AC 2012-3546: TEMPLATE-BASED IMAGE PROCESSING TOOLKIT FOR ANDROID PHONES

Mrs. Santosh Chandana Golagani, University of Texas, San Antonio
Mr. Moosa Esfahanian, University of Texas, San Antonio
Dr. David Akopian, University of Texas, San Antonio

David Akopian is an Associate Professor at the University of Texas, San Antonio (UTSA). He joined the UTSA in 2003 where he founded the Software Communication and Navigation Systems Laboratory. He received the M.Sc. degree in radio-electronics from the Moscow Institute of Physics and Technology in 1987 and Ph.D. degree in electrical engineering from the Tampere University of Technology (TUT), Finland, in 1997. From 1999 to 2003, he was a Senior Engineer and Specialist with Nokia Corporation. Prior to joining Nokia in 1999, he was a member of teaching and research staff of TUT. His current research interests include digital signal processing algorithms for communication and navigation receivers, mobile applications, and learning methods. He authored more than 30 patents and 100 publications.

Dr. Can Saygin, University of Texas, San Antonio

Can (John) Saygin is an Associate Professor of mechanical engineering and a research investigator in the Center for Advanced Manufacturing and Lean Systems (CAMLS) at the University of Texas, San Antonio (UTSA). He is also the Director of the Interactive Technology Experience Center (iTEC) and the Director of the Manufacturing Systems and Automation (MSA) Laboratory. He received his B.S. (1989), M.S. (1992), and Ph.D. (1997) degrees in mechanical engineering with emphasis on manufacturing engineering from the Middle East Technical University, Ankara in Turkey. In his academic career, he worked at the University of Toledo (1997-1999) and the Missouri University of Science and Technology (formerly University of Missouri, Rolla) (1999-2006) before joining UTSA in Aug. 2006. For more, please visit http://engineering.utsa.edu/˜saygin/.

Template–Based Image Processing Toolkit for Android Phones

Abstract

Nowadays smart phones are becoming the most widely used portable devices. Cell phones significantly enrich daily human life experience by providing ubiquitous access to sophisticated multimedia features such as voice and data services, short messaging, built-in cameras, Wi-Fi, Internet, Bluetooth and many more. Attracted by "any time and any place" communication accessibility many educators explore opportunities to improve student learning by using either standard phone features or applications dedicated for learning needs. In these scenarios cell phone is considered as learning content delivery and interaction device. While many such studies are reported, another aspect hasn't been explored thoroughly – efficient learning strategies for smart phone application development itself. This is an important problem for electrical and computer engineering students as there exist credit hour constraints in curricula, technology changes very fast, and offering related chain of courses is not very feasible. This paper presents an approach facilitating the introduction to smart phone application development using short template projects combined in a template library. The idea is to familiarize students with the whole development cycle by minimizing code-programming, by having ready-made templates which can be manipulated for hands-on experience. The complexity of course materials can be varied using different set of modules, and thus such learning modules can be offered independently, integrated in other conventional courses or used for outreach. With this concept the phone application development is demystified, the big picture clarifies, while students can learn specific insights according to their needs using existing learning tracks. The described system is created for Android phones but the idea can apply for other platforms as well. The proposed library offers a convenient user interface to facilitate students to analyze the image processing techniques by making certain modification in the pre-existing code.

1. Introduction

Modern smart phones transformed to sophisticated personal assistant devices being equipped with high resolution cameras, internet, text messaging, Wi-Fi and Bluetooth connectivity, etc. Documents and images can be displayed in different formats and various applications are either deployed or can be downloaded to enhance student ability to improve their learning.

To certain degree cell phones closed digital gaps as students of different backgrounds and in different countries own mobile phones. It not surprising, that educators have demonstrated high interest in transforming cellular phones to a teaching tool. While initially the cell phone was considered a distractive device[1].

For example, an early mobile app in smart phones, 'myHomework'[2] allows students to track their homeworks, classes and assignments. Thornton and Houser[3] used phones to email English vocabulary words to mobile phones of Japanese University students. They reported better learning using phones compared to regular study materials on paper: 71% of students have shown interest in learning English vocabulary on mobile platforms in spite of personal computers and 93% felt that mobile based education is effective method for teaching. According to a survey[4] conducted by the Learning and Skills Development Agency in a project called "m-learning", many young adults aged 16 to 24 years who are not involved in schooling showed

great interest in using mobile phone games for learning English (49%) and Mathematics skills (44%). The research facilitated access to learning materials and also investigated different learning approaches using mobile phones with individuals and groups. Also, Williams[5] suggested that students make use of their cell phones for timing analysis of experiments, communicating through text messaging for surveys, finding definitions to new terms and recording lab results using camera.

