operating system to provide it by reading it from a disk, collecting it from a scintillation camera, or asking the user for data by keyboard entry. The operating system—whether it be Digital Equipment Corporation's RT-11, RSX or Data General's RDOS—determines the computer's personality.

There are a number of different operating systems, ranging in complexity from the very simple single user to the very complex multi-user/multi-processor systems. The early systems were primarily of the single-user type. As its name suggests, this system allows only one user to function at any given time. This one user may be a person collecting data or a person analyzing data. When computer analysis was more of a novelty than a necessity, this system was a perfectly satisfactory mode of operation. However, as soon as a clear clinical need for computer support was demonstrated, the limitations of the single-user system were apparent.

TABLE 1. Type and Function of Software Found in Computers

Type	Function
Applications	Clinical analysis, data collection
Device drivers	Control of disks, displays, data acquisition hardware
System utilities	File transfers, text editors, system maintenance, program linkage
Record keeping	Maintenance of file directories, separation of user areas
Language translators	Conversion of user programs to machine language

In an attempt to answer the need for more efficient and responsive computer support, two other operating systems were developed. The foreground/background system and the multi-tasking/multi-user operating system are found in modern computers. Although there is a clear distinction in theory between these two systems, actual clinical implementations often make the distinction difficult to perceive. The foreground/background system is designed primarily to handle situations in which the computer is controlling some external device or collecting data in the foreground and, consequently, spends a great deal of time simply standing around waiting. During nonactivity on the data collection job, the computer can execute another program in the background in a different memory section. This philosophy is the basis on which nuclear medicine systems are designed. The computer can be effectively used for analysis of previous studies while it is collecting data from a scintillation camera and awaiting completion

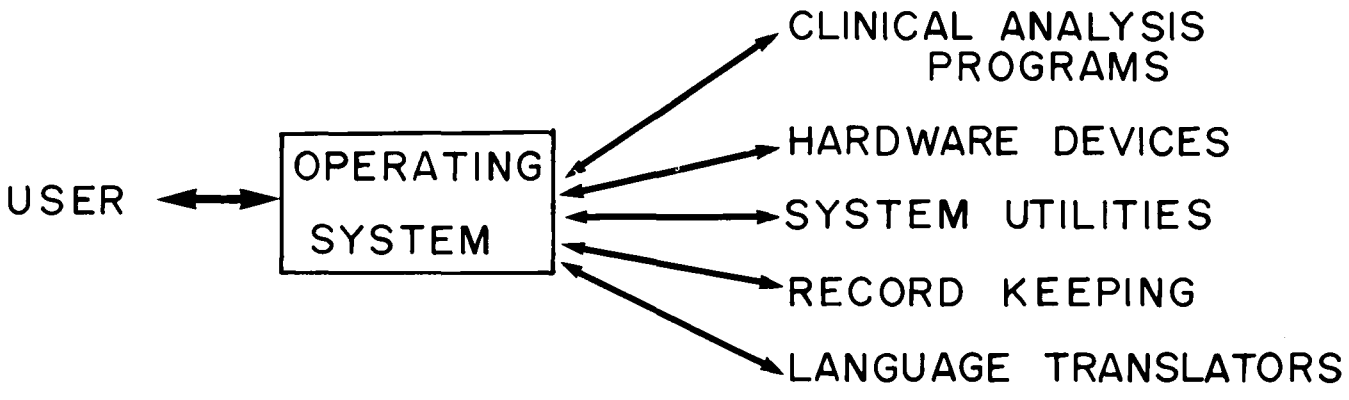

FIG. 2. The operating system runs the computer and acts as intermediary between the applications program, user, and other system software.

of the study. The limitation of foreground/background systems is that it cannot support two programs that spend a majority of the time calculating; hence, simultaneous collection and analysis are allowed but not two simultaneous analyses.

The multi-tasking/multi-user system is capable of handling more than one general user (i.e., collection or analysis) because it divides its time between the users not only on the basis of need but on the basis of a time-sharing operation. When any external piece of hardware needs to be serviced, the computer transfers control to the program using that device. On the other hand, if there are no hardware tasks requiring attention, the operating system moves from user to user on the basis of time. Each user of the computer receives attention from the operating system for some part of each second of time. In this way, each user has the impression that they are working alone. Usually, the only evidence of other users is a slightly slower response as the number of users increases.

PROGRAMMING LANGUAGES

The computer is ultimately controlled by binary instruction words loaded into a command register from which the various sequences of ones and zeros are used to control the execution of a relatively small number of logical operations. Because it is extremely difficult and time consuming for any normal human being to write programs in the one/zero form required by computer hardware, several human language processors have been implemented to aid in writing programs (*2,3*).

Assembly Language

One step above the one/zero form of machine language is the assembly language. Assembly language is relatively primitive so that desired logical operations have to be set down step by step for the computer. The operations can be specified by name through the use of mnemonics. The following outlines give an example of an assembly language program for the addition of two numbers and storage of the results; the desired calculation is C = A + B.

Step 1.	LAC A	(load the value of A into a binary register that can do arithmetic)
Step 2.	ADD B	(add the value of B and leave the result in the register)
Step 3.	DAC C	(deposit the result in a memory location designated for the sum)

Assembly languages are system specific. A program written for a DEC PDP-11 will not work on a DEC VAX or a Data General NOVA. Assembly language programming, usually limited to operating systems and device controllers is advantageous in that the computer will execute the instructions exactly as written without attempting to optimize the execution or otherwise alter the intent of the programmer.

FORTRAN

FORTRAN (FORmula TRANslator) is one of the oldest languages intended for use by ordinary mortals. Primarily designed for arithmetic calculations, it is a very powerful language that is much closer to the human language than any assembly language system. FORTRAN provides the user with the capability of handling data and information in a logical, although sometimes restricted, manner. It provides a structured method for accepting and sending data back to the operator or to outside storage (e.g., disk). It does not usually provide the programmer with direct access to control registers and data channels within the computer, but this is rarely a restriction in application programming.

