# Using *Mathematica* as a teaching tool: A Digital Image Processing Experience

**Han J. W. van Triest[1,2], Hao Wu[1], Bart M. ter Haar Romeny[1,3], Shouliang Qi[1,2] ,Yaonan Zhang[1,2], Yan Kang[1,2]**

[1]Sino-Dutch School of Biomedical and Information Engineering, Northeastern University, Shenyang, China
[2]Key Laboratory of Medical Image Computing, Ministry of Education, Northeastern University, Shenyang, China
[3]Eindhoven University of Technology, Eindhoven, the Netherlands
Corresponding author: han@bmie.neu.edu.cn

**Abstract**

Digital Image Processing is a course taught in a wide variety of educational programs. Due to its visual nature, the usage of modern technology such as projectors is virtually mandatory. In this paper the benefits of using Computer Algebra Software in the classroom are studied. More specifically *Wolfram Mathematica* is used as a replacement of traditional *Powerpoint* presentations, as it contains slideshow functionality, and its **Dynamic[]** functionality enables for quick creation of applets.

**Keywords**: *Mathematica*; Digital Image Processing; Teaching

## 1. Introduction

*Digital Image Processing (DIP)* is a course that is taught in many programs of amongst others Electrical Engineering, Computer Engineering and Biomedical Engineering. The course deals with the analysis and processing of digital images. As a course dealing with image data, it is a very visual subject making the usage of slide-based presentations practically mandatory. Traditionally, slideshow programs such as *Microsoft Powerpoint* are used for this purpose.

Furthermore, to reinforce student learning, the course is often taught in combination with laboratory skill classes, in which students are challenged to solve various challenging image processing problems. A large variety of programming languages have been used to teach image processing, including *Matlab* [1], *C++* [2] and *Java* [3]. A clear disadvantage of this setup is that the presentations and the programming language are separated, and code cannot be run in the slide show software and *vice versa*.

## 2. Material and Methods

*Mathematica* (Wolfram Research, Champaign, Illinois, USA), is an innovative numeric and symbolic computing environment initially developed for the study of automatons. In recent years, it has become the leading computer system in many areas of engineering, hosting an extensive library of algorithms for all types of engineering problems. *Mathematica* has been used in a host of different disciplines, such as Mathematics [4], Physics [5] and economics [6], but mostly as a tool during exercise classes. .

The software *Mathematica* consists of two parts, the *kernel* which is responsible for doing all the calculations and the *FrontEnd*, responsible for displaying the results in an understandable fashion (See Figure 1). The *FrontEnd* enables the user to write code similar to how one would write it on paper, and the extensive typesetting engine makes it suitable for writing professional mathematical documents. In fact, the book [7] is fully written in *Mathematica* and shows the code and text side-by-side.

Since *Mathematica* 6.0 was released in 2007, the **Dynamic[]** functionality was added which enables users to write complicated applets in only few lines of code, letting the user freely explore the influence of parameters on functions. In Figure 1, an example is given where the frequency of a *Sinc* function can be varied.

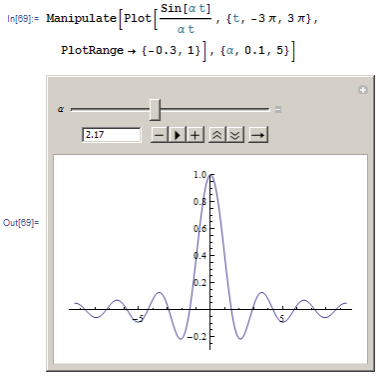



Fig. 1: A simple applet enabling the user to manipulate the frequency of the *Sinc*-function that is plotted. Also note the style of the input, where the function is inputted as a fraction and Greek letters can be used as variables.

A perhaps lesser known function of the *FrontEnd* is the possibility to make and display slide shows. In combinations with the applets created using **Dynamic[]**, a very interactive experience is created, enabling teachers to explain complicated theories and mathematical equations relatively effortless. Moreover, as the presentations are written in the same (programming) language as the exercises, no mismatch exists, and learned skills and theory can directly be applied. Thousands of useful examples with source code are available from the Demonstrations website of Wolfram [8].

## 3. Experiments and Results

To demonstrate the use of *Mathematica* in teaching the course of Digital Image Processing, two examples are given. Both examples come from a course as taught over the past four years by the first author.