While cell phones are already used for different educational purposes, this paper illustrates a different perspective, i.e. teaching application development itself, rather using ready applications. This is necessitated by the fact, that many engineering systems integrate cell phone interfaces into their systems to improve usability, and the job market for mobile phone developers is very hot. Conventional teaching curricula may not quickly adjust to fast changes in industry as typically a chain of courses would be needed, credit hour limitations of programs may constrain integrating additional courses, development environments continuously change, or students may benefit from early exposure to develop various projects.

Our approach is the following. Create a template library of image processing algorithms and an easy to use user interface to work with these algorithms. Incorporate this library as a toolkit for mobile application development and training. Then the toolkit can be used both as a template to learn app development and as a kit for promoting STEM (Science, Technology, Engineering and Mathematics). I.e., engineering students can manipulate the templates for the possibility to see the whole development cycle minimizing specific coding efforts. At the same time the applications related to engineering can be used for outreach to tell prospective students about electrical and computer engineering programs. Applications may include e.g. image processing algorithms and control systems using cell phones for remote control through wireless links. Such short crash course modules can be incorporated in existing conventional courses such as Communication Systems, Wireless Communications etc., or offered as an independent course. Even courses, such as Digital Signal and Image Processing can use sample projects for hands-on experience.

This all is possible because smart phones are created to accommodate external applications. Particularly Android[6] became popular recently as an open source platform for Java-based[7] projects. It is a convenient application development and integration environment.

The described toolkit used in this paper is simple and convenient to learn. The user friendly Graphical User Interface (GUI) is developed to facilitate analysis of image processing techniques by making certain modification in the pre-existing code of digital image processing concepts than developing from scratch. Moreover, assuming that users may have limited knowledge on Java we propose a "learn by example" approach[8] such that students with various backgrounds can explore applications of various degree of complexity. Simple "edit–compile–execute" programming[8] cycle is used for this paper to minimize code development cycle. And finally, the modules are offered in a classroom and survey is conducted to reveal the impact of the proposed strategy.

Figure 1 shows the toolkit running on Android 1.5 emulator. Likewise, it can be deployed in any Android phone independent of SDK version. The tutorial instructions include (1) SDK

installations, (2) template-toolkit installation, (3) compiling and application installations on phones, (4) sample examples-labs of code manipulations to implement projects.

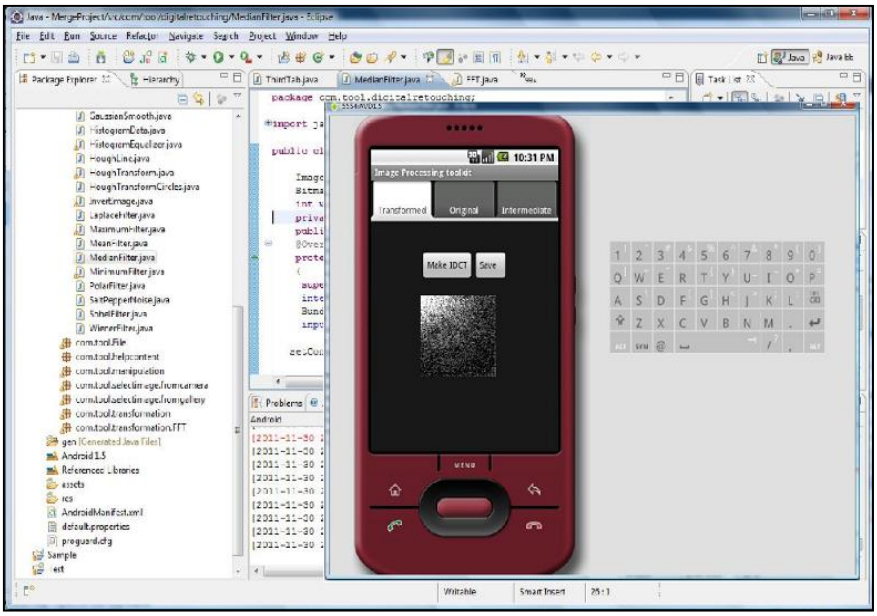


Figure 1. Application deployed on Android 1.5 emulator

2. Overview of image processing toolkit

The image processing algorithms included in the toolkit are classified as: Manipulations, Transformations and Digital Retouching. The first category 'Manipulations' consist of straight forward mathematical arrangement of pixel elements for example, add/subtract, scale, crop, clone, transpose, skew, and rotate an image.

