Image Analysis and Teaching the Concept of Function

Steven L. Tanimoto
University of Washington
Department of Computer Science and Engineering
Box 352350
Seattle, WA 98195, USA
tanimoto@cs.washington.edu

Abstract: The concept of function is key for understanding algebra and all higher mathematics. Using digital images and image processing, it is possible to introduce the concept of function in new ways which has the potential to be more compelling and effective for some students and which can better reflect the spirit in which functions can be applied in contemporary technology. We describe how two programs for PCs and Macintoshes, called the Pixel Calculator and the Image Warper, can support the exploration of function from a variety of viewpoints. While a digital image itself can be considered as a function, the many other kinds of functions which arise in image analysis support discussions of invertibility, composition, and design of functions for specific purposes. This approach encompasses an important aspect of functions: that they are objects to be *designed* and *applied* as well as objects to be contemplated and manipulated with formalisms. The materials and suggestions reported here represent one aspect of an ongoing project that explores applications of image processing in mathematics education.

Introduction

This paper presents an aspect of applying image analysis in education. Some issues related to teaching the concept of function are discussed. Particular materials developed by the project, "Mathematics Experiences Through Image Processing" at the University of Washington are described and their applicability to the teaching of functions described along with possible additional activities using image-analysis algorithm design.

As part of an effort to improve mathematics education in the United States and in other countries (NCTM, 1989), our project has been developing materials that permit mathematical ideas to be learned in a context of connections to the real world, relevance to students, use of computer technology, and rich visual representations (Tanimoto, 1994). This project focuses on middle-school and high-school students (ages 10-18); the age of 14 is a critical one in that more students leak out of the mathematics pipeline at that age than any other.

A key mathematics concept in the middle-school/high-school curriculum is the concept of function. This concept is crucial for understanding algebra, the differential calculus, and most higher mathematics Eisenberg, 1991). While the notion of function is extremely simple, the representations, terminology, and diverse examples of functions often lead to confusion for students. Functions are typically introduced using set theory, formal descriptions of numerical domains or other notation which may complicate the teaching of the basic concept itself. The use of graphs helps some students but may confuse others (Goldenberg, 1988).

An advantage of using computer-based image processing to teach about functions is that functions can be seen as *operational*. When a function is applied to an image, the function "comes alive" in a certain sense. This can have a positive motivational effect on students. At the same time, the computational context provides additional richness for the study of functions. The distinction between mathematical functions and computational functions is typically not made in grades K-12, and students who begin their formal study of computing in college must overcome this confusion at the same time that they are learning a programming language.

Images as Functions

Let's begin by considering images themselves as functions. One often thinks of a function as a rule for answering questions. The function $f(x) = x + 1$ can thus be considered as a scheme for answering questions such as "If you give me the number 2, what number will I give you back?" Another representation of such a rule is a table of values such as that shown below.

x	f(x)
0	1
1	2
2	3
3	0

In this way a function may be viewed as a type of correspondence between questions and answers.

Mapping to and from Coordinate Pairs

By considering a digital image as a slightly more elaborate table, which provides a way to give answers to questions of the form “What value is at x = 153, y = 39?” students can come to understand that a digital image may be used as a function and that it actually is a function. A consequence of this is that an image can be used as the key to a code. Provided that the image contains pixels with enough distinct values, a pair of coordinates can be used to represent a character of an alphabet. A student decodes a pair of coordinates by looking up the pixel value for that pair of coordinates in the image, and taking the k^{th} character of the alphabet, where k is the value of the pixel. The use of codes is a standard approach to involving students in mathematical activities in an informal way.