The addition of two numbers in FORTRAN, as illustrated in the assembly language example, is accomplished by simply writing the expression, C = A + B. The advantages of FORTRAN, aside from its powerful arithmetic capabilities, are its availability on most computers (although it may have to be purchased separately on a clinical system) and standarization by international scientific bodies. A program written for one system will run (with some restrictions) on other systems. The degree of compatability of programs is determined by the extent to which the author of the program has avoided the peculiarities of the original system and the gray areas of the language standard. Unfortunately, these peculiarities and gray areas often provide the advantages of one program over another.

BASIC

BASIC (Beginner's All-purpose Symbolic Instruction Code) is another common programming language. Developed at Darmouth in the 1960s for faster assimilation by nontechnical persons, the early implementations of BASIC used commands that closely simulated English. For example, the command to add two numbers is: LET C = A + B. Modern implementations on the language have retained the ability to handle such niceties as the "LET" statement, but most versions will allow programmers to use a more truncated form (C = A + B) closer to that used in FORTRAN. In general, BASIC provides all of the computational capabilities of FORTRAN.

BASIC belongs to a class of languages known as interpreters (see below). An interpretative language does not require the intermediate step of conversion to computer machine language before it can be used because it converts as it executes, thereby, providing a more interactive environment for program development than FORTRAN. BASIC also provides a very powerful text-handling capability. Words and phrases can be combined into sentences with arithmetic-like commands. For example:

```
If:    A$   =   "This is an"
And:   B$   =   "example."
Then:  C$   =   A$ +B$
Produces the result: C$ = "This is an example."
```

The one disadvantage of BASIC is its execution speed. For programs that perform a large number of calculations, the conversion of the BASIC program to machine language on a line-by-line basis every time it is executed represents a time-consuming operation. BASIC is an excellent language for

report generation and other applications requiring a significant amount of data input by the user, but it is unsatisfactory for large scale image manipulations or data reduction.

Other Languages

A number of other programming languages have been designed either for specific applications or to resolve what were, in the eyes of the designer, flaws in existing languages. Two of the more popular languages are C and PASCAL. Although C is less of a human language than either FORTRAN or BASIC, the user applies the basic structures and operations of the language to build up powerful complex commands designed for specific applications. C can be used for a wide range of functions ranging from complete operating systems and device handlers to very versatile graphics and artificial intelligence software. Standardized through the creation of an industry standard, C can be used for codes that are transportable from one system to another.

PASCAL, developed in the late 1960s by Professor Niklaus Wirth at the Eidgenossische Techniche Hochshule in Zurich, Switzerland (*4*), was designed to produce a language containing a small number of fundamental concepts that would enable it to be used in the teaching of basic programming as a systematic discipline. The rapid spread of its use is evidence of its success. The language is designed so that programs written according to the rules are essentially self-documenting. With only a minimum understanding of the language syntax, it is possible for moderately experienced computer users to read a program written in PASCAL and understand both the intent of the algorithm and the details of the implementation. PASCAL, widely available on smaller clinical computers, may, as with other high-level languages, have to be purchased separately.

LANGUAGE PROCESSORS

Because human-engineered programming languages, such as FORTRAN, BASIC, C, and PASCAL cannot be used directly by the computer, it is necessary to translate them into machine language (*2*). The three types of translators are: assemblers, compilers, and interpreters. Assemblers translate the assembly language programs previously discussed and convert the mnemonic codes into binary machine codes. Compilers translate higher-level languages like FORTRAN into assembly language from which the assembler constructs the machine code. In current systems, the distinction between assemblers and compilers is not as clear as these two definitions imply. In early systems, a programmer would use the compiler to produce the assembly code and then envoke the assembler for the second translation step. Today, a FORTRAN compiler will automatically perform all of the translation steps.

An interpreter differs from a compiler or an assembler in that it translates the program line by line as it is run. BASIC is perhaps the best known interpreted language. Interpreted languages are generally easier for new computer users because they require less understanding of the computer operation. However, because interpreters do not produce a permanent version of the executable code, every line must be translated

