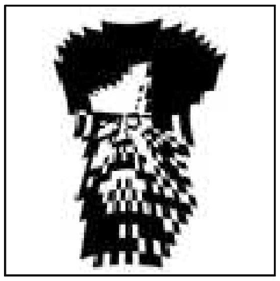
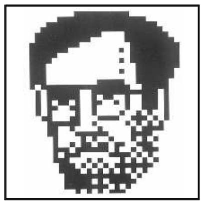


4 Altered Images

About the Photos

Somewhere toward the end of 1983 our group of about fifty researchers started using a new type of computer terminal.* Immediately, people set to work to try and exploit its graphics capabilities. Luca Cardelli (now with DEC's Systems Research Center in Palo Alto) had thought of a way to convert photos manually into little black-and-white icon bitmaps. We found a victim (a picture of computer scientist Edsger W. Dijkstra) and set to work. The process was first to reduce the photograph with a Xerox copier to a wallet-size picture of high contrast, then draw a 12×12 grid on it, make a checkerboard pattern with black-and-white squares, and then type the resulting array of dots into the computer. The result was this picture:



The EWD Icon

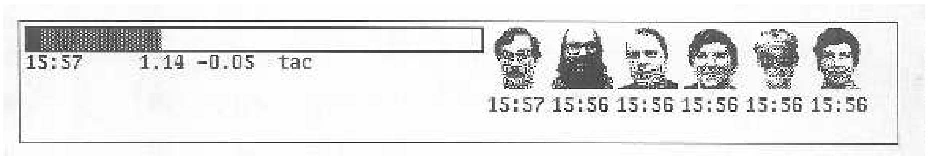
Luca made a demonstration program featuring Dijkstra's portrait as a bouncing ball. We digitized a few more portraits manually, but soon became bored with the process.

* The *blit* terminal was developed jointly by Bart Locanthi and Rob Pike. It was built and marketed by Teletype as the DMD 5620. The acronym DMD stands for "dot mapped display."

Someone then came up with the idea to make a new mail server that would announce the arrival of computer mail on our terminals by showing a little portrait of the sender. Luca Cardelli wrote a first version called *vismon* (a pun on an existing program called *sysmon*). Rob Pike talked us all into posing for 4×5 inch Polaroid portraits. Not knowing what they were in for, everybody, from secretary to executive director, cooperated with the picture project. The pictures were digitized with a scanner borrowed from our image processing colleagues. Within a few days we thus obtained close to 100 portraits that have become a main source for image processing experiments.

The next step in this sequence was when Rob Pike and Dave Presotto developed the software for a centralized data base of portraits. They called it the *face server*. The *vismon* program was rewritten to use the face server. Ever since then *vismon* has been one of the most popular programs on our computers.

The face server is used both by the mail program to announce computer mail and by a printer spooler to identify the owner of jobs sent to our laser printers. The picture below shows an afternoon's worth of mail from vismon: a police lineup of digitized faces. The bar on the left is a remnant of the original program *sysmon*. It shows the load on the computer system itself.



```
15:57     1.14 -0.05  tac
                                    15:57 15:56 15:56 15:56 15:56 15:56
```

Vismon Display

Including the portrait of new colleagues into the face server's data base is now a routine operation. A photo is digitized into 512×512 dots, then halftoned and converted into a 48×48 bit icon with a program called *mugs* written by Tom Duff. The icon is then included in the data base maintained by the face server.

As a side effect from the effort to build the face server we had suddenly obtained a data base of digitized portraits, just begging to be used for purposes other than the mere announcement of computer mail. Although my main research is not in computer graphics, I could not resist the temptation to write a program called *zunk* with which I could swap eyes and noses in portraits.

I quickly found out that people can be remarkably creative when it comes to altering portraits of colleagues. Since *zunk* was not a true picture editor I ended up adding special-purpose portions to the program for each new mean transformation that we thought of. When the number of *zunk* options was over 50 I opted for a different strategy. The language for picture transformations, shown in Chapter **3**, solved the problem and together with Rob Pike and Ken Thompson I built a picture editor called *pico* that is responsible for most of the pictures that follow. The editor itself was built in only a few weeks of frantic work. Over the years it has grown quite a bit. The language of the current version of *pico* is much richer than what we have shown so far. Special-purpose software was also added to support monitors for the real-time display of image transforms. But the basic structure and elegance of the editor has been maintained, and *pico* has proven to be an irresistible toy.

The pages that follow can be read as a *cookbook* of image transformations. Each recipe in this sequence takes two pages. The left page will show the original image with an explanation of the specific transformation used, and perhaps some intermediate images that were needed to create the final version. The right-hand page will show the finished image. Naturally, since most of the pictures in our data base are from colleagues, they will be featured most prominently here. Most transformations, however, can be applied to any image whatsoever. And you do not even have to be very precise when you try to duplicate the effects. One of the nicest things about image transformations of this type is that the result of typos and even of utter mistakes can still be quite fascinating and more often than not lead to solid improvements.

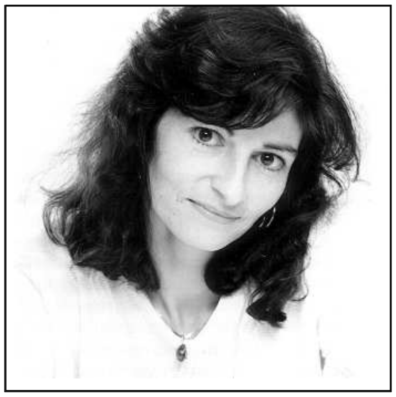
1

In the original version of this book, the victim of the first transformation was a famous image of Albert Einstein, taken by Philippe Halsman in 1947. Although the original and its transformation appeared with permission of the current copyright owners of the image, they later professed to have regretted their decision and revoked permission. In this online version, therefore, we have chosen a new victim: a portrait of system engineer Margaret Smith, defenseless for this purpose due to marriage to the author.

The unedited version is shown below. The photo was transformed with the following expression, where r and a give radius and angle of location x, y. Function *sqrt* computes a square root, and *cartx* and *carty* convert from polar coordinates to Cartesian coordinates.

$$new[x, y] = margaret[cartx(sqrt(r*400), a), carty(sqrt(r*400), a)]$$

The effect is that the image shrinks toward the center. You can think of it as a projection of the image onto a cone, with the tip of the cone in the middle of the picture.



[Page differs from the 1st edition]