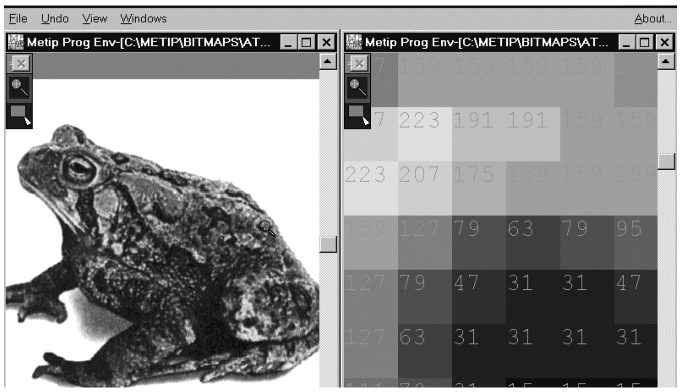
The Application of a Function

Once the student understands the concept of function as a rule for answering questions, it is important to distinguish the *application* of a function from the function itself. Where the function provides a *way* to answer questions, the application of the function is the actual *answering* of such a question. This point can be confusing, as the answer to a question could itself be an image and thus a function. An example of such a question is this: “What image results if we subtract each pixel value from 255?” (answer: a photographic negative of the original).

Pixel Calculator Operations as Functions

The Pixel Calculator

The Pixel Calculator is a program for displaying and manipulating digital images on personal computers. While its principal intended use is to introduce students to the digital representation of images, it may also be used to support discussions and exercises about the mathematical concept of function. The Pixel Calculator is first of all a display tool. As the user zooms into the image using the magnifier tool, the pixels of the image appear as gray squares, and when the squares are large enough, the numerical pixel values are overlaid upon them. In this way, the display shows both the visual and numerical aspects of digital images, and therefore it portrays a digital image as both a visual object and a mathematical object. Figure 1 shows two displays of a toad image to illustrate the combination of visual and numerical views afforded by the Pixel Calculator.



An illustration of visual and numerical displays of an image with the Pixel Calculator program.

In addition to the magnifier tool, the Pixel Calculator provides a rectangle-selection tool. With it, the user may select a rectangular region of pixels. When such a selection is made, a calculator object appears on the screen. The user may then modify the values of the selected pixels by entering values and operations on the calculator.

Operations on Single Pixels and on Multiple Pixels

One possibility when using the rectangle-selection tool is that the user clicks once without dragging and thereby selects only a single pixel. Performing calculator operations then affects only this one pixel. For example, entering # + 50 would cause the selected pixel to take on the value 50 plus its current value or 255, whichever is smaller. In this mode of use, the user is specifying a function and applying it to a single argument at the same time.

When the user has selected a rectangle containing more than one pixel, the operations entered into the calculator are applied to *all* the selected pixels. If the selection rectangle includes all the pixels of the image, then the user is causing the function to be applied to the entire image. One may consider this to be a multiple application of a pixel-oriented function or to be a single application of a different kind of function --- one that takes as its input argument an entire image rather than a single pixel value.

The Concept of Variable

A concept that is closely associated with functions and with algebra more generally is that of "variable." Mathematicians use variables for a variety of different purposes, and when variables are introduced to students for the first time, it is important to be clear on which purpose is intended. A variable may be an *independent* variable in a function definition. It may represent an *unknown* quantity to be established (in an algebra problem). A free variable may also indicate implicit universal quantification as in F(x,y) = 0. It has been pointed out (Kline, 1972), (Philipp, 1992) that the concepts of function and variable have been linked ever since they were introduced by Leibniz.

Related numbers that change together, like x and y in the above equation, are called variables. When one variable depends on another for its value, we say that it is a function of the other. (Upton 1936, in Philipp, 1992, p.239)

The most important interpretation of variable is "varying quantity." This concept is specifically designed into the Pixel Calculator. The Pixel Calculator uses the symbol "#" to mean "the current value of the pixel." The operation "# + 50" corresponds to the function f(x) = x + 50. Thus # represents a quantity that generally varies from one pixel to another, and it plays the role of independent variable in formulating functions with the Pixel Calculator. The symbol #, which looks like a pixel, is preferable in this context to the symbol x, which not only has nothing obvious to do with the value of a pixel, but which can easily be confused with the x-coordinate of the pixel.

The same symbol can represent an unknown in an algebra problem without difficulty. Consider the following question: "If the user applied the function # + 10 to a pixel and the result was 25, what was the pixel value?" Expressed with the equation # + 10 = 25, the problem can be solved in the conventional way by subtracting 10 from each side to yield # = 15. The Pixel Calculator implements an arithmetic calculus that we have named "pixel arithmetic." Among other properties, pixel arithmetic obeys truncation semantics. Thus 255 + 1 produces 255 and 0 - 1 produces 0. Therefore, the next algebra problem has a multiplicity of solutions: "If the user applied the function # + 10 to a pixel and the result was 255, what was the pixel value?" The answer is "one of the values 245, 246, …, 255." This sort of algebra problem, where the function as well as the result are given, and where it is required to find the starting pixel value, can be called "doing Pixel Calculator operations backwards."