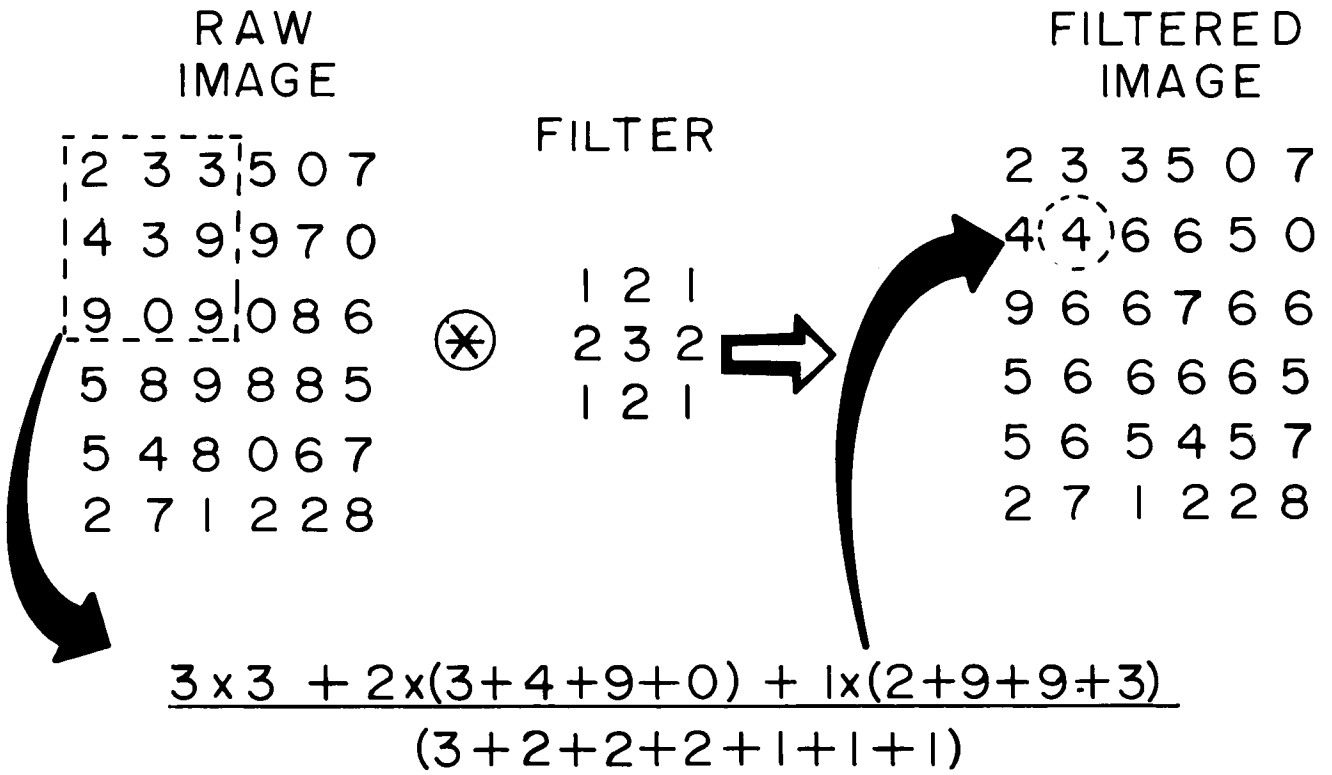

FIG. 3. Image filtering is accomplished by convolving the raw image with a symmetric array of filter weights to produce the smoothed (shown here) or edge-sharpened image.

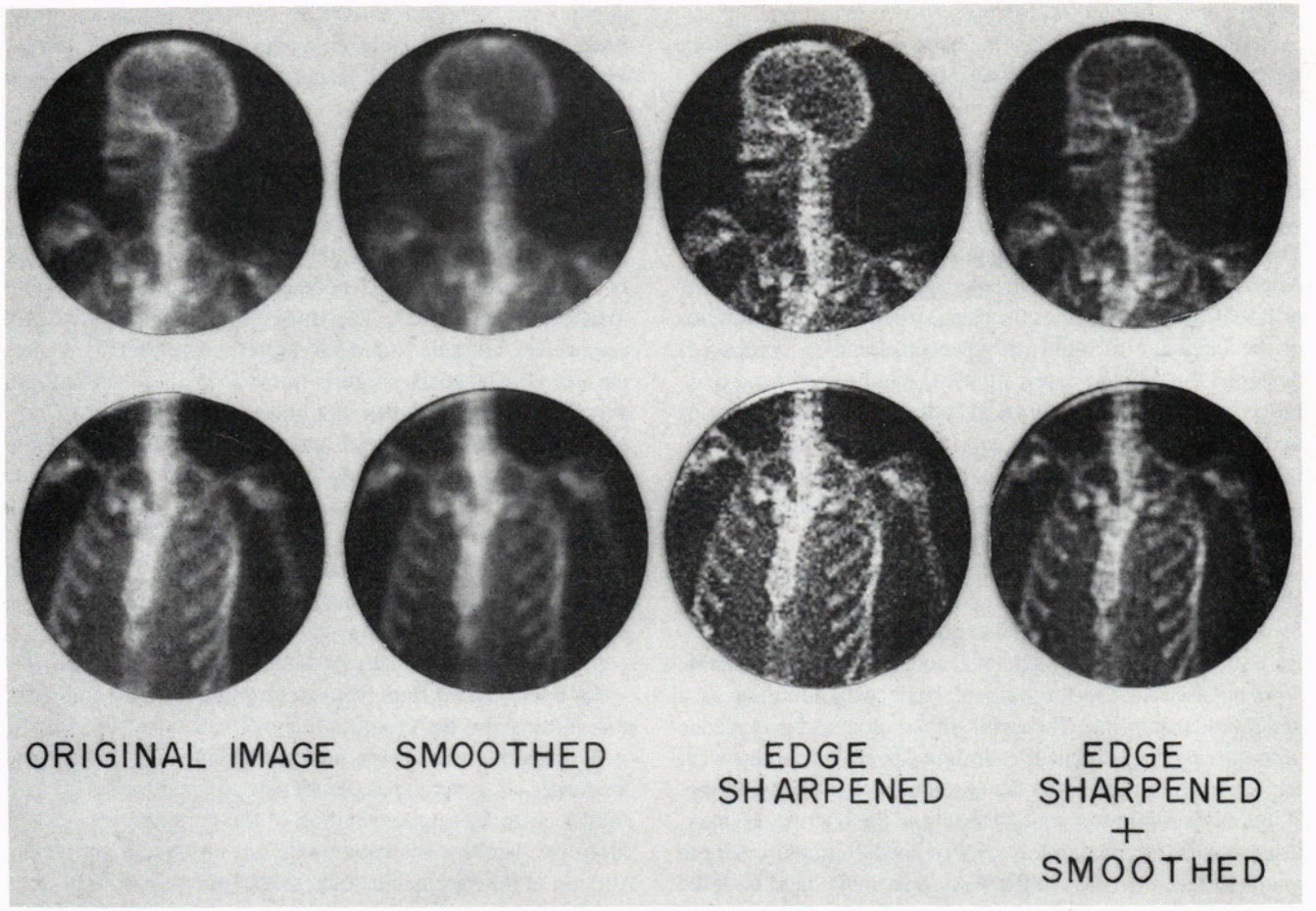

FIG. 4. Images may be smoothed or edge sharpened by the appropriate choice of filter weights.

every time it is used, resulting in time loss for complex, repetitive arithmetic (e.g., image smoothing or SPECT image reconstruction). Some systems provide a hybrid interpreter, a combination of interpreter and compiler, in which the interpreter is used until the program is complete and error-free, after which the compiler produces a permanent machine language version not requiring translation for execution.