### 3.1. Example 1: Affine Transforms

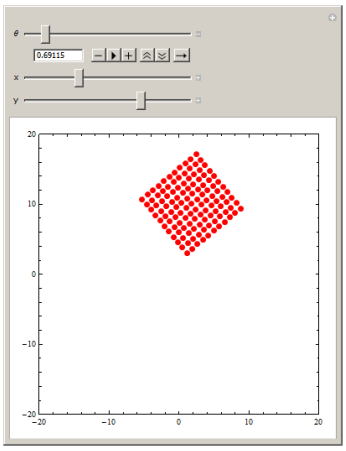
Affine transforms are transforms that can be applied to images that will preserve straight lines and ratios of distances between two points lying on a straight line. For this purpose a 2D coordinate is represented as a 3-tuple, by appending the value 1 to the end. A rotation over $\boldsymbol{\theta}$ in point $\vec{\boldsymbol{p}} = (\boldsymbol{p_x}, \boldsymbol{p_y})$, can now be calculated by simple matrix multiplication. First, the point $\vec{\boldsymbol{p}}$ is translated to the origin after which the rotation over $\boldsymbol{\theta}$ is performed, and the whole figure is moved back again. The whole process for coordinate $\vec{\boldsymbol{x}}$ can now be expressed as the concatenation of two translation and one rotation operation [1]:

$$\boldsymbol{T_{\vec{p}} \cdot R_{\theta} \cdot T_{-\vec{p}} \cdot \vec{x}} \qquad (1)$$

Where $\boldsymbol{T_{\vec{p}}} = \begin{pmatrix} \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{p_x} \\ \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{p_y} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{pmatrix}$ and $\boldsymbol{T_{\theta}} = \begin{pmatrix} \boldsymbol{\cos\theta} & \boldsymbol{\sin\theta} & \boldsymbol{0} \\ \boldsymbol{-\sin\theta} & \boldsymbol{\cos\theta} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{pmatrix}$ are the translation and rotation transform matrices respectively. In Appendix A, the code for performing the transforms are given, and in Figure 2, a small applet is shown in which the student can alter the parameters

of the transform and thus studying the effect of each.

```
In[51]:= Manipulate[
  ShowGrid[Transform[MyRotationInPTransform[θ, x, y], grid],
   PlotRange → {{-20, 20}, {-20, 20}}], {θ, 0, 2 π},
  {{x, 0}, -5, 5}, {{y, 0}, -5, 5}]
```




Out[51]=

Fig. 2: Applet for displaying the principles of affine transformations.

### 3.2. Example 2: Fourier Descriptors

The second example demonstrates the filtering of shapes using Fourier Descriptors. Fourier Descriptors can be used to describe the edge of object in an image [1]. Similar to the traditional Fourier Transform, the edge is decomposed into its constituent frequencies. As noise on the edge of an object typically can be represented as the high frequency Fourier Descriptors, smoothing the edge can be done by removing these high frequency descriptors (see Figure 3).

Fourier Descriptors are calculated by first sampling the edge of an object equidistantly, resulting into the functions $\boldsymbol{e_x(t)}$ and $\boldsymbol{e_y(t)}$, representing the $x$ and $y$ coordinates respectively. The two functions can be combined into one complex function according to $\boldsymbol{e(t) = e_x(t) + j\, e_y(t)}$. The Fourier Descriptors are now retrieved by calculating the Fourier Transform of function $\boldsymbol{e(t)}$.

```
Manipulate[Block[
  {newEdge = FourierDescriptorFilter[edge30, nk]},
  ArrayPlot[img6, Epilog → {Green, Thick,
     Line[Join[newEdge, {First[newEdge]}]], Red,
     PointSize[.015], Point[edge30]}]],
 {nk, Length[edge30], 1, -1}]
```

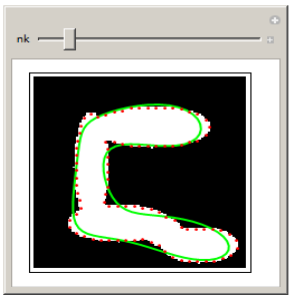



Fig. 3: Applet to display the effect of Shape Filtering based on Fourier Descriptors

## 4. Discussion

As shown in this paper, *Mathematica* can serve as an excellent teaching tool. With new features such as slideshows and **Dynamic[]**, it becomes easy to create instructive educational presentations with only little effort. When adapting *Mathematica* as a teaching tool, however, several points should be kept in mind. First of all, as the language *Mathematica* uses a functional programming paradigm, the learning curve may be a bit steep. Moreover, in some countries the software has gone somewhat unnoticed by the scientific community, leading to limited code reuse. Fortunately, code in *Mathematica* is easy to write, and as compared to other programming languages only requires few lines to do complicated actions, speeding up substantially the *design* process by the students.