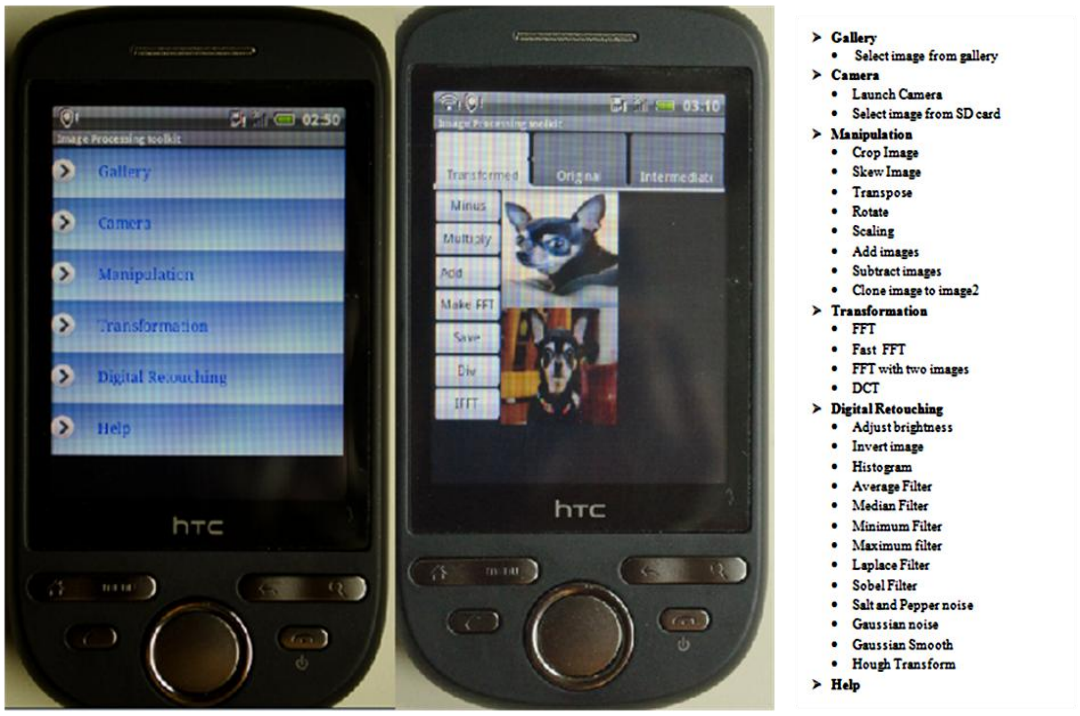


Figure 2. Menu structure display on Android phone(left hand side). On right side is the full 'Menu' structure implemented in this toolkit

Second category, ‘Transformations’ include algorithms which convert images from spatial domain to transformed domain and also inverse functions to convert back to spatial domain such as. Examples are well known transformations such as Discrete Cosine Transform (DCT) and Fast Fourier Transform (FFT). Finally, ‘Digital Retouching’ includes a set of image evaluation algorithms for instance, histogram, noise removal and edge detection filters. Figure 2 shows the menu structure implemented in this toolkit and deployed in an Android HTC phones. Furthermore,

3. Template Examples

3.1 Demonstration 1- FFT

This template illustrates how to process FFT. Open source FFT Java codes are widely available for usage, e.g. the package in[9]. The readers refer to[9] for details on original implementation. FFT is a fast implementation of Discrete Fourier Transform (DFT) which is used for transforming signals and images to frequency domain. Many algorithms exploit frequency domain for processing signals and images. In this demonstration image sizes are limited to powers of two. As input image is a two dimensional matrix, the Separable two dimensional transform is performed by applying one-dimensional FFTs to image matrix rows and columns. Since the output of FFT consists of real and imaginary components, only magnitudes of transformed image are used for displaying FFT domain results. The Inverse Fast Fourier Transform (IFFT) is then applied to reconstruct the original image from the transformed image.

Studying shapes in space and frequency domains:
In practice, assignment examples are assigned to the students to explore on how various geometric shapes are represented in frequency domain. Figure 3 shows the results of this assignment. The procedural steps are the following.

1. Perform FFT on shapes like rectangle, square, triangle etc.
2. Also perform FFT on geometrical shapes like circle, sphere, cylinder etc.
3. Observe vertical and horizontal low frequency components concentrated at the center of square image in frequency domain. One can also observe that FFT of circle image has low frequency symmetric circular components concentrated at the center. On the other hand, for triangular image, its FFT has lines perpendicular to the sides of triangle.

In other case, consider different alphanumeric characters to study shapes in frequency domain. For numerical value 8, its frequency transformation has symmetric property resembling the shape of ‘eight’. For alphabets X and N, we can observe that the transformed images consist of bright lines perpendicular to the lines shown in these alphabets. Furthermore, for special character ‘#’ its transformed image also has multiple lines perpendicular to lines in spatial domain.