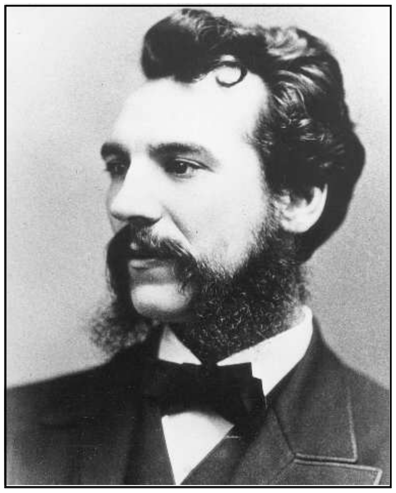
Einstein Caricature

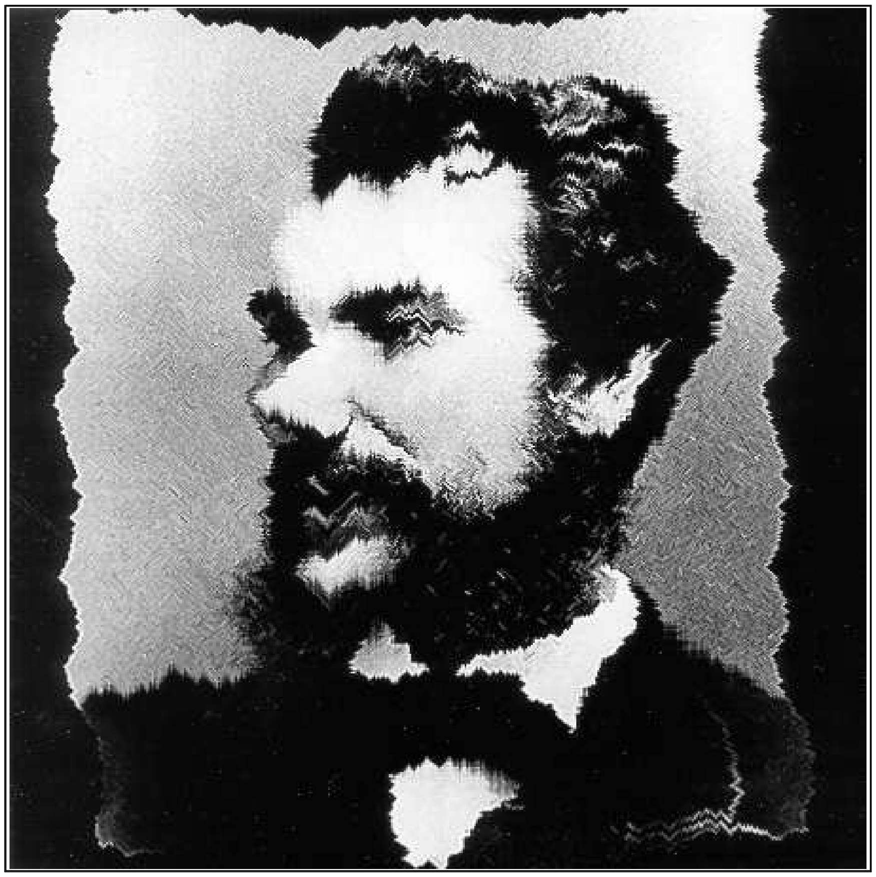
[Page differs from the 1st edition]

2

For the next transformation we use a classic portrait of Alexander Graham Bell, taken by a photographer named Holton in 1876. This strange variant of the portrait was created by randomly moving rows in the picture array left and right and by moving columns up and down. The difference in shift between two adjacent columns or two adjacent rows is never more than one dot. The transformation function is given in Chapter **6**.

The original portrait is shown below.





Bell Shear

3

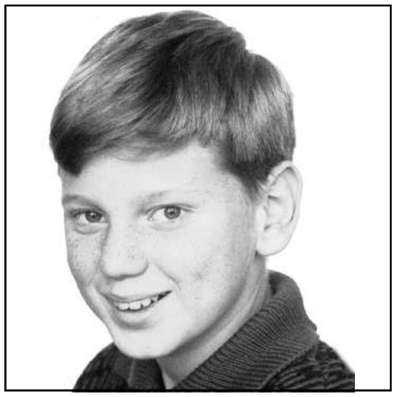
The photo on the right is an example of how smooth picture editing operations can be performed in the digital domain. No airbrush was used, no paint was needed to cover up anything in this picture. In the original version of this book, the image shown here was a composite of the solemn face of Robert Oppenheimer and the characteristic hair of Albert Einstein, both from portraits taken by Philippe Halsman. Like the original for the first image in this collection, copyright issues prevent us from using those originals here. They are replaced with a very young portrait of the author and of the even younger Tessa Holzmann, his daughter. The two portraits were first lined up and averaged, as in

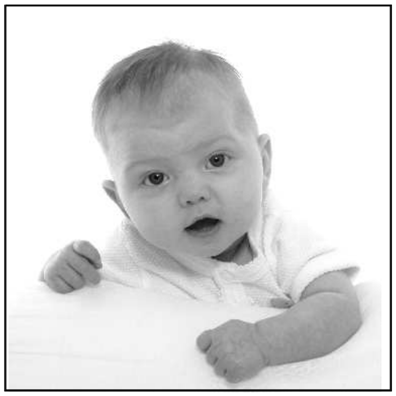
$$new[x,y] = (gerard[X-x,y] + tessa[x,y])/2$$

The picture on the left was mirrored and scaled for a better fit. The average was used to find the best points for a transition between the two pictures. Using this line a *matte* was created: a separate picture that is black where one picture is supposed to be and white everywhere else. The edge from black to white was then blurred, again digitally (Chapter **3**, equation *3.17*), into a soft slope of gray values to make the transition less abrupt. The final picture is a simple addition of the two portraits, using the matte to decide how much of each picture should be visible at each dot:

$$new[x,y] = matte[x,y]*gerard[X-x,y] + (Z-matte[x,y])*tessa[x,y]$$

The whole operation took less then an hour of my time, and mere seconds of the computer's. A routine for extracting an image matte from a picture is given in Chapter **6**.





[Page differs from the 1st edition]

Opstein

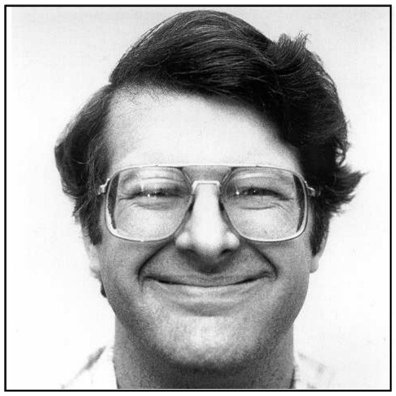
[Page differs from the 1st edition]

4

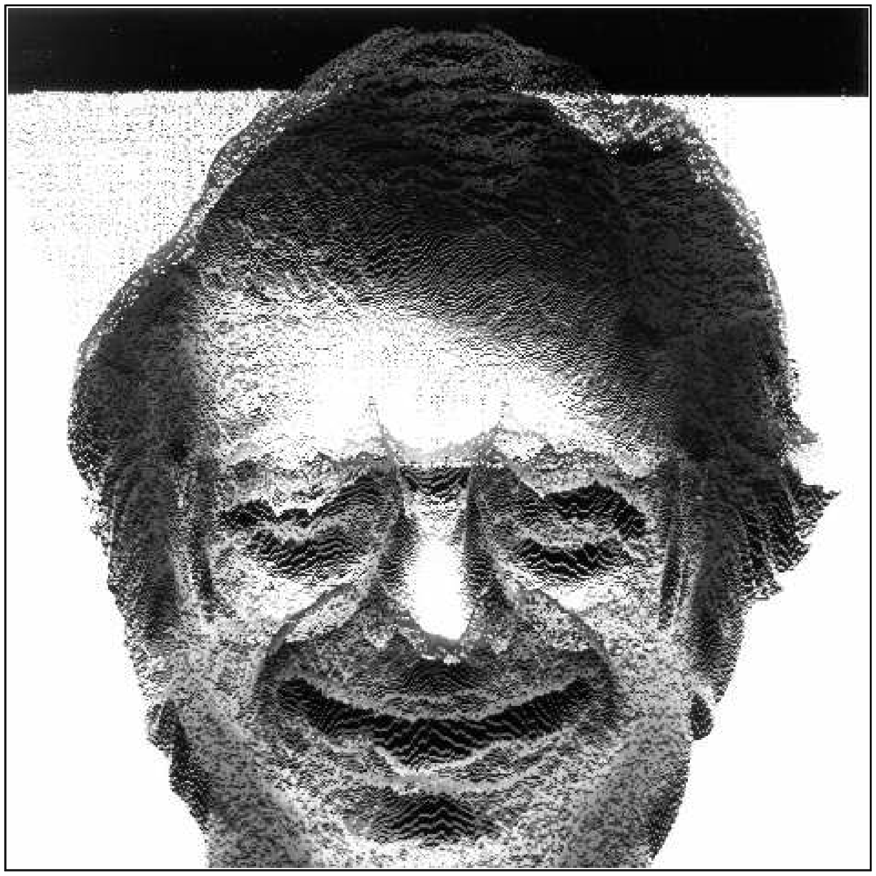
Many transformations shown here result from merely wondering “what would happen if” some strange operation is applied to an image. In this case I wondered what would happen if all the dots in the picture were shifted down by a distance that varies with their brightness. In the picture language, this is written as

$$new[x, y - jlb[x, y]/4] = jlb[x, y]$$

where this time we change the index of the destination image instead of the source. The brighter the dot, the more it moves, with a maximum shift of $Z/4$. The portrait that was subjected to this operation is Jon Bentley’s. Jon is a popular author of textbooks on program efficiency.* Here is first the original portrait, scanned in from a 4×5 inch black-and-white Polaroid picture.



* Jon’s best known books are *Writing Efficient Programs,* Prentice Hall, 1982, and *Programming Pearls,* Addison-Wesley, 1986.



