Image Geometrical Analogies

Adnan Jumaa Jabir

University of Baghdad \ College of Science \ Computer Science Dept.

مناظرة التحويلات الهندسية للصور الرقمية

عدنان جمعة جابر

جامعة بغداد / كلية العلوم / قسم علوم الحاسبات

Abstract

Geometric transform (G.T.) of images is a critical operation in commercial television, film producing and advertisement design. All geometric transformation operations are performed by moving pixel values from their original spatial coordinates to new coordinates in the destination image. The traditional algorithms for geometric transformation are time consuming and not accurate. With a very few exceptions, all geometric transformations result in some output pixel locations being missed because no input pixels were transformed there. This paper presents an easy-to- implement and very efficient algorithm for image geometric transform. The idea behind the proposed algorithm comes from the current work on transferring properties between images, where various types of properties (i.e. transformation filters) are to be learned from one source image – and applied to another target image. The results demonstrate the efficiency effectiveness of the proposed algorithm.

المستخلص

التحويلات الهندسية (المكانية) تستخدم في الكثير من التطبيقات، مثل: تصميم الاعلانات الضوئية و اضافة بعض المؤثرات الفنية على الصور وكذلك تصحيح بعض الاخطاء الناتجة بسبب خلل في طريقة الحصول على الصورة. ان الطرق التقليدية المستخدمة في تطبيق التحويلات الهندسية تعمل على تغيير مكان النقاط المكونة للصورة، دون التأثير على القيم اللونية للنقطة. هذه الطرق عادة ما تكون بطيئة لانها تستخدم صيغ رياضية معقدة، وكذلك تكون غير دقيقة لان بعض النقاط الناتجة من التحويل قد تكون خارج حدود الصورة.

هذا البحث محاولة لتقديم طريقة سهلة وكفوءة لتطبيق التحويلات الهندسية من دون العلم المسبق بالمعادلات الرياضية التي تمثل التحويل الهندسي. ان الفكرة الاساسية التي اعتمدها البحث، هي عملية نقل الخصائص بين الصور عن طريق تعلم التاثير من الصورة المصدر وتطبيقه على الصورة الهدف من دون الحاجة الى معرفة تفاصيل قناع التأثير. ان النتائج المستحصلة اكدت الكفاءة العالية للخوارزمية المقدمة من حيث الدقة والسرعة في تطبيق التحويلات الهندسية.

Keywords: Geometric Transformations, Image Analogies, Image Properties Transfer

1. Introduction

Geometric transformations provide the ability to reposition pixels within an image. Pixels may be relocated from their (*x,y*) spatial coordinates in the source image to new coordinates (x',y') in the destination. Each pixel of the source image is transformed, pixel by pixel, to its new location in the destination image. Geometric transformations are used to register multiple images, correct geometric distortions introduced in the image acquisition process, or to add visual effects. [Bocheck 2000]

Geometric transforms which might be called *Spatial Image Transform,* like any other image transforms take a long time to perform the transform operations. In spite of the time is acceptable for one image, it is very long when it is used for a large numbers of movie frames. Moreover, some transformations are very complex; they need a large and complex program. Another problem with geometric transform is that several geometric image operations, such as Rotate, Shear, Fish-Eye and others use a geometric transformation to compute the coordinate of a source image point for each destination image pixel. In most cases, the destination pixel does not lie at a source pixel location, but rather lands somewhere between neighboring pixels. The estimated value of each pixel is set in a process called interpolation or image re-sampling.

In recent years, much interest has been directed toward capturing properties from one image (or movie) and transfers them to another. For example, the general technique of *Hertzmann et al* – image analogies can learn various types of filtering transformations from pair of images and then apply the learned filters to other images that as a result of the process, inherit properties from the original image. In practice, image analogies involves two stages. In *design phase,* a designer creates a filter from training images *A* and A'. The designer also sets the parameters that control how various types of features in the images are weighted in the image analogy. The filter is then stored to a filter library. Finally in the *application phase,* the end user, probably without further knowledge of image processing, can apply the filter to some image [Hertzman2001].

The image analogies algorithm assumes that the colors in *A* and A' correspond to each other in the same pixel position *p*. The images need to contain the information of RGB color, luminance and various filter responses. This information combined we will have the feature vector for each pixel [Huttunen2004].