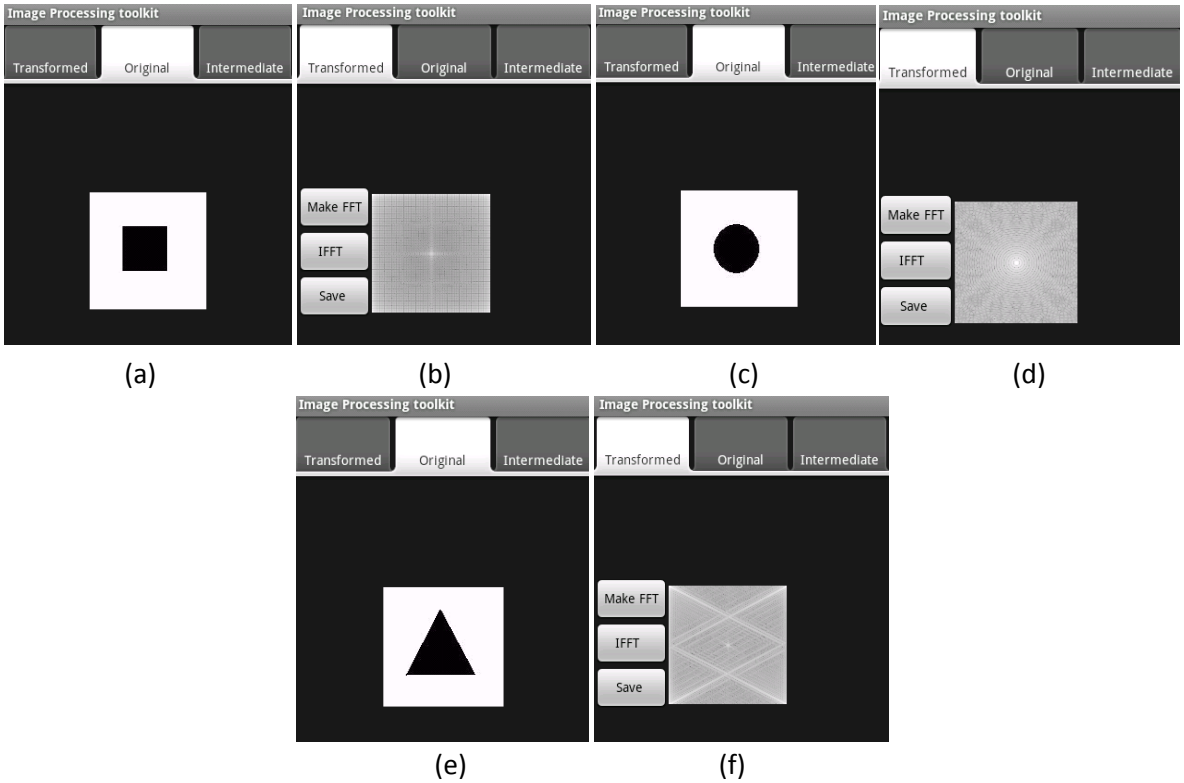


Figure 3. (a) Rectangle image; (b) Rectangle in frequency domain; (c) Circle image; (d) Circle in frequency domain; (e) Triangle image; (f) transformed to frequency domain