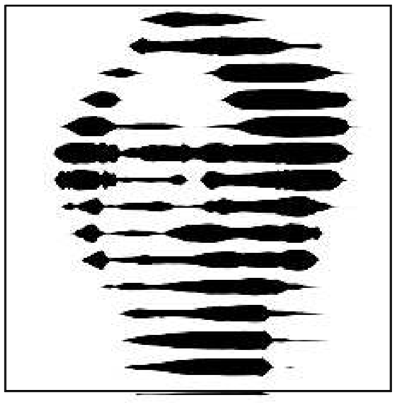
The Bentley Effect

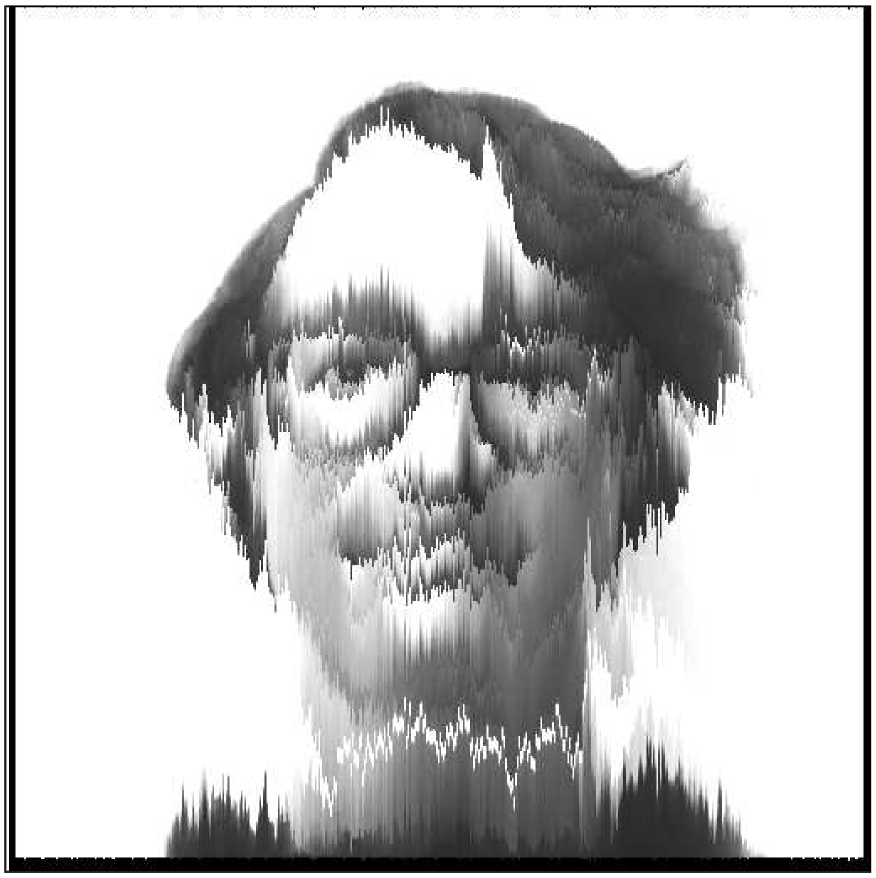
5

For some reason, the portrait of Peter Weinberger has always been our most popular target for picture editing experiments. It started a few years ago when Peter was raised to the rank of department head and was careless enough to leave a portrait of himself floating around. On a goofy Saturday evening at the lab, Rob Pike and I started making photocopies and, to emphasize Peter's rise in the managerial hierarchy, prepared a chart of the Bell Labs Cabinet with his picture stuck in every available slot. Peter must have realized that the best he could do was not to react at all, if at least he wanted to avoid seeing his face 10 feet high on a water tower. Nevertheless, Peter's picture appeared and reappeared in the most unlikely places in the lab.

Within a few weeks after AT&T had revealed the new corporate logo, Tom Duff had made a Peter logo that has since become a symbol for our center. Rob Pike had T-shirts made. Ken Thompson ordered coffee mugs with the Peter logo. And, unavoidably, in the night of September 16, 1985, a Peter logo, 10 feet high, materialized on a water tower nearby. As an ill twist of fate, Peter has meanwhile become my department head at Bell Labs, which makes it very tempting to include him in this collection of faces. The picture was created by randomly selecting dots and sliding them down the page until a darker dot was met. The melting routine itself is included in Chapter **6**.

For comparison, an unedited version of Peter's face and the Peter logo are shown below.





Peter Melted

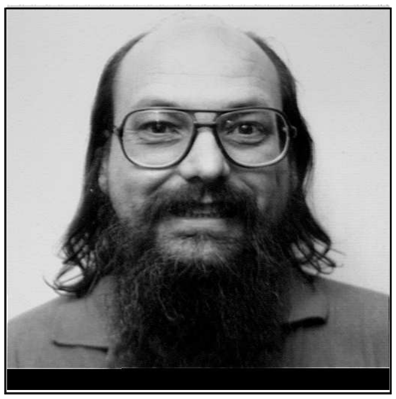
6

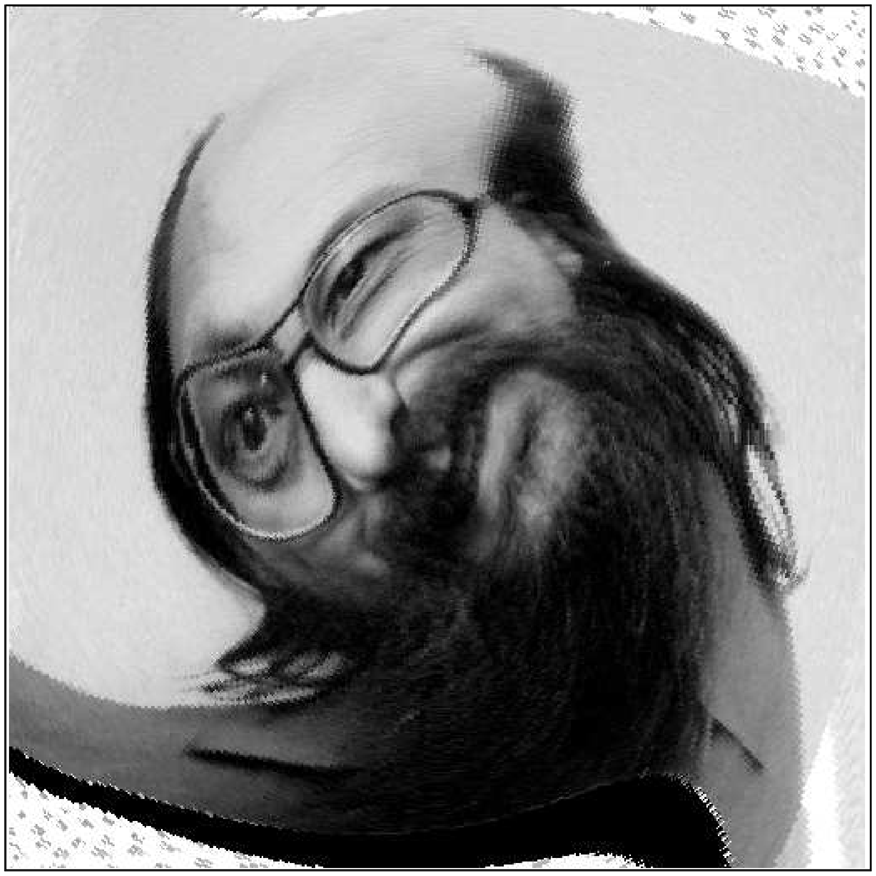
With a few exceptions the more striking transformations seem to be the ones that are particularly easy to describe. On the title page for Chapter **3** we used a portrait twisted into a spiral. Here is the expression that makes it happen:

$$new[r, a] = ken[r, a + r/3]$$

The dots in the image are again addressed with polar coordinates. The angle *a* is incremented with a third of the radius *r*, thus making the edges swirl around the center of the picture.

The operation is applied to a portrait of Ken Thompson. Ken has left his mark at Bell Labs with the development, together with Dennis Ritchie, of the UNIX® operating system. Needless to say, the picture on the right does him no justice. His real unswirled portrait is shown below.