EDITORS

Programs are entered into the computer using an editor program, similar to those used on personal computers for word processing. Many different editor programs are available, and it is not unusual to find two or three different editors on a single computer system. Editor programs have individual advantages and disadvantages, but the widespread availability of word processing currently makes a detailed discussion of them rather needless.

CLINICAL APPLICATIONS

Image Filtering

Image processing, with the intent of improving display information, was one of the first applications of the computer in nuclear medicine (5). Early investigators hoped that application of digital computer analysis to images obtained from rectilinear scanners and scintillation cameras would improve diagnostic quality. The primary difficulty was that the limited data acquisition matrices so degraded the spatial resolution of the imaging device that no amount of image processing could restore it to anything near the original analog image.

Image filtering consists of modifying the original image by logically "reimaging" it with a mathematical imaging device in which spatial response can be controlled by the user. The difference between a logical and a real imaging device (e.g., scintillation camera) is that the response function of the logical device can have negative values, whereas that of the real device can only be positive. The practical result of this difference is that the logical imaging device used for image filtering can be specified to improve spatial resolution and make edge and count density transitions more obvious. Alternatively, the response can be specified so that filtering smooths the image and reduces the noise so that that small differences in count densities can be more readily perceived by the viewer.

Image filtering is performed by a mathematical operation called convolution, which is simply the successive replacement of each point in the original image by a new value produced by a weighted combination of the original point and its surrounding neighbor points. Figure 3 illustrates this pro-

cess for a 3 × 3 filter function. Using larger filter functions, such as 5 × 5 or 9 × 9, will cause larger portions of the original image to have an effect on the value of the new image points.

Figure 4 shows the results of applying a number of different filter functions to the same clinical image. By changing the weighting of the filter elements, it is possible to produce either edge sharpening or image smoothing (*6,7*).

Cardiology

Cardiac studies represent another principal application of computers in modern nuclear medicine. A number of factors, evaluated through the use of radionuclide imaging procedures, include left and right ventricular ejection fraction, cardiac output, wall motion, cardiac size, intraventricular shunt quantitation, and myocardial ischemia (*8*).

The gated cardiac study, perhaps the most visually dramatic, is used for ejection fraction calculation, cardiac output, and evaluation of wall motion abnormalities. In the gated study, data collection is synchronized with the cardiac cycle through the ECG signal (*9,10*). The data collection is setup in such a way that a series of images representing short segments of the cardiac cycle are collected in the memory of the computer. At the occurance of the QRS complex (end-diastole), the computer begins data collection in image one. At relatively short intervals (20–40 msecs), the data collection is moved to the next image in the series until the number of desired segments are collected or the cardiac cycle restarts with another QRS complex. The data are collected over a large number of cardiac cycles so that the images represent an average cycle rather than any one single cardiac cycle.

The first analysis step for this study is to replay the sequence of images in a movie-type display in which the viewer is given the impression of a beating heart and the clinician is allowed to evaluate, subjectively, cardiac size and wall motion uniformity. The opportunity to obtain an overall subjective impression of cardiac function is also provided. This study is a perfect example of the ability to obtain a relatively large amount of clinical information from a computer-augmented study without a great deal of sophisticated quantitative analysis software that depends on a programmer's ability to anticipate every eventuality in the clinical study.

Ventricular ejection fraction is obtained from gated study data by calculating the relative difference between the volume of end-diastolic and end-systolic ventricular images (*11*). Simple ejection fraction, acquired by ventricle definition and background by hand drawn regions of interest (ROIs), is usually adequate and often as reproducible as any automatic analysis method. For cases, however, in which a more objective analysis or a background-corrected ejection curve is desired, it is necessary to have automatic and objective methods that define the ventricular outline. A number of programs have been written for this purpose. The basic difference between most versions of these programs is the method used for the definition of the ventricle edge (*12*).

Automatic edge detection techniques have received a great deal of attention in computer analysis of clinical images. In an attempt to provide objective and automatic methods for organ definition in a number of different applications (i.e., renal size and function analysis), the detection of edges in a clinical image is based on a number of criteria, the most obvious of which is count level or threshold (Fig. 5). Using this criterion, pixels above the cutoff threshold intensity are considered to be inside the organ of interest and those below to be outside. This is a good edge detection method for high-contrast images, such as in renal studies, but has problems in studies that have shifting background intensity. In this case, varying background intensities may confuse the decision because on one side of the organ of interest the background itself may be above the threshold, a situation that is often found in cardiac imaging. When this situation occurs, a second criterion can be used in making decisions about the location of the edge. The derivative of signal intensity can be used to ask questions about the rate of change of levels in the image. The inflection point in which the curve changes from concave upward to concave downward is also the place in which the derivative is a local maximum or minimum. By taking the derivative, which is the difference between successive pixels, one can locate the inflection points representing the organ edge. Determining left or right ventricle edges is usually a combination of these two techniques in that the threshold is used on the outer border while the inflection point is used in the septal region between the ventricles.