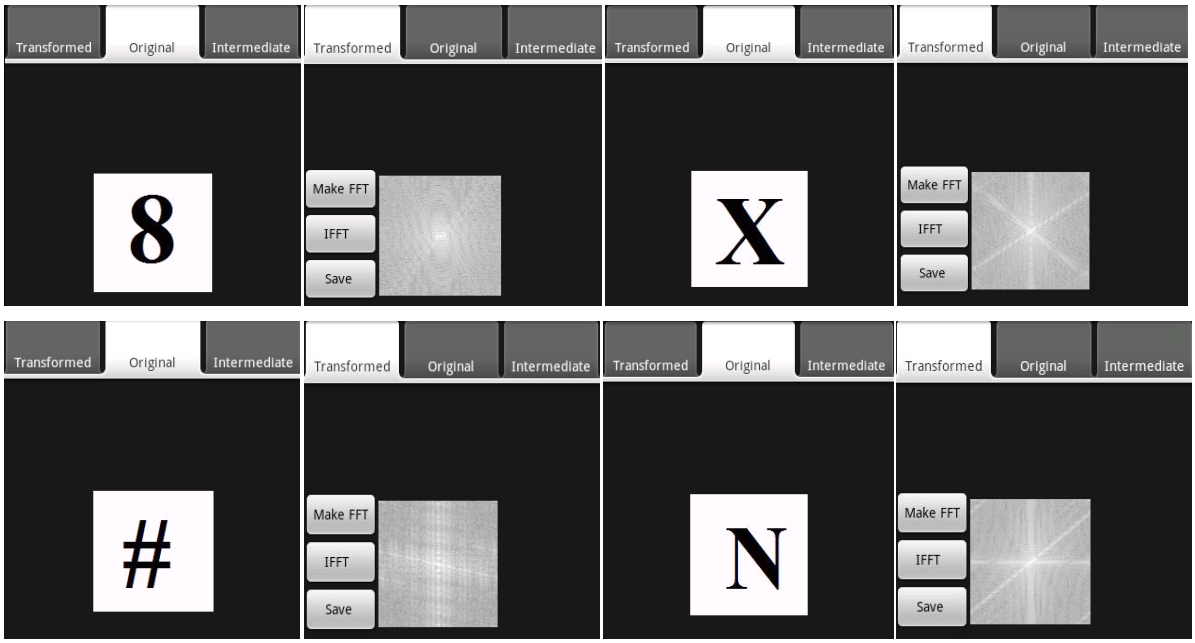


Figure 4. Alphanumeric images in space domain and their respective FFT images

Frequency-domain processing of two images

Image manipulation can be performed in frequency domain. Images can be added, subtracted, multiplied etc. in frequency domain for various needs, e.g. filtering. Then converted to spatial domain. The following are the procedural steps to apply FFT on two images:

1. Select two images of same size and size of power of two.
2. Apply FFT on two images.
3. Manipulate transformed (FFT) images by applying add / minus/ multiply / division.
4. Then apply IFFT to reconstruct the original image.

Figure 5 illustrates convolution through multiplication in frequency domain. The following are the steps to apply convolution: (1) Open two images; (2) Apply FFT on each image; (3) multiply images in frequency domain and (4) transform the product to spatial domain using Inverse FFT (IFFT).

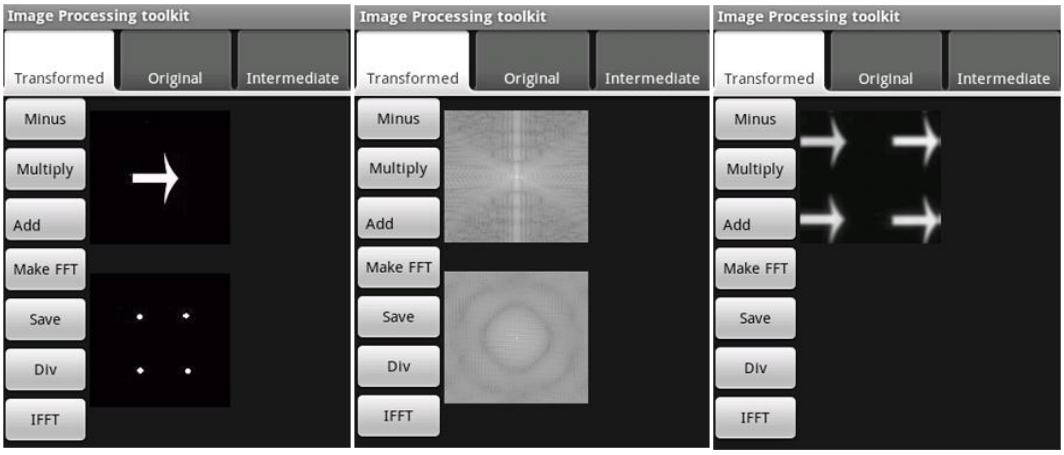


Figure 5. (a) Original two images; (b) FFT applied; (c) convolution applied and its IFFT

The linear property of FFT is explained by adding two images in frequency domain and transforming them into space domain. The resultant image in space domain is the addition of two original images. This property is explained as shown in Figure 6.

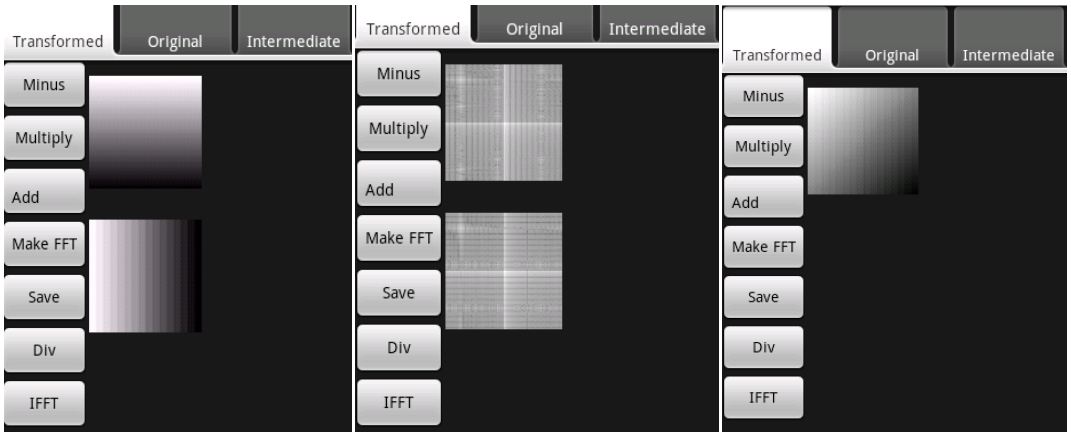


Figure 6 (a) Original two images; (b) FFT applied; (c) Add image FFTs and apply IFFT.