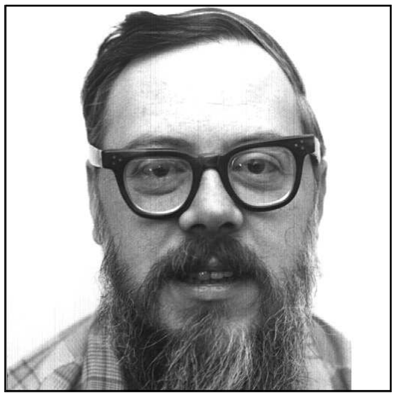




Ken Thompson

7

The painting-like effect of this transformation requires a little more work. For every dot in the image a program calculated a histogram of the surrounding 36 dots and assigned the value of the most frequently occurring brightness value. The result looks almost like an oil painting. In this case the portrait is of Dennis Ritchie. Dennis is of course best known for his work on UNIX and the *C* programming language. Locally, though, he is equally famous for his most impressive Halloween costumes. No picture transformation can achieve a comparable effect, so I won't even try: here is Dennis in oil. The transformation routine is included in Chapter **6**.



Dennis Ritchie

8

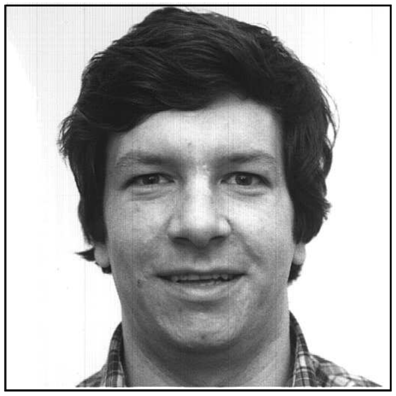
What if we took the polar coordinates of a dot and pretended they were Cartesian coordinates. That is, we use the angle a to calculate a value for the x-coordinate and the radius r to calculate the y. Let's say that we have a function $x(a)$ to cast the a into an x and $y(r)$ to cast the r into y. Adding scaling we can try the following expression:

$$new[x,y] = luca[x(a)*X/A,\ y(r)*Y/R]$$

The picture on the right is the result of this transformation applied to the portrait of Luca Cardelli, rotated by 90^o to put him upright:

$$new[x,y] = old[Y-y,\ x]$$

It would be hard to maintain that computer graphics brings out the best in people. Below is a picture of the real Luca.

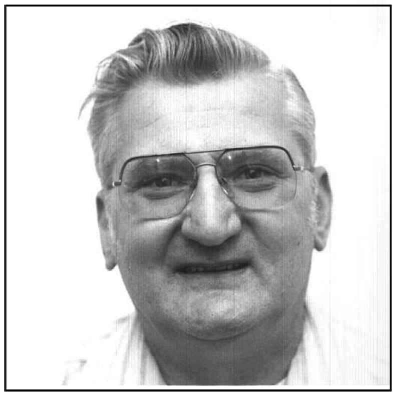


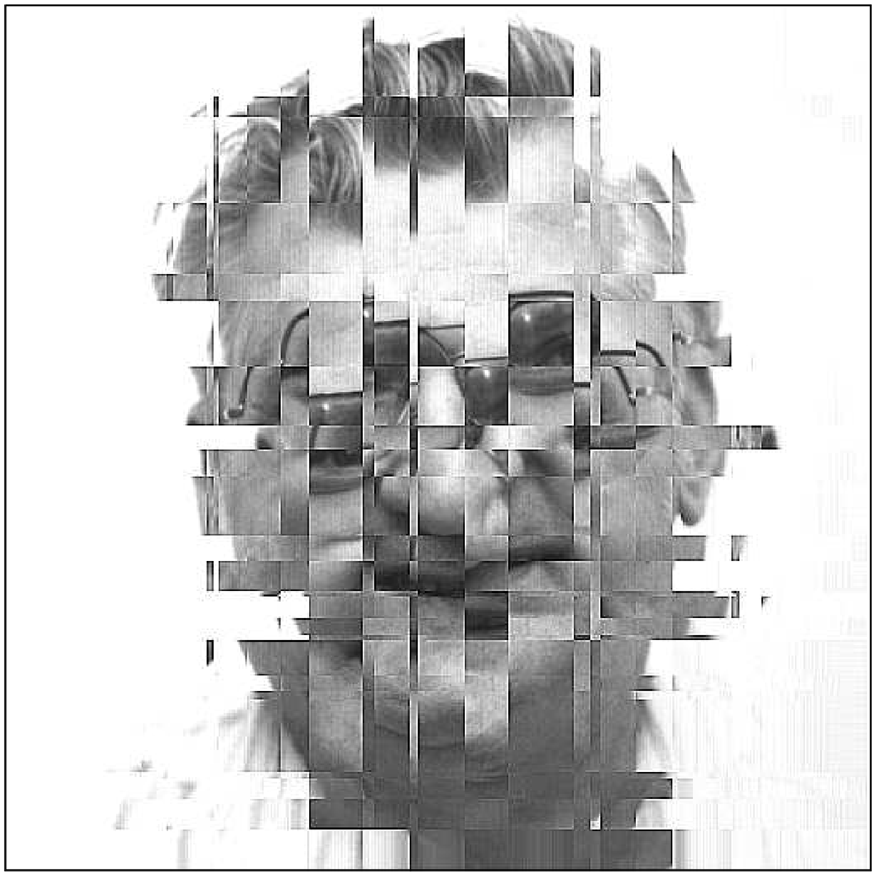
Luca Cardelli

9

The transformation shown here is similar to the one used for photo **2**. In this case a number of rows and columns are shifted at a time, with all numbers selected randomly. The amount of each shift was anywhere between 0 and 32 dots in either direction. The width of a shift was chosen randomly between 8 and 40 dots. See Chapter **6** for details.

The portrait used is that of Ed Sitar. Ed is a true magician with computer hardware and has achieved the impossible to keep machines running when they desperately want to be down. In fact, what you see on the right is probably what he feels on an average Monday when he is working on ten different major catastrophes at a time. But, in all fairness, below is a picture of what Ed really looks like.





Ed Sitar

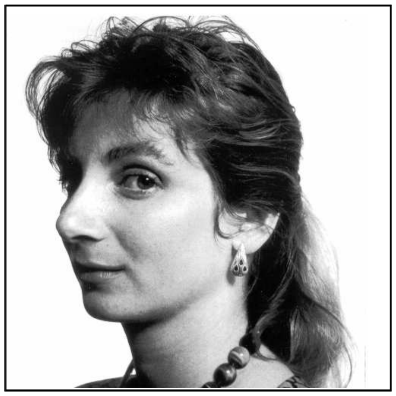
10

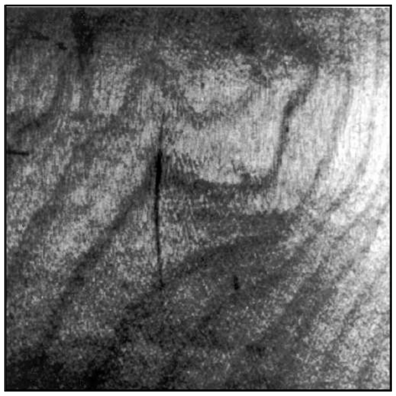
With a small library of photos of fairly standard patterns (e.g., wood grain, pebbles, brick, cloth) a range of new picture transformations becomes possible. Here we used a coarse picture of wood grain to transform a portrait of New York artist Hillary Burnett. The transformation is defined as follows:

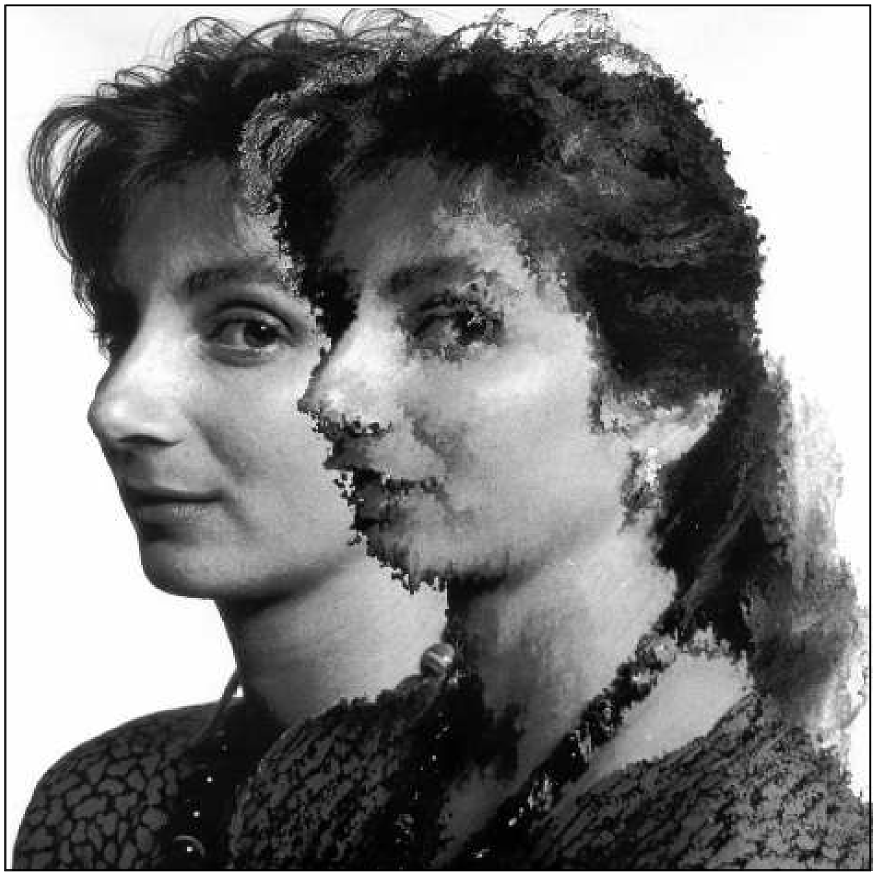
$$new[x,y] = hillary[clamp(x - pattern*F), y]$$

where F is a factor that determines how large the distortion will be at each pixel. In the picture selected F was set to 3/4, which means that each dot in the image was displaced up to $Z*3/4$ columns depending on the brightness of the pattern. *clamp* is a function that protects against overflow or underflow of the calculated x index (it always returns a value between 0 and X).

The distorted picture was combined with the original, using an image matte to define the transition, as was done in photo 1. For comparison, here is also the original portrait and the pattern that was used.







Warp

11

Here is a rather complex transformation expression that has a surprising op-art effect:

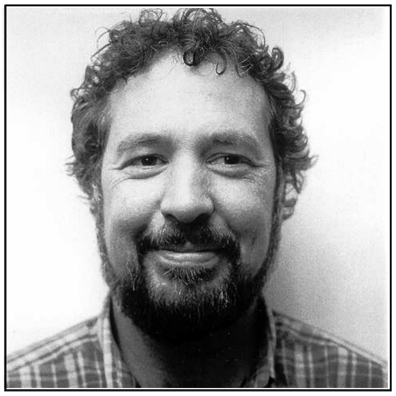
$$new[x,y] = greg[x,y]\hat{}(greg[x,y]*factor)>>17$$

where

$$factor = (128-(x-128)*(x-128) - (y-128)*(y-128))$$

A few operators in the expression will need some explanation. The circumflex operator used in the first expression is the exclusive or from the *C* programming language. The $\gg$ sign is a bitwise right shift of a value. In this case the shift by 17 positions is a fast way to get the effect of a division by 2^{17}.

Below is the original photo, a portrait of Greg Chesson. Greg is one of the contributors to the early versions of the UNIX operating system. As the transformation fittingly illustrates, Greg now lives in California.



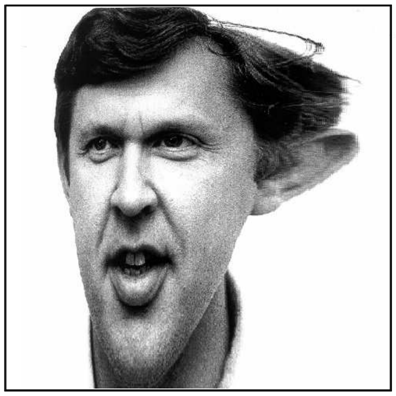
Gregory Chesson

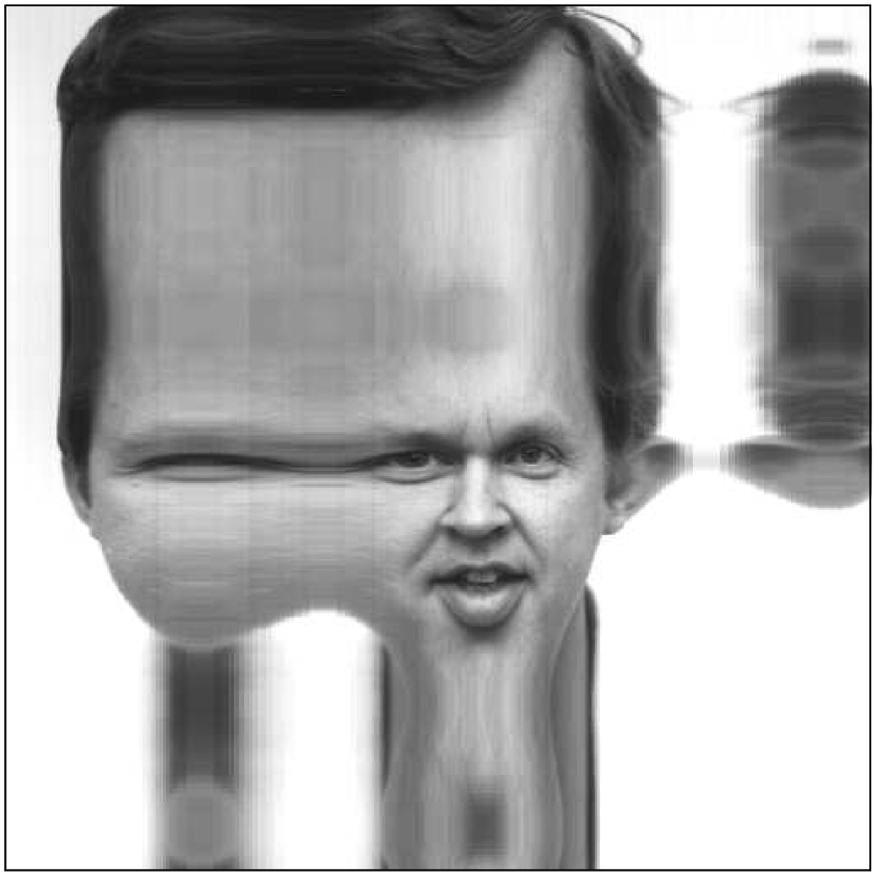
12

The mirrors in a funhouse obviously work because the mirror surface is curved. The effect can be simulated with a simple *sine* function to index an image array. We have to experiment a little to find appropriate scaling factors, for instance to control the number of curves across the width of the image or the depth of each curve.

$$new[x, y] = ava[x + sin(C1*x)*C2, y + sin(y)*C3]$$

One appealing variant of this tranformation is shown below. The factors used for the picture on the right are $C1 = 1.15$, $C2 = 160$, and $C3 = 89$. The transformation is applied to the portrait of Al Aho. Every student in computer science is familiar with Al's books on algorithms and compiler design (known as the *dragon* books; see also the list of books at the end of Chapter **5**). Here is a rare glimpse of the author in a funhouse mirror.





Al Aho

13

Well, if a funhouse mirror can be simulated with the computer, we should also be able to mimic the effect of looking through one of these wavy bathroom windows. This expression will do the trick:

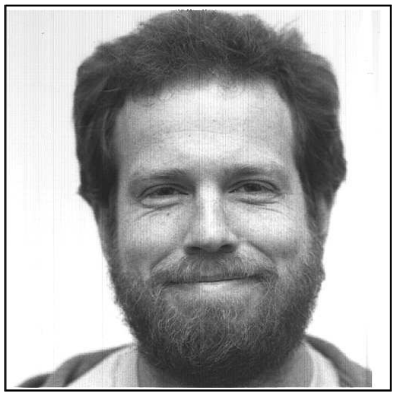
$$new[x, y] = andrew[x + (x\%32) - 16, y]$$

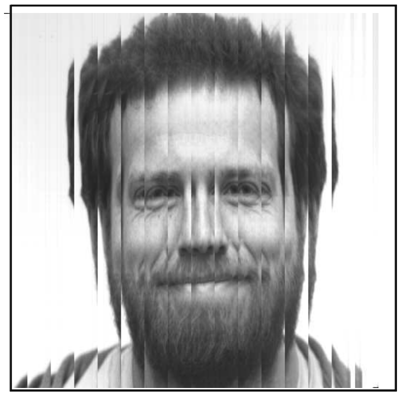
Andrew Hume is a rugged Australian programmer, known throughout Bell Labs for a peculiar wardrobe that defies the seasons (shorts, even in the dead of winter).