Image analogies can be used for many different applications. Applications where an image analogy has been applied so far are: *Image filters, Texture synthesis, Texture*

transfer, Artistic filters, Texture-by-numbers, super resolution and image colorization. [Huttunen2004]

Although the applicability of image analogies in a wide area of applications, it encounter some drawbacks. The biggest problem in image analogies framework is probably the slowness of the algorithm. The memory intensiveness is largely due to the use of Approximate-Nearest-Neighbors (ANN) in approximate search. Also, the feature selection is problematic since each approach fails in some situation [Huttunen2004]. Additionally, image analogies cannot be used to learn the geometrical transformations where only the pixel coordinates are affected by the transform. That is because the algorithm focused on the image colors information rather than pixel locations.

In addition to image analogies, a variety of methods exist which are specifically tailored for transferring a single property from one image to another. For example, the curve analogies of *Hertzmann* et al [Hertzmann2002] is used to learn statistical models of 2D curves, and shows how these models can be used to design line art rendering styles by examples.

Color transfer of *Reinhard* et al [Reinhard2001] used a simple statistical analysis to impose one image's color characteristics on another, a color correction can be achieved by choosing appropriate source image and apply its characteristics to another image.

Geometric texture transfer of *Lai* et al [Lai2005] presents an automatic method which can transfer geometric textures from one object to another and can apply a manually designed geometric texture to a model. It involves geometric texture extraction, boundary consistent texture synthesis, discretized orientation and scaling, and reconstruction of synthesized geometry.

Texture transfer by *Mertens* et al [Mertens2006] presents a technique to transfer such geometry-dependent texture variation from an example textured model to new geometry in a visually consistent way. It captures the correlation between a set of geometric features, such as curvature, and the observed diffuse texture.

In this paper, a new method is proposed to transfer any geometrical transform from source filtered image to the target unfiltered image. The contributions of the work are:

- *Geometric transformation filters are to be performed on the target image without any need to know in advance or to define explicitly the original transformation equations.*

- *to be easy-to-implement and user-friendly, and*
- *to provide accurate results.*

Next section presents a general introduction to geometrical filters. Then, section (3) discusses the framework of the proposed algorithm followed by results and conclusions with possible future directions.

2. Geometrical Filters

An image filter can be thought of as 'something' applied to one image in order to produce another image [Waltman2001]. There are four categories of image filtering, or processing, algorithms: point processes, area processes, frame processes, and geometric processes. Point processes alter each pixel in an image based only on the original value of the pixel. Area processes alter each pixel based on the values of the original pixel and the values of the surrounding pixels. The surrounding pixels are usually defined by choosing a radius and a square 'kernel' of pixels (with the chosen radius) around a center pixel. All pixels inside the kernel are processed in order to get a new value for the center pixel. Frame processes are used to combine images; new pixel values are determined by that pixel's relationship to one or more other images' corresponding pixel values. Geometric processes are like point processes (i.e. only one pixel value is used to determine a new pixel value) but the new pixel value is based upon some geometric transformation. [Lindley1991]

2.1. Image Deformation Model

The general image deformation model can be described as:

• Let (u, v) represent the image coordinate in an original image, and (x, y) in a deformed (or warped) image. We use a function pair to relate corresponding pixels in the two images:

- Forward mapping: $x=x(u,v), \quad y=y(u,v)$ ……. (2.1)
- Inverse mapping: $u=u(x,y), \quad v=v(x,y)$ ……. (2.2)

• Let f(u, v) denotes the original image and g(x, y) the deformed image. Then they are related by:

Inverse ◊ $g(x,y) = f(u,v) = f(u(x,y),v(x,y))$ ……. (2.3)

Forward ◊ $f(u,v) = g(x,y) = g(x(u,v),y(u,v))$ ……. (2.4)

2.2. Affine transformations:

An affine transform is a transformation of an image in which straight lines remain straight and parallel lines remain parallel, but the distance between lines and the angles between lines may change. Affine transformations include translation, scaling, and rotation.

a) Translation: An object is translated by using two parameters, tx, ty, the new coordinates are defined as: [LaValle2006]

$x' = x + t_x,\ \ y' = y + t_y$ ……………………………..………. (2.5)

b) Scaling: Changes the size of the object by multiplying the coordinates of the points by scaling factors: Sx, Sy, such that

$x' = x * s_x,\ \ y' = y * s_y,\ \ \text{where } s_x, s_y <> 0$ ……..………. (2.6)

c) Rotation: an object can be rotated counterclockwise by some angle (t), by mapping every (x, y) of object as: [LaValle2006]

$(x', y') \rightarrow (x \cos(t) - y \sin(t),\ x \sin(t) + y \cos(t))$ …….. (2.7)

d) Shearing: Changes the shape of the object.