Geometric Transformations with the Image Warper

While the Pixel Calculator supports the exploration of functions as operations on pixel values, another sort of function takes entire images as input values and produces transformed images as outputs where each output pixel may depend on the values of one or more input pixels in different locations.

The Image Warper is a program for personal computers which permits the user to specify geometric transformations on images using sets of line segments called "control lines." The program uses an algorithm reported in the SIGGRAPH'92 conference (Beier and Neely, 1992) which is part of a "morphing" method. The functions on images effected by the Pixel Calculator are specified in terms of operations on individual pixels. The pixel-wise function operates on each pixel independently and independent of the pixel's x and y coordinates. The Image Warper, on the other hand, transforms an image according to geometry—that is, according to the x and y coordinates of the pixels. After a student has warped an image by defining a set of control lines and their displacements, one may ask her/him the question, "Where is the function?" A student unfamiliar with functions as methods for transforming images may be tempted to say that the resulting image is the function, or that the pair consisting of the source and destination images represents the function. Once it is clear that the function is something that can be applied to any image of given dimensions, they may be able to realize that the function is represented by the set of control lines and their displacements. In order to save the function, one saves the control lines (together with their displacements). The lines (and displacements), together with the algorithm implemented in the program, provide a specification of the function.

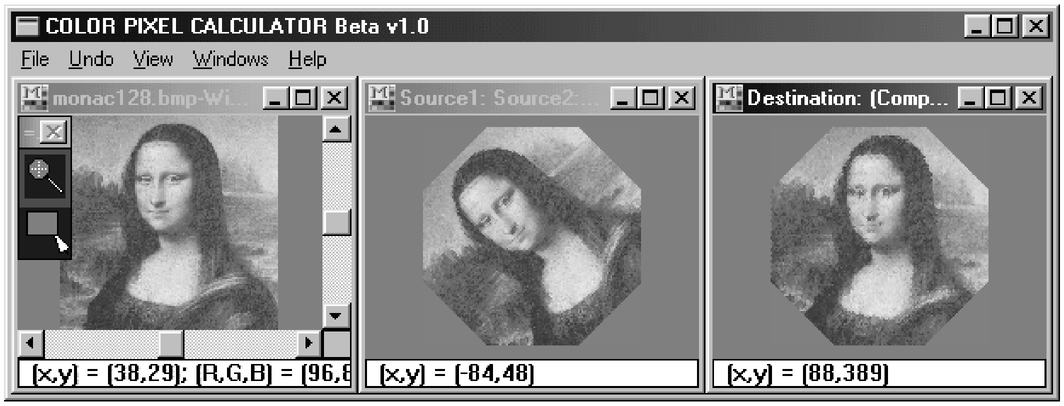
Invertibility of Functions

A useful property of functions that can be easily introduced through the image-processing context is invertibility. As mentioned above, doing Pixel Calculator operations backwards provides a simple way to pose algebra problems using the variable symbol #. The multiplicity of answers to the second problem above indicates, that it is not always possible to get back to precisely the value one started with when applying a function and then trying to go backwards. Such a discussion leads naturally into the concept of *invertibility* of functions.

While the concept of undoing the effect of a function is a natural one, the concept of *inverse* of a function is more subtle. The inverse of a function is also a function, and it must undo what the original function did to any element of the domain (e.g., any image, in the case of an image transformation). Strictly speaking, the inverse of a function must always produce the original element from the transformed element exactly.