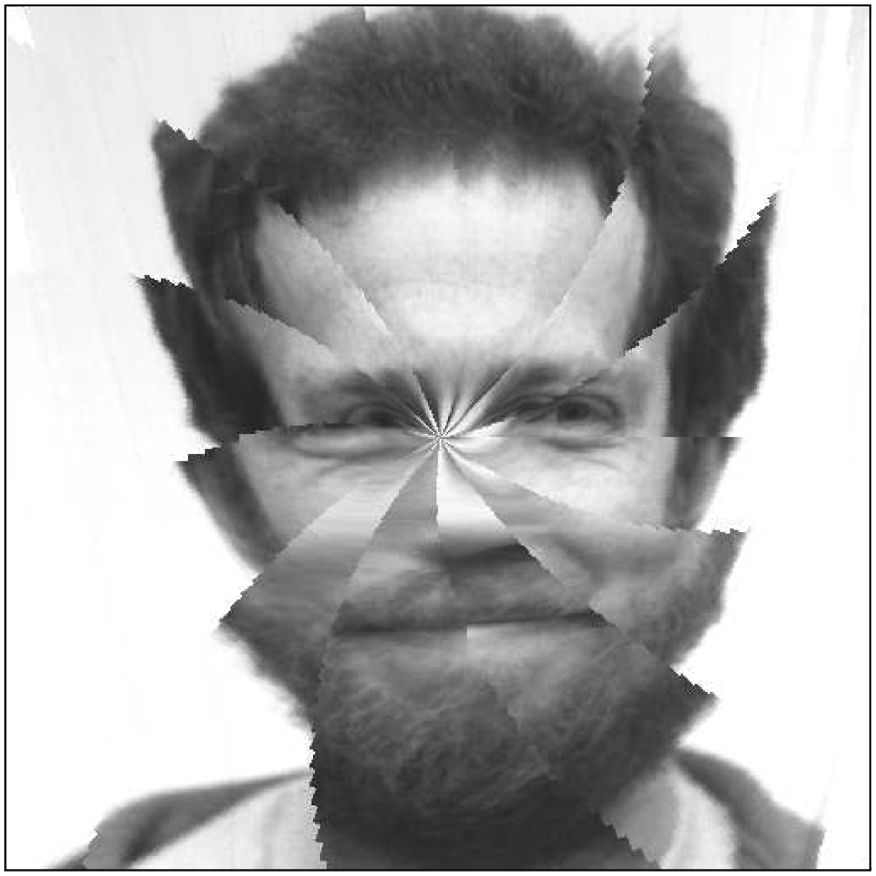
The impact of the transformation is enhanced somewhat if we add a spiraling effect, not seen in many bathroom windows.

$$new[x, y] = andrew[x + ((a + r/10)\%32) - 16, y]$$

The result is shown on the right. Andrew's real portrait and the effect of the first transformation are shown below. And, yes, Andrew likes cats.







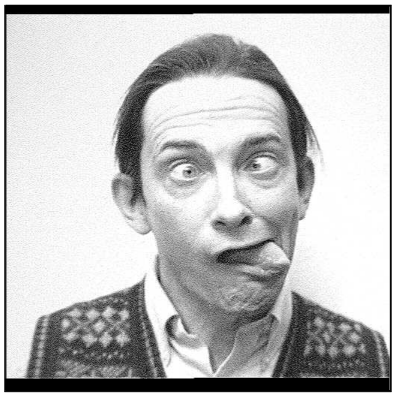
Andrew Humed

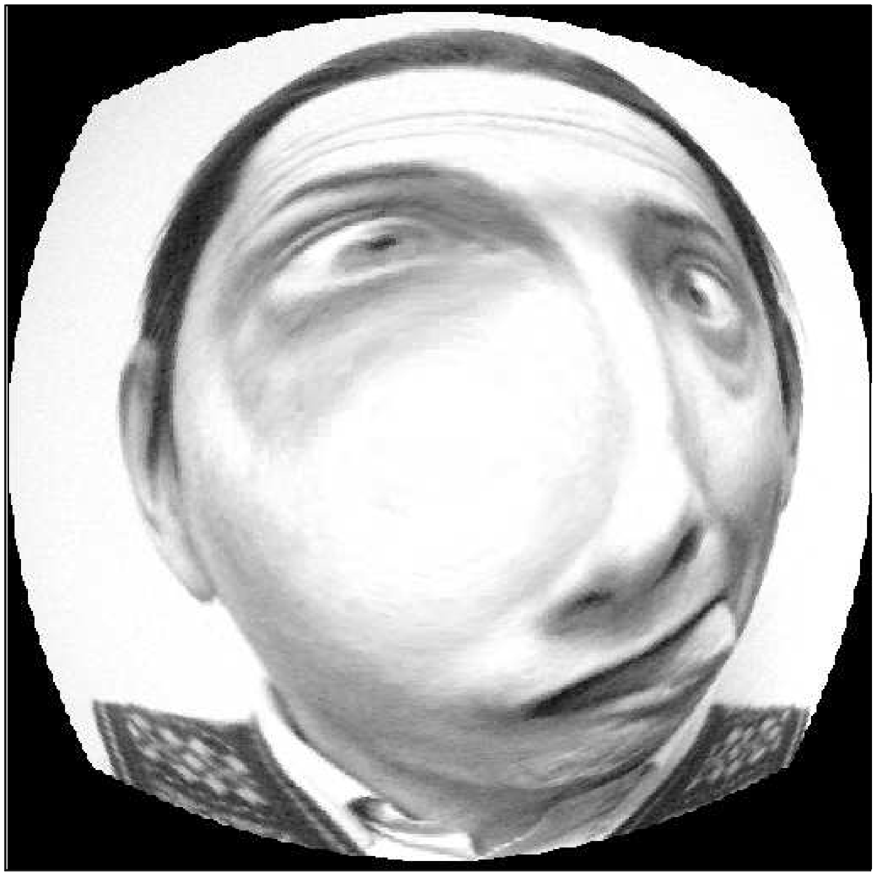
14

The obvious counterpart for the *caricature* transformation that was illustrated in the first of these images is the *fisheye*. Before our picture editor *pico* was even born, Tom Duff wrote a program that would simulate the effect of a fish-eye lens. The effect can be mimicked with the following transformation, using polar coordinates:

$$new[r,a] = psl[(r*r)/R,a]$$

The subject of this operation is Peter Langston. Peter worked at Lucasfilm for a while and is now with Bellcore. He is the author of popular computer games such as *empire* and *ballblazer*. The picture on which the transformation is based is shown below. In this case it is rather hard to decide which picture is more intriguing.