3.2 Demonstration 2- DCT

Discrete Cosine Transform (DCT) is used in image and video compression standards. It is closely related to Discrete Fourier Transform (DFT) but consists of only real valued transformed coefficients. It tends to accumulate higher energy at low frequency components in the upper left corner of the transformed image in frequency domain. Subsequently, quantization and lossless coding are applied to transformed coefficients to obtained compressed representation of images. A tutorial and an implementation can be found e.g. in [10]. This template uses separable two-dimensional DCT approach where-in one-dimensional DCTs are applied on row and column vectors. The toolkit also contains Inverse Discrete Cosine Transform (IDCT) to retrieve the original image from the transformed image. In this template the compression properties of DCT are studied.

E.g. the following algorithm illustrates a general compression concept: 1) Convert color image to grayscale image; 2) Read image in 8x8 blocks; 3) Apply forwardDCT ( ) to perform DCT .Store this result in temporary matrix; 4) Apply quantization; Then reconstruct the image in the following sequence 5) Apply de-quantization; 6) Run Inverse DCT; 7) Apply Normalization.

Two quantization techniques are included in this toolkit: 1) Quantization by dividing DCT matrix by a factor and round off to nearest integer. This logic is implemented and is ready to use as shown in Table 1; 2) Quantization retains the low frequency coefficients and nullifies to zero high frequency ones (zonal thresholding).

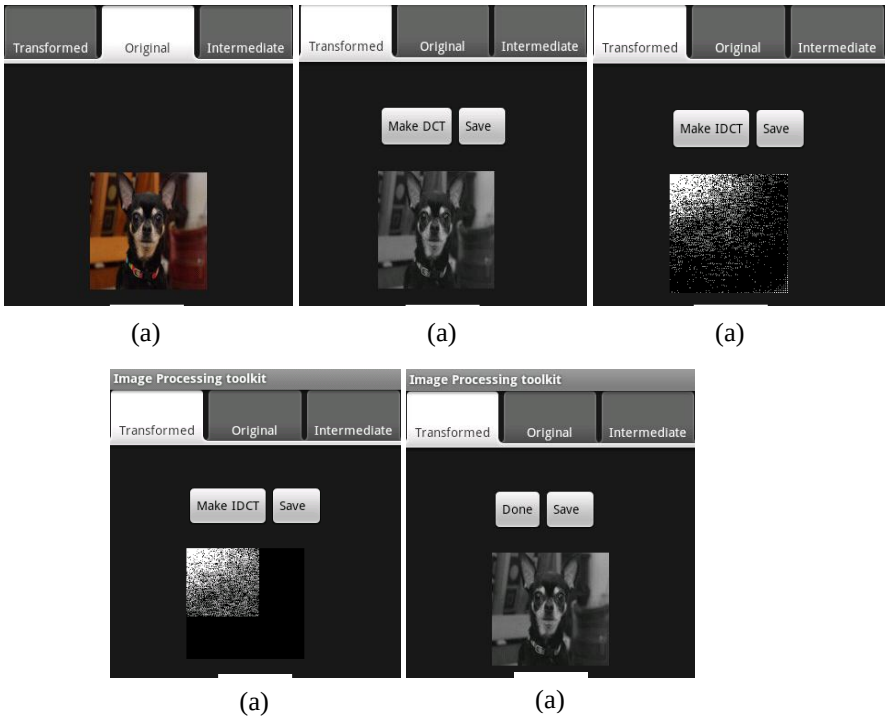


Figure 7 a) Original image; b) grayscale image; c) Quantization using scaling d) Quantization using zonal zeroing; (r=0); e) reconstructed image.

Table 1. Code changes for DCT algorithm.

	Changes	Code
1)	To implement quantization by scaling.	dct_data = dct2.getQuantization_Scaling(dct_data,2) *// dct_data=dct2.getQuantization_Zeroing(dct_data,100)*
2)	To implement quantization by zeroing.	dct_data=dct2.getQuantization_Zeroing(dct_data,100) // d*ct_data=dct2.getQuantization_Scaling(dct_data,2)*
3)	Change the block size from 100 to 40.	dct_data=dct2.getQuantization_Zeroing(dct_data,40)

3.3 Demonstration 3- Histogram

Histogram is a graphical representation of pixel intensity value frequencies. This template illustrates the technique to analyze histogram of 8 bit grayscale images representing 256 intensity levels. If a color image is selected, convert it to 8 bit grayscale image as shown in Figure 8. The histogram consisting of low grayscale range indicates darker image while that of high grayscale range represents brighter image. On the other hand, if the histogram is a narrow graph, it signifies low contrast image whereas wide graph indicates high contrast image. Manipulations to image brightness change histogram in reasonably predictable way. In addition, histogram equalizer can be used to distribute the image intensity uniformly and improves the brightness of image yielding a flat graph.