Intraventricular shunt determination is accomplished by an analysis of the time rate of activity change as a bolus of activity flows through the left ventricle (*11,13*). Once the bolus of activity passes through the right ventricle, it goes through the lungs and then back through the left heart. If there is an

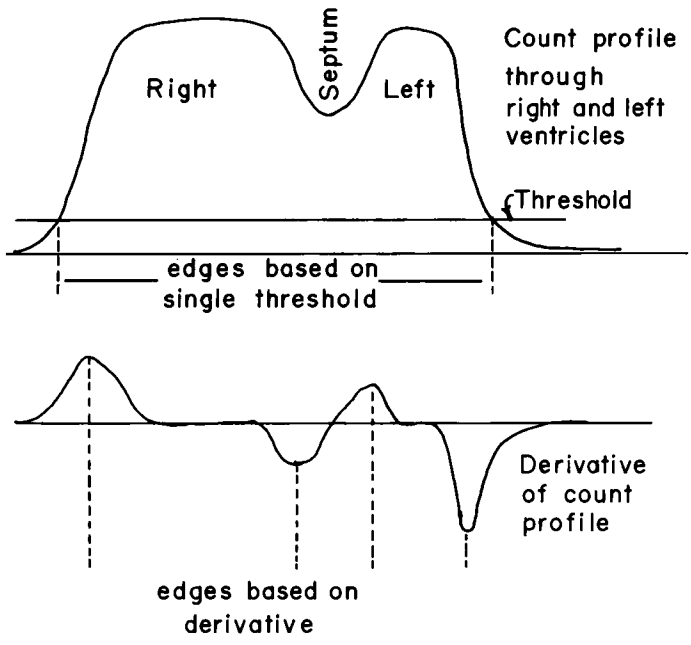

FIG. 5. Organ edges may be determined by use of a threshold count value or by more complex processing such as the derivative of the count profile.

intraventricular shunt, blood flows from the left to the right ventricle without going through systemic circulation. Existence of a shunt is indicated by the reappearance of activity in the curve obtained from a ROI placed over the lung before enough time has elapsed for the bolus to have passed completely through systemic circulation (Fig. 6). In the case of a normal heart, the downward side of the bolus curve will not be distorted by the second peak. The analysis of the lung curve for calculation of shunt magnitude consists of fitting a theoretical curve to the primary peak and subtracting the fitted curve from the original lung curve. A second theoretical curve is then applied to this difference curve. The magnitude of the shunt is specified by the ratio of pulmonary blood flow to systemic blood flow (QP/QS ratio). Pulmonary blood flow is determined from the area under the primary peak of the lung bolus curve and systemic blood flow is found by substracting the area under the second peak from that under the first. The QP/QS ratio is thus given by:

$$QP/QS = \frac{\text{counts under peak one}}{\text{counts under peak one} - \text{counts under peak two}}$$

Although this shunt determination is conceptually a simple analysis process, it is usually complicated by practical considerations such as the inability to obtain good lung bolus curve because of patient positioning, and statistical noise in the curve which interferes with the fitting of the second peak.

The computer can be used to obtain quantitative evaluation of mycardial redistribution of ^{201}Tl in the evaluation of myocardial ischemia (*14*). In this study, the patient is first imaged within minutes of the injection of the ^{201}Tl tracer, often during an exercise test. A baseline image for determination of initial perfusion of the myocardium is provided. After a delay period to allow redistribution of the thallium, the patient is imaged one or more times. Without computer support, the subjective analysis of this study consists of a visual estimation of the amount of redistribution of tracer in the scintigrams. The computer, however, not only can provide a quantitative comparison of the images, but can also realign images that may vary in exact orientation because of difficulties in patient positioning. Quantitative analysis may include the radial location of defects in the myocardial image as well as precise numerical estimates of the amount of change in the defects.

Flow-Curve Deconvolution

The use of radioactive indicators as a monitor of organ function extends back to the very beginning of nuclear medicine (*15*). The early measurements of renal or cardiac function were made using stationary probes positioned at the appropriate place on the patient's body. Probe data provided a great deal of clinical information that could not otherwise be obtained. However, uncertainties in the location of the internal organs and the exact field of view of the probes introduced significant questions about the reproducibility and accuracy of flow measurements. The introduction of the computer-interfaced scintillation camera provided a powerful clinical tool for more