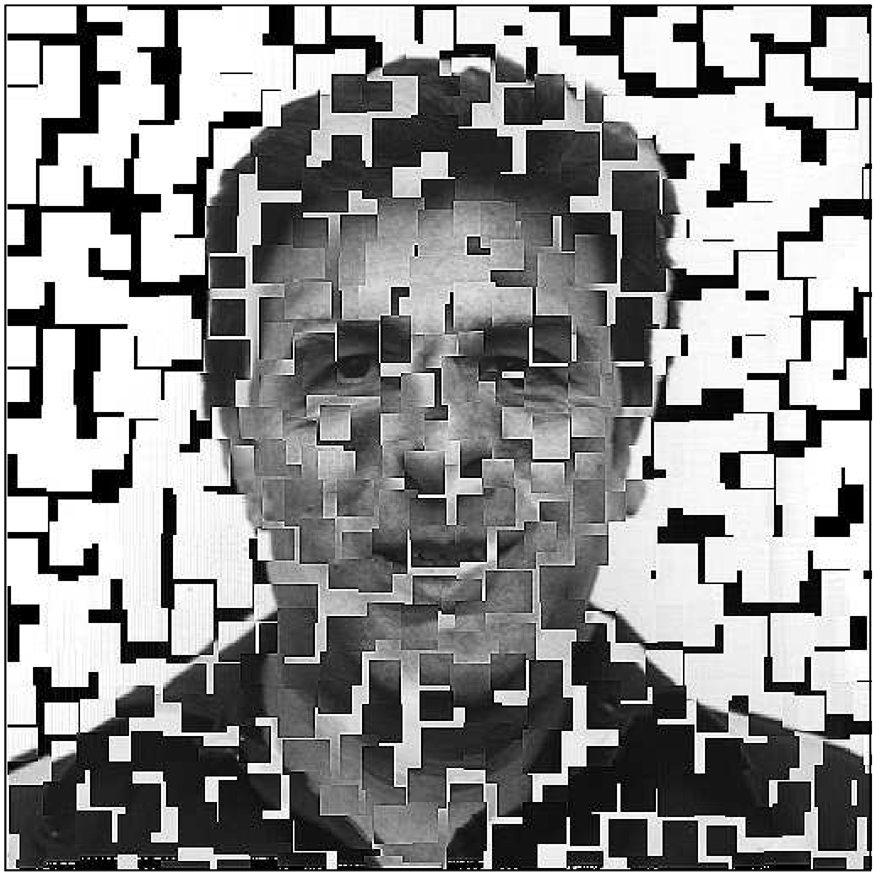


Fisheye

15

This is an example of a slightly more intricate transformation. The picture is sliced up into small squares which are moved by a random amount in the x and y direction. The background for the picture on the right is a negative of the original portrait ($new[x,y] = Z - theo[x,y]$). The tiling effect was obtained with a 10-line program written in the *pico* language. (A similar routine written in *C* is given in Chapter **6**.) The portrait is of Theo Pavlidis, known for more serious contributions to the field of image analysis and image processing.[*]

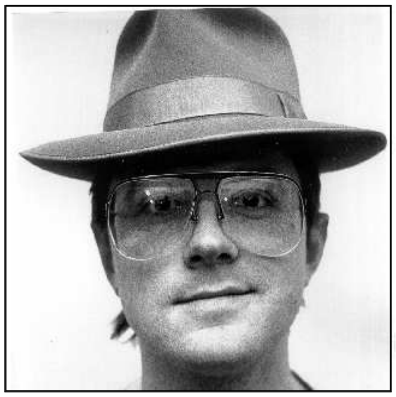
[*] For instance, *Algorithms for Graphics and Image Processing,* Computer Science Press, 1982.

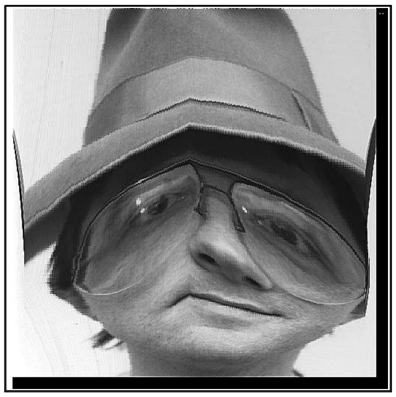


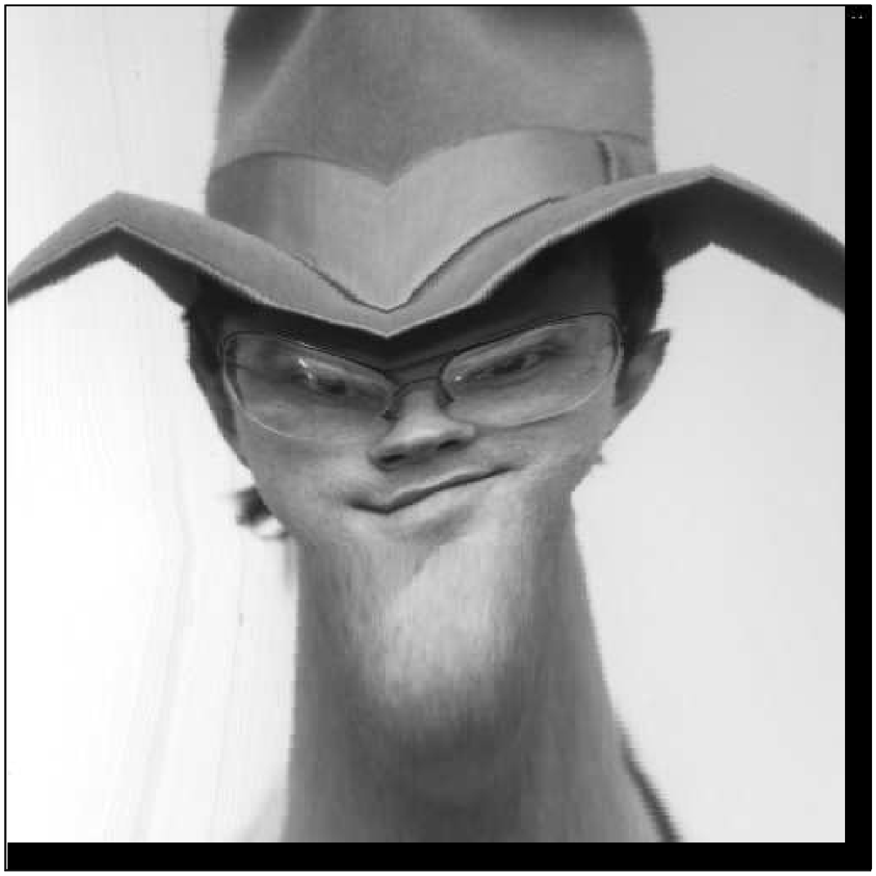
Tiled Theo

16

This is an example of the type of transformation that can be done with a *rubber-sheet* program. With such a program you can scale, stretch, and move parts of an image interactively. In this case it allows one to make either nasty or friendly variants of a neutral portrait such as Rob Pike's. Below the original and a friendlyfied variant; to the right a meaner version.