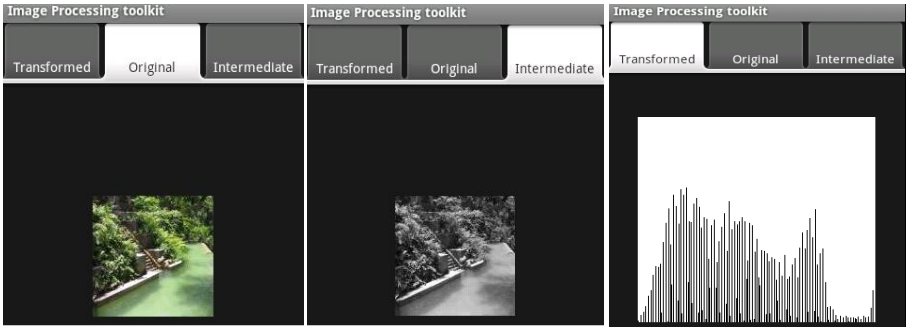
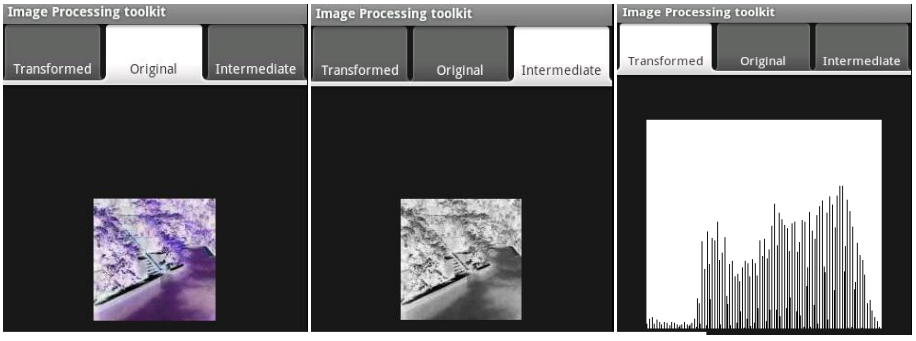


Figure 8 a) Original image; b) Grayscale image; c) Histogram



d) Negative image; e) Grayscale image f) Histogram

The code fragment of histogram is shown in Table 2. In this code, the input image data is read by using "*image.getData( )*" and stored in byte array. Since the signed Byte ranges from -128 to 128, the histogram array '*result*' is offset such that element '0' corresponds to signed byte value Byte.MIN_VALUE. In Figure 8, the histogram of original image and its negative version are shown. We can observe from the graph that histogram of negative image is the inverse graph compare to that of the original image.

3.4 Demonstration 4- Hough Transform

Hough transform is used to identify lines, circles or ellipses in images and is quite insensitive to image noise. The details and implementation samples can be found in e.g.[11]. The important feature of Hough transform is identify straight lines of shape $y = mx + b$. However, lines in x ,y coordinates give unbounded values ,therefore, we consider straight line in polar coordinates as $x\cos\theta + y\sin\theta = r$ where r is the length of the normal from the origin and θ is the orientation of r w.r.t X – axis.

Table 2. Code fragment –Histogram

```
public static int[]
computeHistogram(Gray8Image image){
   int[] result = new int[256];// histogram
array of size 256
    …
  byte[] data = image.getData();
  for (int i=0; i<image.getHeight(); i++) {
     for (int j=0; j<image.getWidth(); j++)
{ // byte offset to get data in range of 0-
256
   result [data[i*image.getWidth()+j]-
Byte.MIN_VALUE]++;
   } }
  return result;
   }
```

Table 3. Algorithm of Hough transform

1) Read the image pixels in an array
2) Initialize Accumulator Array "houghArray" to all zeros
3) For each edge point (x, y) in the image
For θ = 0 to 180
r= xcosθ+ysinθ
H(θ,r)= H(θ,r)+1
End
End
4) Find the values of (θ, r) where H(θ,r) is a local maximum
5) The detected line in the given image is r = xcos θ +y sinθ