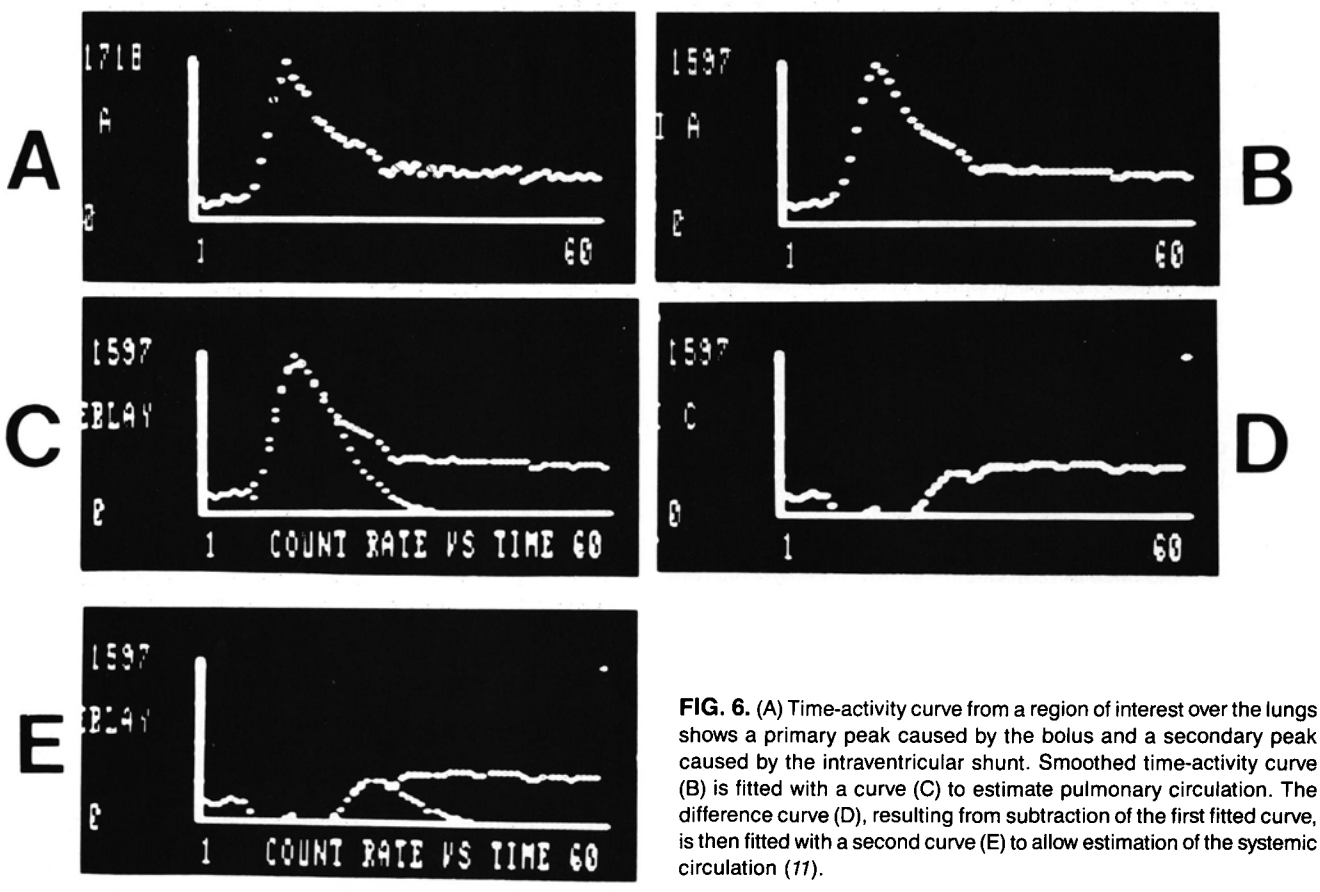

FIG. 6. (A) Time-activity curve from a region of interest over the lungs shows a primary peak caused by the bolus and a secondary peak caused by the intraventricular shunt. Smoothed time-activity curve (B) is fitted with a curve (C) to estimate pulmonary circulation. The difference curve (D), resulting from subtraction of the first fitted curve, is then fitted with a second curve (E) to allow estimation of the systemic circulation (*11*).

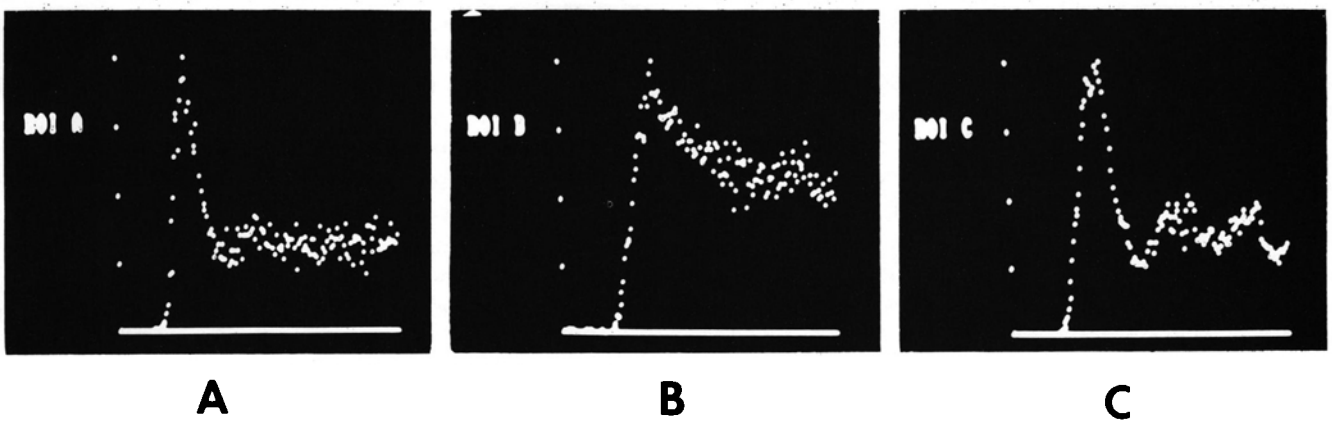

FIG. 7. The effects of curve deconvolution. Curve A is the aorta curve in a renal flow study. Curve B is the observed kidney curve. Curve C, the new kidney curve corrected for the width of Curve A by deconvolution, is narrower than Curve B (*11*).

accurate analysis of organ function and flow studies. The data collection used for functional evaluation is a dynamic study with the frame rate specified so that desired details of the organ function could be detected. For renal evaluation, one frame every 30 sec is common, whereas as many as 10 frames per sec are sometimes used for cardiac shunt studies.

Following the data collection, the region representing the organ to be evaluated is outlined with a ROI and the flow curve reconstructed. The resulting flow curve is essentially the same as that which would be collected from a probe. This curve is not directly representative of the organ function because the bolus used to produce the curve is usually introduced into the patient at some distance from the organ of interest. If a very narrow bolus is introduced into any organ system, the activity in the organ output will have some finite width and shape that is determined by the function of the organ. If the input bolus is already distorted by passage through other organ systems, the shape of the output curve from the organ of interest will be a combination of input bolus distortion and organ function. For a renal study, the injection of the bolus in the patient's anticubital vein means that the activity must move through a significant portion of the blood stream, the heart, and a number of other blood vessels before reaching the kidney. The practical effect of this procedure is to expand the bolus from the short, sharp spike of activity at the injection site to a fairly broad, smoothed out bolus at the entrance to the kidney. In order to properly evaluate kidney function, it is necessary to compensate for this extended bolus by a mathematical process called deconvolution, similar in concept to the mathematics of image filtering (Fig. 7). It is equivalent to filtering the organ output curve with a filter that has been designed so that its application to the input bolus would result in the bolus of zero width (*11*).