Shear along the x-axis: $x' = x + ay,\ \ y' = y$ ………..…. (2.8)

Shear along the y-axis: $x' = x,\ \ y' = bx + y$ ……………. (2.9)

e) Transposing (flipping): It reflects an object over a line. Reflections over a line take every point to a point directly across the line, as if the line were a mirror. More precisely, the image of a point P under a reflection over line L is on the line through P perpendicular to L and is the same distance from L as P is. In still other words, the line L is the perpendicular bisector of the segment between P and its image.

2.3. Warping:

Affine transformations are linear geometric transformations, which mean it cannot introduce curvature in the mapping process. Image warping is a type of geometric transformation that introduces curvature into the mapping process. The introduction of curvature is important when an image has been distorted through lens aberrations and other non-linear processes. Warping transformations, also known as rubber sheet transformations, can arbitrarily stretch the image about defined points. This type of operation provides a nonlinear transformation between source and destination coordinates.

a) Fish-eye lens effect:

In this effect, the image appears to bulge outward, ideally, as if it was pushed out from behind using a sphere. This effect is done by a geometric transformation of pixels within a certain area. In this case, the area is the largest circle that will fit inscribed in the center of the image. It is modeled on the actual optic structure of a fish's eye that the resolution is highest in the center and degrades toward the periphery [Waltman2001]:

new_radius = s log(1 + w(old_radius)) ……. (2.10)

where w [0.001..0.1], appears to be the amount of curvature, or how much the center of the image is pushed forward. R is the maximum radius and s is calculated based on the following: s = R / log(w*R+1) ……. (2.11)

if the inverse of the fish-eye function is used, the resulting image would appear to be pushed in, instead of being pushed out. Here is the inverse function [Waltman2001]:

new_radius = (e^(old_radius / s) – 1) / w) ……. (2.12)

b) Polar coordinates effect:

The idea here is simple. Pretend that the pixels in our Cartesian-based image are actually polar coordinates and display them using the Cartesian system. That is, a pixel at coordinate (x,y) has a corresponding polar coordinate radius and angle (r,a), using the center of the image as the polar origin. Here, to find a value for the filtered image, we find (r,a), then use r = x, and a = y and plot the point in our Cartesian system. The conversion to polar coordinates from Cartesian, is based on geometry and trigonometry [Waltman2001]:

radius = sqrt(x*x + y*y) …….. (2.13)

angle = arctan(x / y) …….. (2.14)

Although not used in this filter, the conversion back to Cartesian is given by:

x = radius * cos(angle) …….. (2.15)

y = radius * sin (angle) ……..… (2.16)

c) Wave effect: The Wave effect distorts the image as if it had been disturbed by a number of random waves, as follows:

x(u,v)=u+sin(v); y(u,v)=v; ……. (2.17)