When one considers an image itself to be a function $F: X \times Y \rightarrow \{0, 1, \ldots, 255\}$, the question arises whether such an image can have an inverse. If the image were a bijection (which would imply that the image has precisely 256 pixels), then the image would have an actual inverse, a mapping that assigns to each pixel value in the range a unique coordinate pair where that value occurs in the image. Since such images that might have inverses are so restricted and unlikely to occur if one were to use a commercial digital camera or scanner in the normal fashion, it is natural to ask whether there is some other way in which an image can have an inverse. The answer is “yes.” It is common in image analysis algorithms to require answers to questions such as “Where in the image does the value 73 occur?” A proper answer consists of a *set* of coordinate pairs. If the number 73 is not the value of any pixel in the image, then the answer set is the empty set. Otherwise, it is a set consisting of all the (x,y) locations where the pixel value is 73. This kind of inverse is sometimes called the “relational inverse” of the image. Traditional lessons on invertibility of geometric transformations such as rotation, reflection, and scaling are implicitly based on a pure notion of invertibility. Applying a transformation to an image and then applying the inverse transformation to the image must reproduce the original image exactly.

In the discrete world of digital image processing, we can also identify different kinds of *partial invertibility*. For example, a translation of an image to the left by k pixels causes data to “fall off the edge” of the image and brings in potentially undefined at the right border of the image. The subsequent translation of the image to the right by k pixels is a partial inverse for the left translation, because some of the pixels are restored to their exact original values, but values in the leftmost k columns of the image are either undefined or set to a “padding” value. In the case of rotation around the center of the image by $\pi / 4$ followed by rotation by $-\pi / 4$, there is a similar loss of data due to pixels at the corners again “falling off” the image. However, there is an additional source of information loss. The resampling that must be performed in computing the rotated image in the discrete space of pixel samples causes a loss of detail. If the original image happened to be extremely detailed, this loss might not be noticeable. However, close inspection of the pixel values is likely to indicate that the pixels are not always brought precisely back to their original values. Therefore, this illustrates a second kind of partial invertibility.



[x,y] = (38,29); (R,G,B) = (96,8
[x,y] = (-84,48)
[x,y] = (88,389)

An original image, its rotation by $\pi/4$, and the result of rotating it back.
The rotation transformation has a “partial inverse.”

These transformations which are at least partially invertible can be contrasted with image mappings that contain singularities. Such a mapping is easily specified with the Image Warper by drawing two line segments in the shape of a cross, and then rotating one of the segments by π in the destination window. The resulting transformation has a drastic effect on the images it warps.

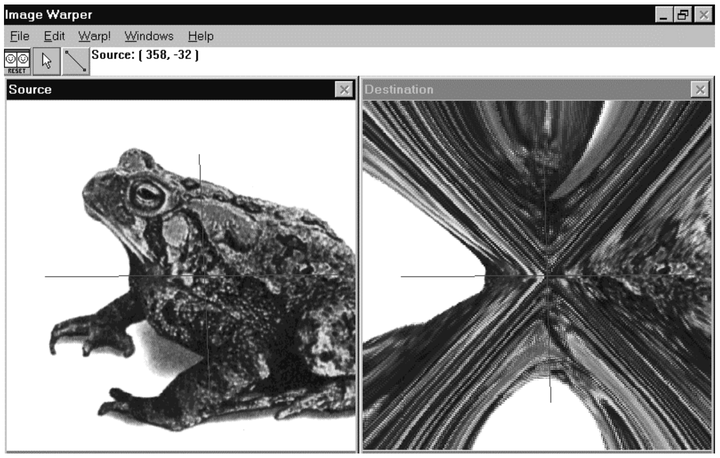


Illustration of warping with the “cross” transformation (non-invertible).

Functions as Processes

In presenting functions as process, one must note that unlike purely mathematical functions, computational functions have additional aspects such as temporal behavior and consumption of memory resources. The temporal behavior may depend upon not only the way in which the function is programmed, but the presence and behavior of other processes running on the same computer, the particular input values given, and the particular hardware that the program is running on. Computational functions are also different in that their domains and ranges are generally not the same as those of purely mathematical functions. Whereas mathematical functions often take as their domain the real numbers or vectors of real numbers, computational functions must usually limit themselves to finite representations such as double-precision floating-point values. Integers are often restricted to 32 or fewer bits.

Construction of Functions (or Functions as Designable Objects)

Mathematical functions are often conceived of as Platonic, pre-existing ideas that we may discover but not create. While it is indeed likely that any function that we can describe very simply (such as f(x) = 2 sin x) has been “created” before and therefore is not “original,” there is no reason not to consider functions as objects to be designed or created, and in image analysis and other areas of applied mathematics this is commonplace.