The process of performing such a deconvolution in clinical patient studies is not as easy as this short discussion would have one believe because a number of factors complicate the problem. Statistical uncertainty in both the bolus curve and the organ output curve causes mathematical failure in some cases. Furthermore, the determination of the input bolus shape is sensitive to the size of the ROI and its relative location to anatomical structures that may also contain radioactive tracer.

Single Photon Emission Computed Tomography

The true power of the computer is probably best exemplified

FIG. 8. Tomographic images are reconstructed from count profiles generated during data collection by using a process of filtered backprojection. Filtering of collected data prior to backprojection reduces background that results from simple backprojection process and increases image contrast.

by the tomographic capabilities provided by single photon emission computed tomography (SPECT) imaging systems (*16*). The ability to provide high contrast images of internal organs without the interference of overlying activity distributions gives the clinician the capability of evaluating low contrast structures that might otherwise be obscured.

The data collection process for SPECT imaging, discussed in the first of these articles (*1*), generally consists of the collection of a number of routine planar images (32-128) taken at sequential angular positions about the patient. These planar images are then processed to produce tomographic views. Although a number of reconstruction methods have been used in the past, the one most commonly used today is filtered backprojection. This reconstruction is illustrated in figure 8 with a simple round source of activity. If the activity is imaged with a scintillation camera at the four orientations shown, then profile lines drawn through each of the camera images at the source level will have the shapes shown, where a deflection from the base line indicates an increased number of detected counts.

The simplest reconstruction procedure would be to simply spread (backproject) the counts from each of these projection lines back along the path from which they could have come. Because there is no *a priori* basis for believing that one location along this backprojection line is more likely to be the source of the counts than any other, this uniform redistribution of counts, as illustrated by the striped bars in figure 7 is as reasonable as any other distribution. This simple backprojection only results in a very rough approximation of the original circular source with a relatively high background "star" effect. However, if the original raw data collected in the imaging process is first filtered so that each of the projections has a small negative component on either side of the peak, then the backprojection will produce an image as shown in the figure. In this case, the negative components of one projection tend to partially cancel the positive components from other projections, resulting in a much more faithful reproduction of the circular source and a reduction in the significance of the background star artifacts. Although clinical tomography uses a larger number of planar views, the concept of tomographic image reconstruction is identical to this primitive example.

The choice of what filter to use for modifying the data prior to backprojection is dependent on a number of factors. The filter may be adjusted by the operator to enhance either spatial resolution, with attendant increase in counting noise, or enhance low contrast lesion detection with a corresponding reduction in image noise. Figure 9 shows tomographic

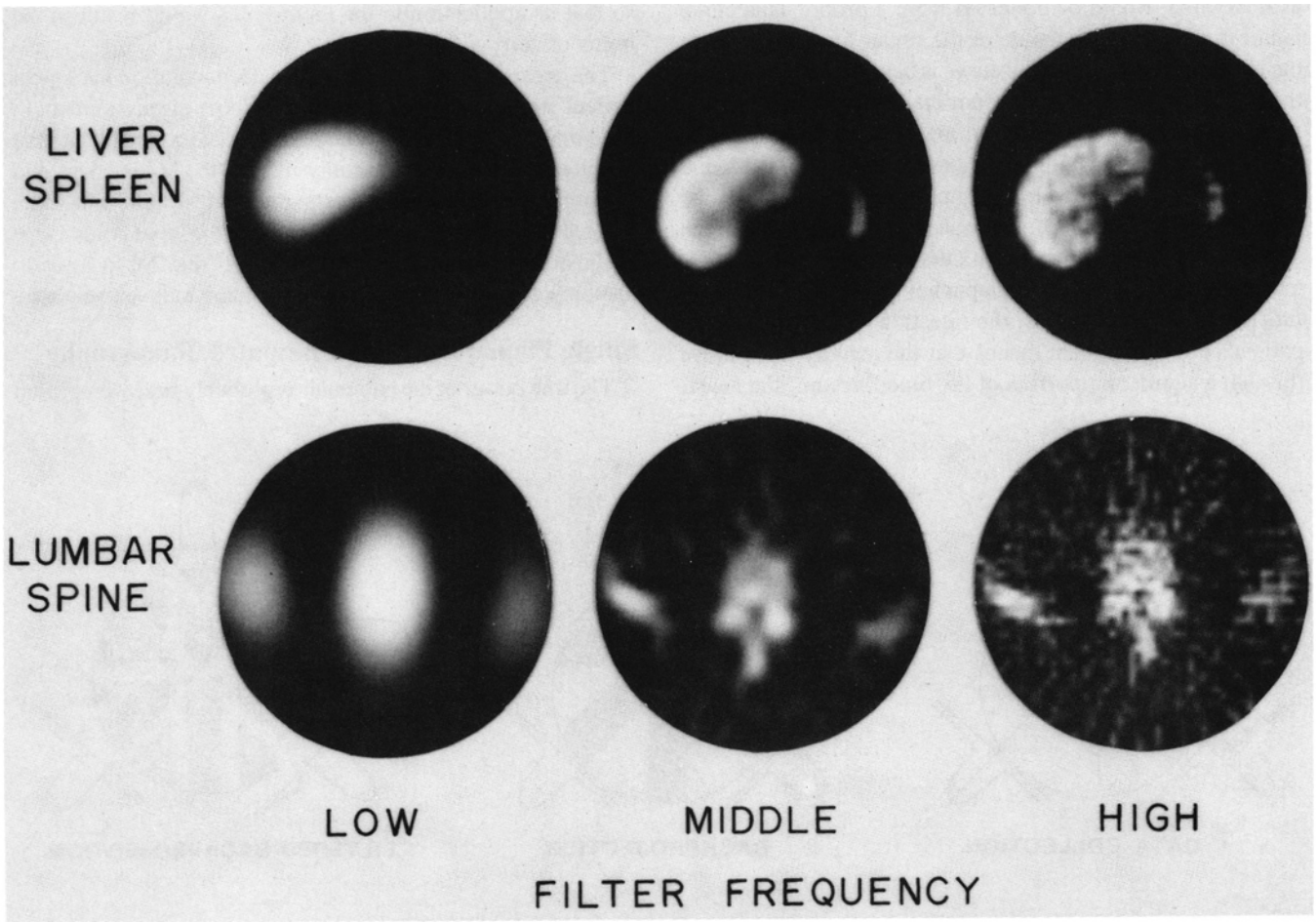

FIG. 9. The quality of the tomographic image is strongly affected by the operator's choice of filter frequency response.