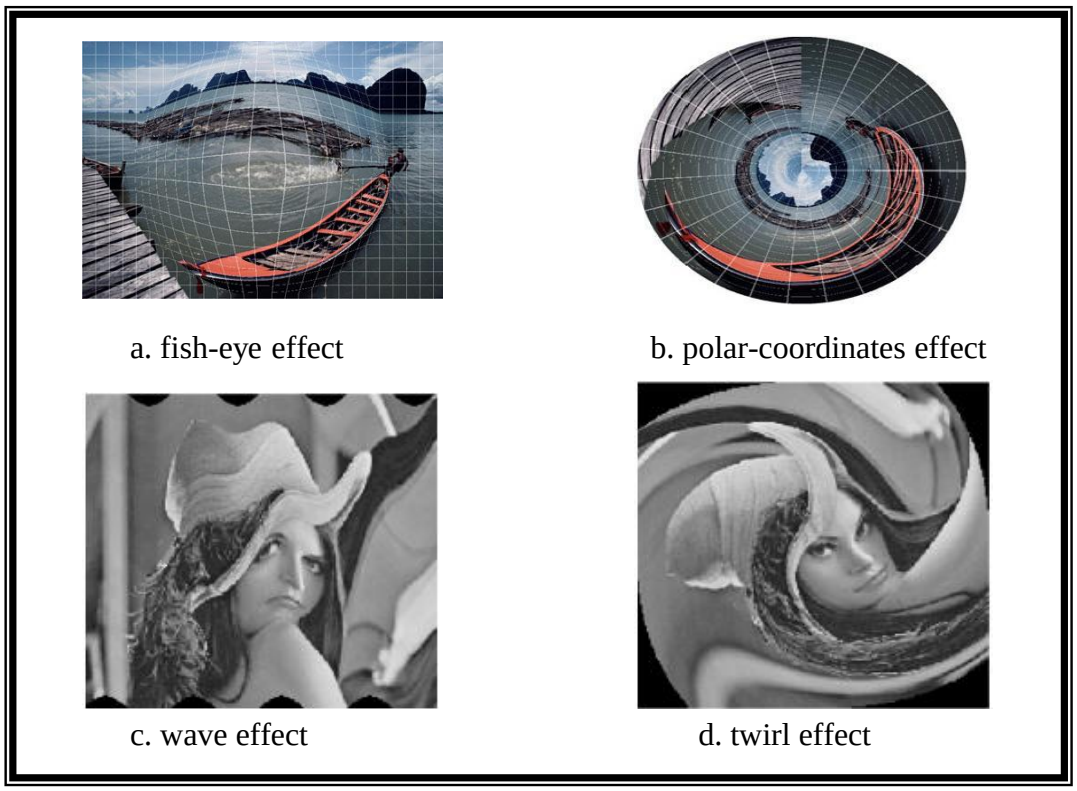
d) Twirl effect:

The effect distorts the image as if it had been twisted. Rotates a selection more sharply in the center than at the edges.

$x(u,v)= (u-x_0) \cos(t) + (v-y_0) \sin(t) + x_0$ ……. (2.18)

$$y(u,v)= -(u-x_0)\sin(t) + (v-y_0)\cos(t) + y_0 \quad \ldots\ldots\ldots (2.19)$$

where (x_0, y_0) is the center point coordinates. Figure (1) shows the fish-eye, polar-coordinates, wave, and twirl effects.



a. fish-eye effect

b. polar-coordinates effect

c. wave effect

d. twirl effect

Figure (1): fish-eye, polar, wave, and twirl effects

depending on the above transform equations, it can be seen that these transformations take a long time to complete. That is because the transformations use a time consuming functions like sine, cosine, logarithm, and exponential.

3. Framework of the Proposed Algorithm

The proposed algorithm is mainly divided into three phases: source image creation, training, and application phases. At first, the S-Image is created by setting the color value of each pixel by the pixel's coordinates, then applying any geometric filter on S-Image using any software (ex. Photoshop) to produce S'-Image. Finally this S'-image can be used to apply the learned filter to any target image.

3.1 Source Image Creation

This is the core phase of the proposed algorithm. In this phase, the S-Image is created. The main idea in S-Image creation process is to set the color components of S- pixels to their locations. This can be done easily by setting the R-components to the X-coordinates and the G-components to the Y-coordinates of the S-pixels. Formally speaking, S-Image should hold the following properties:

a) Its size equal to the size of the target image:

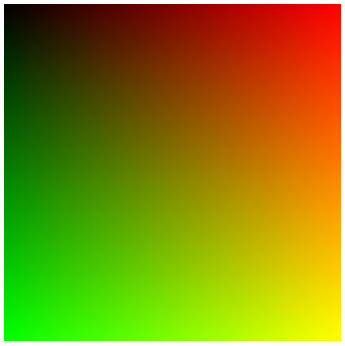
   $Size_S = n_T \times m_T$ pixels where n_T and m_T are, respectively, width and height of the target image.

b) For each pixel, set red component to x-coordinate and green component to y-coordinate of that pixel, set blue component to zero.

Pseudo code1 states the S-Image creation process.

Pseudo code1 (S_Image Creation)
Input: n_T: width of the target image.
m_T: height of the target image.
Output: S_Image of size $n_T \times m_T$
Begin
For i=1 to n_T
For j=1 to m_T
S_Image(i, j).R=i
S_Image (i, j).G=j
S_Image (i, j).B=0
Next j,i
End

Figure (2), shows the S-Image with size of 255 X 255, and for simplicity, it shows the pixel values of the first 5x5 pixel area.