## 5. Conclusion

In this paper we have demonstrated the use of modern Computer Algebra Systems in the context of teaching Digital Image Processing. The powerful **Dynamic[]** functionality of *Mathematica* enables

the user to quickly create complicated applets with only the minimum of code, thus making difficult concepts accessible for students to play with. For a large database of demonstrations, the reader is referred to [8].

## 6. Acknowledgements

This work has been partly sponsored by the Digital Image Processing Project by the Graduate Student Education program of Northeastern University, Shenyang, China.

## 7. Appendix A: Source Code

Below, the code for performing and displaying Affine Transforms is displayed. The function *GridPoints* is used to create a grid of *nx* by *ny* points:

```
GridPoints[nx_, ny_] :=
  Flatten[Table[{x, y}, {x, 1, nx},
    {y, 1, ny}], 1];
```

Grids can be displayed using *ShowGrid*:

```
ShowGrid[grid_] :=
  Graphics[{PointSize[Large], Red,
    Point[grid]}, Frame → True,
   PlotRange → {{-20, 20}, {-20, 20}}];
```

The function *Transform* applies transform *T* to grid *cl*:

```
TransformOneCoordinate[T_, c_] :=
 Block[{cc = c}, Drop[T.AppendTo[cc, 1], -1]]

Transform[T_, cl_] :=
 Map[TransformOneCoordinate[T, #] &, cl]
```

Finally, the function *MyRotationInPTransform* can rotate a grid over angle $\theta$ centered in *x* and *y*.

```
MyRotationInPTransform[θ_, x_, y_] :=
 {{1, 0, x}, {0, 1, y}, {0, 0, 1}}.{{Cos[θ], -Sin[θ], 0}, {Sin[θ], Cos[θ], 0}, {0, 0, 1}}.{{1, 0, -x}, {0, 1, -y}, {0, 0, 1}}
```

Next is the *Mathematica* code for filtering the edges of shapes using Fourier Shape Descriptors. The inputs to the function are *pnts* which contain the edge-points of the figure, and *n*, the number of points that are to be used during the reconstruction of the shape filtered image.

```
FourierDescriptorFilter[pnts_, n_: All] :=
 Block[{edge, fr, frr},
  (*Calculate descriptors*)
  edge = pnts[[All, 1]] + i pnts[[All, 2]];
  fr = Fourier[edge];
  (*Filter the descriptors*)
  frr = If[n === All, fr,
    Table[
     If[Round[Length[pnts]/2 - n/2] + 1 ≤
       t ≤ Round[Length[pnts]/2 + n/2 - 1] + 1,
      0, fr[[t]]], {t, 1, Length[fr]}]];
  (*Return the inverse*)
  With[{if = InverseFourier[frr]},
   Transpose[{Re[if], Im[if]}]]
 ]
```

## 8. References

[1] R. C. Gonzalez, R. E. Woods and S. L. Eddins, *Digital Image Processing Using Matlab*, 2nd Edition (2009), Gatesmark Publishing.

[2] M. Wang and C. H. Lai, *A Concise Introduction to Image Processing using C++* (2008), Chapman & Hall/CRC Numerical Analysis and Scientific Computing Series.

[3] W. Burger and M. J. Burger, *Digital Image Processing: An Algorithmic Introduction Using Java* (2008), Springer Press.

[4] J. Muzangwa, "Student Experiences with Visualization of Abstract Entities in the Learning of Multivariable Calculus", Journal of Arts, Science & Commerce, Vol IV:1, pp 59-66, 2013.

[5] N. Hothi and S. Bisht, "Contemporary Physics Teaching using Mathematica Software", *International Journal of Innovative Research & Development*, Vol 2:2, pp 12-20, 2013.

[6] G. Hodgkin, "Using Mathematica as a Teaching Tool in the Undergraduate Economics Curriculum", Journal for Economic Educators, Vol 3:1, 1999.

[7] B. M. ter Haar Romeny, "Front-End Vision and Multi-Scale Image Analysis" (2003), Springer Press.

[8] http://Demonstrations.wolfram.com