reconstructions with three different filters. Because the method of chosing filter characteristics is strongly system-dependent, comprehensive advice cannot be provided in general discussion. For discussions on filter choice and the multitude of other practical problems encountered in the actual implementation of this imaging procedure, the reader is referred to the published literature (*17,18*).

SUMMARY

Computer software is a difficult topic to discuss in a general article because the word "software" has so many different meanings to different people. To those interested in system design, the discussion of operating systems and the nuances of device control are much more interesting than the use of the computer in mere clinical applications. Whereas, programmers are often more interested in the techniques of writing "clean code" and basic algorithms used in various clinical applications, clinicians and technologists who operate computers in a confusing and distracting environment may be concerned with "What do I put in, in order to get the answer I want?" Most clinical software packages are simply variations on a basic theme and provide the user with another method of presenting data to the physician. Although this article has presented a discussion of only a small portion of the software available in current clinical systems, particular topics were chosen, not for their relative importance in clinical nuclear medicine, but as a basis for understanding the general concepts of image processing and flow curve analysis.

JON J. ERICKSON, PhD
Veterans Administration Hospital
Nashville, Tennessee

REFERENCES

1. Erickson JJ. Nuclear medicine computer systems—hardware. *J Nucl Med Technol* 1985;13:97-102.

2. Pickens DR. The operating system. In *Digital Nuclear Medicine,* Erickson JJ, Rollo FD, eds, Philadelphia, JJ. Lippincott, 1983:60-80.

3. Holman BL, Parker JA. Operating systems. In *Computer-Assisted Cardiac Nuclear Medicine,* Holman BL, Parker JA, eds, Boston, Little, Brown and Company, 1981:249-62.

4. Welsh J, Elder J. Introduction to *PASCAL.* Englewood Cliffs, NJ, Prentice-Hall International, 1979.

5. Brown DW, Kirch DL, Reyerson TW, et al. Computer processing of scans using Fourier and other transformations. *J Nucl Med* 1971;12:287-91.

6. Price RR. Static image processing. In *Digital Nuclear Medicine,* Erickson JJ, Rollo FD, eds, Philadelphia, JJ. Lippincott, 1983:94-104.

7. Miller TR, Sampathkumaran KS. Digital filtering in nuclear medicine. *J Nuc Med* 1982;23:66-72.

8. Holman BL, Parker JA. Cardiac imaging procedures. In *Computer-Assisted Cardiac Nuclear Medicine,* Holman BL, Parker JA, eds, Boston, Little, Brown and Company, 1981:263-310.

9. Erickson JJ. Nuclear medicine computer hardware. In *Digital Nuclear Medicine,* Erickson JJ, Rollo FD, eds, Philadelphia, Lippincott, 1983:47-59.

10. Holman BL, Parker JA. Data collection. In *Computer-Assisted Cardiac Nuclear Medicine,* Holman BL, Parker JA, eds, Boston, Little, Brown, 1981:311-30.

11. Jones JP. Clinical-imaging studies. In *Digital Nuclear Medicine,* Erickson JJ, Rollo FD, eds, Philadelphia, Lippincott, 1983:105-54.

12. Cahill PT, Knowles RJR, Tsen O. Software reliability: Edge detection algorithms. *J Nucl Med* 1981;22:P14 (abstr).

13. Greenfield LD, Bennett LR. Detection of intracardiac shunts with radionuclide imaging. *Semin Nucl Med* 1973;2:139-52.

14. Garcia EV, Maddahi J, Berman DS, et al. Space/time quantitation of thallium-201 myocardial scintigraphy. *J Nucl Med* 1981;22:309-17.

15. Taplin GV, Meredith OM, Kade H, et al. The radioisotope renogram. An external test for individual kidney function and upper urinary tract patency. *J Lab Clin Med* 1956;48:886.

16. Patton JA. Emission tomography. In *Digital Nuclear Medicine,* Erickson JJ, Rollo FD, eds, Philadelphia, Lippincott, 1983:163-77.

17. King MA, Schwinger RB, Doherty PW, et al. Two-dimensional filtering of SPECT images using the Metz and Wiener filters. *J Nucl Med* 1984;25:1234-40.

18. Freeman LM, Blaufox MD, eds, Emission computed tomography. I. *Semin Nucl Med* 1980;4;321-403.

ADDITIONAL READINGS

Esser PD, ed. *Digital Imaging: Clinical Advances in Nuclear Medicine.* New York, The Society of Nuclear Medicine, 1982.

Cradduck TD, Busemann-Sokole E. Computers in Nuclear Medicine. *Radiographics* 1985;5:51-82.

Esser PD, ed. *Functional Mapping of Organ Systems.* New York, The Society of Nuclear Medicine, 1981.

Price RR, Gilday DL, Croft BY, eds. *Single Photon Emission Computed Tomography and other Selected Computer Topics.* New York, The Society of Nuclear Medicine, 1980.

Answer Sheet to Self-Assessment Quiz

1. d	7. a	12. b	17. c	22. a
2. c	8. b	13. d	18. b	23. d
3. d	9. d	14. b	19. c	24. c
4. b	10. c	15. c	20. c	25. b
5. a	11. c	16. d	21. b	26. d
6. a				