	1	2	3	4	5
1	11	12	13	14	15
2	21	22	23	24	25
3	31	32	33	34	35
4	41	42	43	44	45
5	51	52	53	54	55

Figure 2: S-Image creation

This image can be used to learn any geometric filter. The new pixel location can be traced using its color value.

3.2 Training Phase

Following source image creation phase, is the creation of the filtered source image S' which corresponds to the training phase, where any geometric filter can be trained on S to create S'. Thus, we can imagine that the role of this phase is as meta-learning the geometric transformation function. If for simplicity, the target image is

5x5 pixels size, figure (3) shows the contents of S' after applying the horizontal-reflection filter on S-image as usually done in the graphic software (e.g. in Adobe Photoshop).

11	12	13	14	15
21	22	23	24	25
31	32	33	34	35
41	42	43	44	45
51	52	53	54	55

S-Image

15	14	13	12	11
25	24	23	22	21
35	34	33	32	31
45	44	43	42	41
55	54	53	52	51

S'-Image

Figure 3: S-Image and its reflection (S'-Image)

3.3 Application Phase

In this phase, both source filter image (S') and the target image (T) are used to create the filtered target image (T'). Every pixel in T' is set by the value of T at the position (x, y) denoted by the (Red, Green) color components of S'. Thus: S' is used as a lookup table. Pseudo code (2), states the application phase steps, and figure (4), shows the contents of S'-Image, T-Image (sample contents), and T'-Image.

```
Pseudo code 2: Application Phase
        Input: S'_Image, T_Image
       Output: T'_Image "The geometrically filtered target image"
Begin
            For i=1 to n_T
                For j=1 to  m_T
                        X=S'_Image (i, j).R
                        Y=S'_Image(i, j).G
                        T'_Image (i, j).R =T_Image(x, y).R
                        T'_Image (i, j).G =T_Image(x, y).G
                        T'_Image (i, j).B =T_Image(x, y).B
            Next j,i
End
```

15	14	13	12	11
25	24	23	22	21
35	34	33	32	31
45	44	43	42	41
55	54	53	52	51

S'-Image

150	151	152	153	154
155	156	157	158	159
160	161	162	163	164
165	166	167	168	169
170	171	172	173	174

T-Image (sample contents)

154	153	152	151	150
159	158	157	156	155
164	163	162	161	160
169	168	167	166	165
174	173	172	171	170

T'-Image

Figure 4: T'-Image production, application phase

The contents of T'-Image are obtained from by applying the application phase algorithm as follows:

T'_Image(i,j)=T_Image(S'_Image(i,j).R, S'_Image(i,j).G)

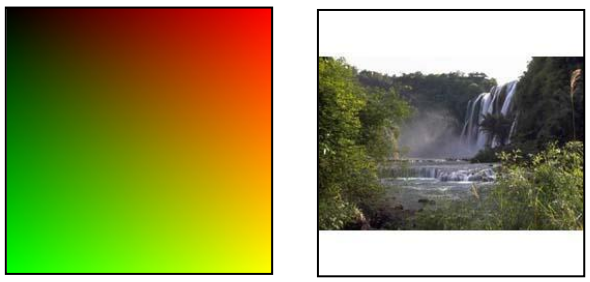
T'_Image(1,1)=T_Image(1,5)

T'_Image(1,2)=T_Image(1,4)

T'_Image(1,3)=T_Image(1,3) , and so on.

4. Results

Figures (5,6), show the results obtained by applying the learned filters on the target image, and for comparisons, the results of the same filters applied by Adobe Photoshop software are also shown.



S-Image

T-Image

S'-Image (Fish-Eye) filter

T'-Image (Fish-Eye) filter, using S'-Image

T'-Image (Fish-Eye) filter, using Photoshop

Figure (5): Results

S'-Image (Ripple) filter | T'-Image (Ripple) filter, using S'-Image | T'-Image (Ripple) filter, using Photoshop

S'-Image (Wave) filter | T'-Image (Wave) filter, using S'-Image | T'-Image (Wave) filter, using Photoshop

S'-Image (Shear) filter | T'-Image (Shear) filter, using S'-Image | T'-Image (Shear) filter, using Photoshop

S'-Image (Polar) filter | T'-Image (Polar) filter, using S'-Image | T'-Image (Polar) filter, using Photoshop