In order that students develop a sense that functions are objects to be designed, they need a means of expressing those designs. The basic building blocks of complicated functions are themselves functions, though they are simpler. Transformations on images may be built out of simpler transformations. Simple ones include translations, rotations, scale changes, and arithmetic operations on pixel values. Geometric distortions can be expressed in terms of mathematical functions applied to the coordinates of pixels.

Students can explore the design of functions from either the bottom-up or the top-down approach. Bottom-up creation of functions begins with some available functions and involves combining them in various ways, exploring the effects of the resulting composite functions. Top-down design is encouraged when a student is given a precise goal and asked to create a function that achieves that goal. The goal may be to correctly classify some objects or images.

In a traditional image understanding framework, the recognition process begins with the sensing of an image, its preprocessing, segmentation, feature extraction, and classification or description. If students are provided with a suitably rich vocabulary of functional building blocks, it should be possible for them to construct working image analysis systems. Two good activites in this realm are segmentation and measuring faces. The process of separating the pixels of an object (such as a person) from those of the background provides a suitable challenge for students to design a function. Working with primitives such as a thresholder, a mean-pixel-value function, a central region selector, and various parameters, a student can develop a segmentation function that will extract the shape of the person from the background, at least under appropriate conditions.

Another challenge for the student is to design a function that takes an outline of a person's face, expressed as a binary image, and comes up with a value that can be used to discriminate correctly among the faces in a given set. The aspect ratio of the face is one straightforward possibility.

Pedagogical Issues

The approach to teaching about functions outlined here may have the following benefits: an apparent relevance to students, because they can understand how functions are needed for generating special effects and for automatic pattern recognition; an aura of excitement because the functions often have striking visual effects; and an authenticity in terms of design and construction of functions, which is something very real in the engineering community. While functions have been traditionally taught in terms of correspondences and other recent approaches that have considered co-variational approaches and rates of change (Confrey and Smith, 1994), the teaching of functions as designable objects and/or as process objects with computational properties is a new approach that fits well in the context of image analysis techniques.

Software Availability

The Pixel Calculator and the Image Warper are available for download free of charge for educational purposes. For more information, see `http://www.cs.washington.edu/research/metip/`.

Acknowledgements

This project has been supported in part by National Science Foundation Grant MDR-9155709. Programmers involved in writing the Pixel Calculator and the Image Warper include James Ahrens, Jeremy Baer, Lauren Bricker, Dennis Lee, Evan McLain, and Alex Rothenberg. Thanks to Adam Carlson, Kirsten Hildrum, Jim King, Michelle LeBrasseur, and Gary Pounder for other contributions to the project.

References

Beier, T., and Neely, S. (1992). Feature-based metamorphosis. *Proc. SIGGRAPH'92*, Comp. Graph, 26, (2), 35-42.

Confrey, J., and Smith, E. (1994). Exponential functions, rates of change, and the multiplicative unit. *Educational Studies in Mathematics*, 26, (2-3), 135-164.

Eisenberg, T. (1991). Functions and associated learning difficulties. In D. Tall (Ed.), *Advanced Mathematical Thinking*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 140-152.

Goldenberg, E. P. (1988). Mathematics, metaphors, and human factors: Mathematical , technical, and pedagogical challenges in the educational use of graphical representation of functions. *J. Mathematical Behavior*, 7, 135-173.

Kline, M. (1972). *Mathematical Thoughts from Ancient to Modern Times*. New York: Oxford University Press.

National Council of Teachers of Mathematics. (1989). *Curriculum and Evaluation Standards for School Math*ematics. Reston, VA: NCTM.

Philipp, R. A. (1992). The many uses of algebraic variables. *Mathematics Teacher*, 85 (7), 557-561.

Tanimoto, S. L. (1994). Image processing in middle-school mathematics. *Proceedings of the First International Conference on Image Processing*, held at Austin, TX, 501-505.

Tanimoto, S. L. (1998). Connecting middle school mathematics to computer vision and pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*. 12, (8), 1053-1070.