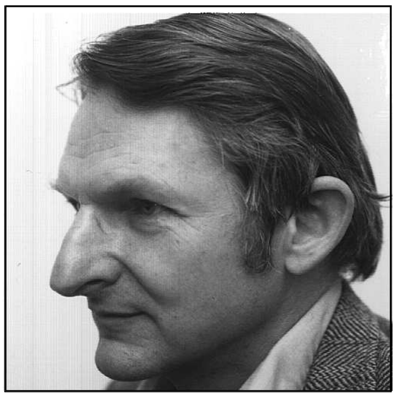


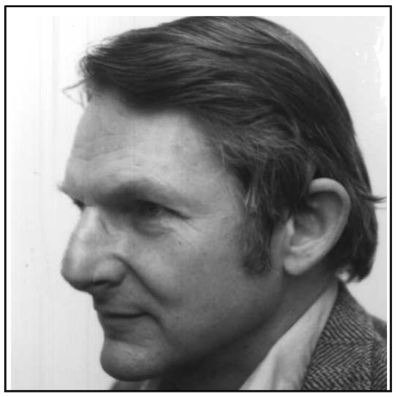
Mean Rob

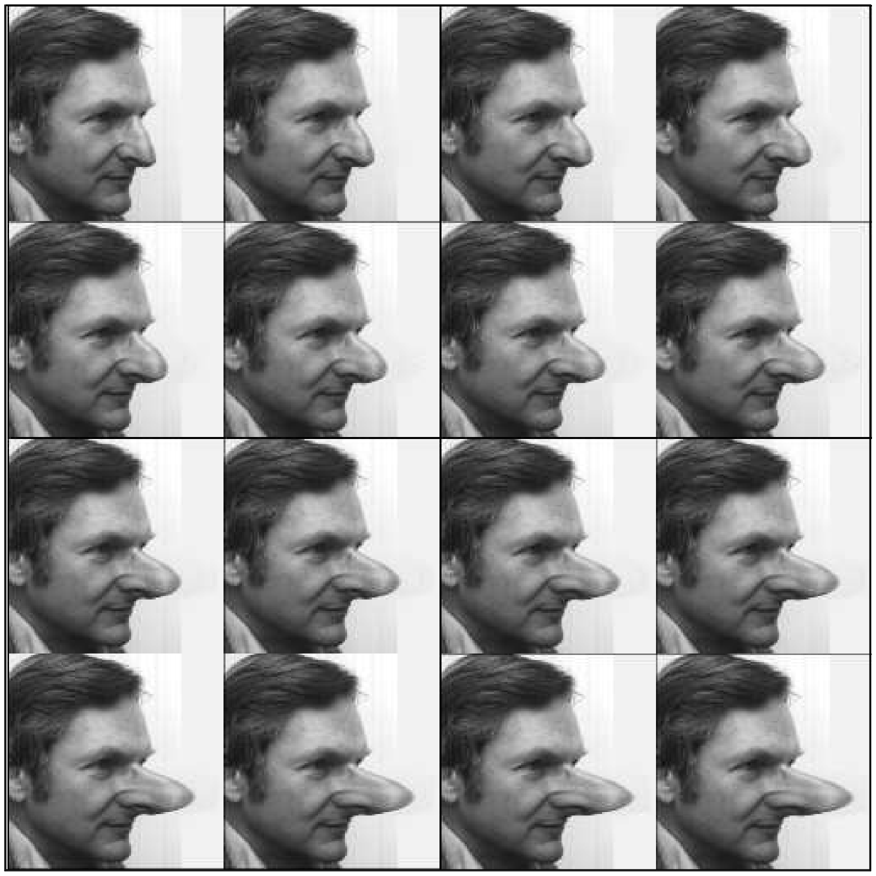
17

After 16 “ordinary” transformations we can no longer avoid showing that, yes truly, a plastic surgeon could do wonders with software like this. When I first started exploring the possibilities of image transformations I wrote a little demonstration program, called “Pinocchio,” that made the obvious change to a profile. At the time we had only two pictures in our data base with a profile: Bart Locanthi and Doug McIlroy. Since Doug McIlroy was my department head at the time, I courageously ran the program on his portrait. Sixteen frames from a little movie generated with the Pinocchio program are reproduced on the right. To my relief Doug does not hold it against me, and his children derive great joy from these pictures.

In the pictures on the right the nose was stretched. For comparison, in the small picture on the right below, the nose was shrunk. The original portrait is shown on the left. The Pinocchio images are mirrored.







Pinocchio

18

The portrait that was used for this transformation has been a more or less standard test picture in digital image processing for almost 20 years. Everybody in image processing knows the image as the *Karen* picture. Yet nobody seems to know who Karen really is. Let history record that Karen's real name is Karen Nelson. Karen was a secretary at Bell Labs in the late sixties. She left the Labs in 1969, was married, and had four children. The Karen picture was first published in the *Bell System Technical Journal* of May/June 1969.

In the picture on the right a combination of a few earlier transformations is used. This is the expression:

$$new[r,a] = karen[r,\ a + karen[r,a]/8]$$

The angle *a* is incremented with an amount that depends on the brightness of the image at that spot. The result is much like a spin-painting.

Karen

19

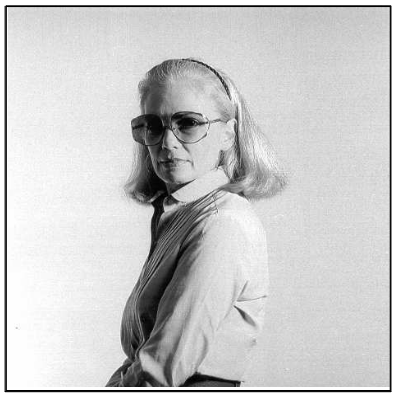
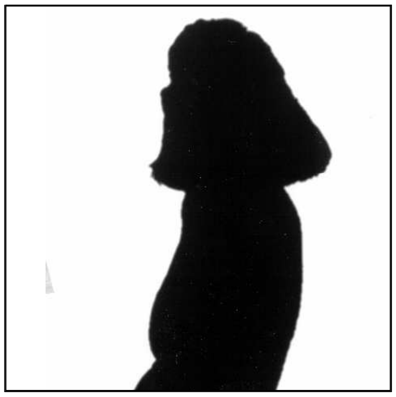
Perhaps one picture in this series is in order to illustrate that one can, of course, use the digital darkroom tools to just plainly enhance a photo without distorting it. The picture on the right is a composite of two photos and an artificial background. The original portrait is shown below. To reduce the highlight on the hair and to replace the background I made two separate image mattes (see Chapter **6**). The mattes are blurred to smooth the edges. The background is generated with the expression

new[x,y] = (x/2)ˆ(y/2)

The composite of foreground image *F* and background image *B* using *matte* (shown below) is accomplished with the following image arithmetic:

new[x,y] = ((matte[x,y]∗F[x,y] + (Z-matte[x,y])∗B[x,y])/Z

We can use smaller mattes in the same manner to darken highlights (called "burning in" in conventional photography). The portrait is of Lillian Schwartz, a seasoned pro in the art of computer graphics.

Lillian Schwartz

20

Many years ago, Leon D. Harmon of Bell Labs studied the recognizability of faces, depending on the resolution at which they were reproduced.[*] One image from his series made history: a low resolution, but perfectly recognizable, image of president Lincoln. The transformation is quite simple to mimic in the picture language.

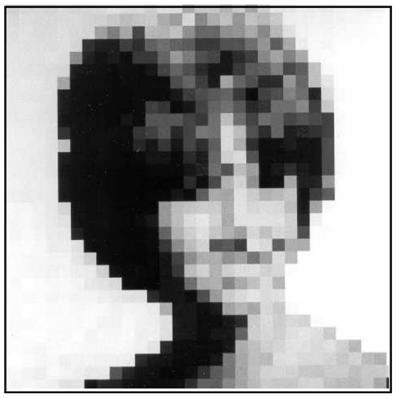
$$new[x, y] = old[(x/16)*16, (y/16)*16]$$

We use a truncation here that is implicit in integer arithmetic. Note that 156/16 = 9.75 which, when stored as an integer, truncates to 9. Therefore $(156/16) * 16 = 144$ and not 156.

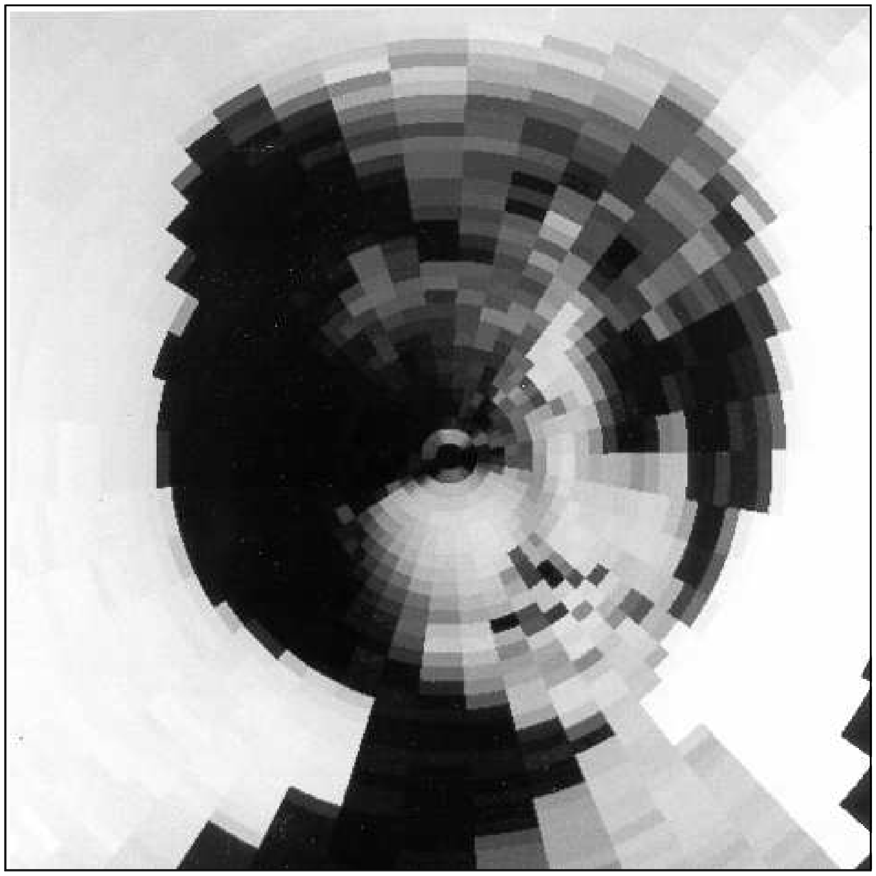
The result of the transformation is shown below, together with the original portrait of Judy Paone, secretary in our Computing Techniques Research department. Since recognizability is not really an issue in this book, we can do even better, for instance, by switching to polar coordinates

$$new[r, a] = old[(r/32)*32, (a/32)*32]$$

The result of this transformation is shown on the right.



[*] See, for instance, his article "The recognition of faces," in *Scientific Amercican*, November 1973, p. 71.



The Lincoln Transform