Figure (6): Results

5. Conclusions

Based on the results presented in this work, it can be concluded that the proposed algorithm presents a good and efficient solutions to the problem of image geometric properties transferring. While developing the algorithm, the following aspects are realized:

a) The size of the target image should be the same as the source image. So the target image should be preprocessed to have the same size of the source image. The image size should not exceed 255 x 255.
b) In the proposed algorithm, there are no complex computations; the time is waste only for extracting the color components.
c) Using the proposed algorithm makes all transformations take the same time to complete. (i.e. the time needed for the simple reflection transformation, is the same for the very complex fish-eye transformation) , that is because the algorithm applying filters by just make mapping between images.
d) Scaling transformation cannot be learned by the proposed algorithm, that is because, scaling affect the pixel values as well as pixel coordinates.
e) The proposed algorithm is very suitable to be used in advertising screens, this is because, it is very simple, so that, the hardware design which controls the advertising screen will be very easy.

6. Future Work

Following are some suggestions which can be considered as ideas for future work for researchers interested in this field:

a) The B-component of RGB color system, can be used to extend the size of S-Image from 255 x 255 to 4096 x 4096, by dividing the B-component between X and Y, so that each one will be 12-bit instead of 8-bit length.
b) All geometrical filters that are used here, are applied on a single image, one can study the application of these filters on a large movie frames.
c) The accuracy of results appears acceptable in comparison with those obtained from PhotoShop, but the accuracy is not computed mathematically. One can use a proper method to compute the exact error between images resulted from this algorithm and those resulted from other programs like PhotoShop.

References

[Bocheck 2000] Paul Bocheck, ***"Geometric transformations: Warping, registration, Morphing"***, Seminar on computer graphics, based on *A. K. Jain, Fundamentals of Digital Image Processing,* Polytechnic University, Brooklyn, NY11201, 2000.

[Hertzmann 2001] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, David H. Salesin, ***"Image Analogies",*** pages 327-340 of: proceeding of ACM SIGGRAPH 2001. ACM press/ACM SIGGRAPH.

[Hertzmann 2002] Aaron Hertzmann, Nuria Oliver, Brian Curless, Steven M. Seitz, ***"Curve Analogies"***, thirteen eurographics workshop on rendering, 2002

[Huttunen 2004] Jari Huttunen, ***"Image Analogies",*** Seminar on computer graphics, Helsinki University of Technology, Telecommunications Software and multimedia Laboratory, spring 2004.

[Lai 2005] Y.K.Lai, S.M.Hu, D.X.Gu, and R.R.Martin, ***"Geometric Texture Synthesis and Transfer via Geometry Images"***, Association for computing machinery, inc, 2005.

[Lindley 91] Lindley, C. A. ***"Practical Image Processing in C"***, New York: John Wiley & Sons, Inc., 1991.

[Mertens 2006] Tom Mertens, Jan Kautz, Jiawen Chen, Philippe Bekert, Fre'do Durand, ***" Texture Transfer Using Geometry Correlation",*** Eurographics Symposium on rendering, Eurographics Association, 2006.

[Reinhard 2001] Erik Reihard, Michael Ashikhmin, Bruce Gooch, Peter Shirly, ***"Color transfer Between Images"***, IEEE Computer Graphics and Applications, 21(5), pp. 34-41, Sep. 2001.

[LaValle 2006] Steven M. LaValle, ***"Planning Algorithms"*** University of Illinois, Cambridge university press, 2006.

[Waltman 2001] Jason Waltman, ***"Bluring the line, Algorithms and aesthetics of image processing"***, an honors theis presented to department of mathematics& computer science at Wittenberg University, 2001.

Bibliography

Angel, E. "*Interactive Computer Graphics: A Top-Down Approach with OpenGL"* Addison-Wesley, 2000.

Davies, A., and P. Fennessy. *"Digital Imaging for Photographers"*. Boston: Focal Press, 1998.

Foley, J., A. van Dam, S. Feiner, and J. Hughes. "*Computer Graphics: Principles and Practice",* Addison-Wesley, 1990.

Gonzalez, R. C., and P. Wintz. Digital Image Processing. Reading, MA: Addison-Wesley, 1988.

Lyon, D. A. *"Image Processing in Java"*. Upper Saddle River, NJ: Prentice Hall PTR, 1999.

Pitas, I. *"Digital Image Processing Algorithms and Applications"*. New York: John Wiley & Sons, Inc., 2000.

Teuber, J. *"Digital Image Processing"*. New York: Prentice Hall, 1993.