The Hough transform is implemented by initially transforming each value in (x, y) plane to polar coordinates. If we plot curves represented for x, y points on one line will cross at a single point. These points represent the common values of r and theta and stored in a two dimensional array called '*accumulator'* wherein for each value of θ the value of r is calculated and consequently the value in accumulator array at (θ, r) is incremented by one. Thus Accumulator array is filled with values and peak values are found. Peak values represent more x ,y points have crossed that curve . Peaks are identified using threshold

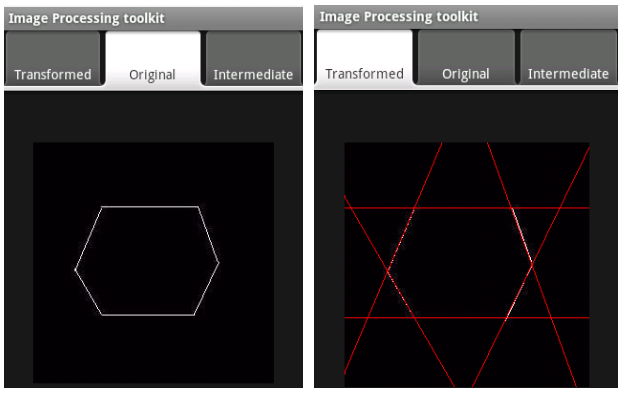


Figure 9. a) Original image; b) Hough transform.

technique and one line is identified per peak. Figure 9 demonstrates detecting straight lines for a given image using Hough transform. The algorithm steps are illustrated in Table 3.

3.5 Demonstration 5- Gaussian and Mean Filters

This template students illustrates filtering techniques applied on noisy images. Gaussian smoothing and Mean are linear filters useful to remove Gaussian noise by smoothing the noisy image. Gaussian smoothing is similar to mean filter but the kernel representation is different from mean filter. The kernel represents the shape of the Gaussian bell-shaped. The following two dimensional convolution kernel approximation is used in this template for smoothing a noisy image which is defined as 1/16 * [1, 4, 6, 4, 1] considering sigma as 1. Mean filter replaces center value of the kernel with average of all the pixel values in a 3x3 kernel. For example, if the kernel is defined as [6, 1, 2, 3, 93, 4, 19, 5, 11] the average for the window is 16, so the modified kernel becomes [6, 1, 2, 3, 16, 4, 19, 5, 11]. The smoothing property depends on the kernel size, it increases with increase in kernel size but the sharpness of image is reduced.

Figure 10. a) Original image; b) Gaussian noise added; c) Resultant image of Gaussian filter; d) resultant image of "Mean" filter.

Figure 10 shows the processed images for mean and Gaussian filters. The following are the procedural steps to apply filters: 1) Open a noise-free image; 2) Add Gaussian noise; 3) Apply 'Mean' or Gaussian filter 4) Examine the filtered images. Table 4 shows simple code changes wherein users can change kernel size by changing the value of the variable 'kernelsize'.

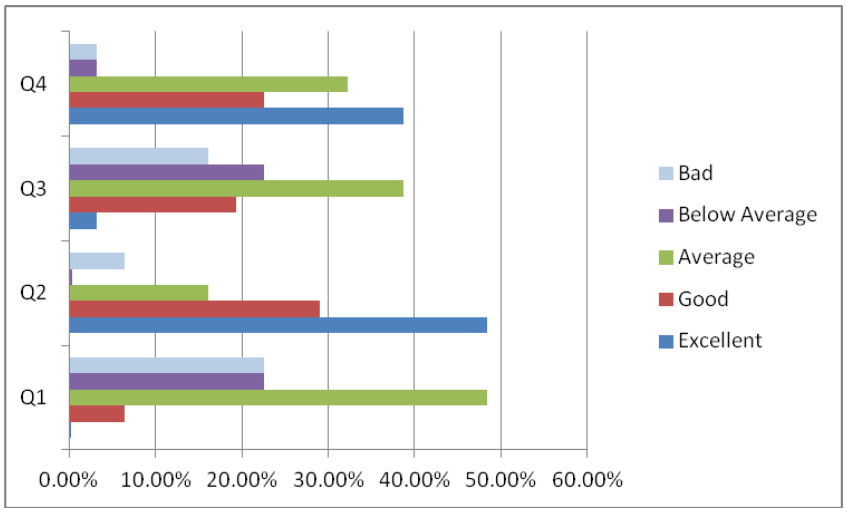
Table 4. Code change examples for "Mean" and "Gaussian" filters

	Changes	**Code**
1)	*MeanFilter.java:* Change kernel size to 4by4. This is done by changing the value of variables '*kernelsize'* and '*total_kernelelements*'.	int kernelsize = 4 ; int total_kernelelements= 16;
2)	*MeanFilter.java:* Sort the kernel in descending order.	//*Arrays.sort(kernel);* sortDescendingOrder (kernel);
3)	*GaussianSmooth.java*: Adjust kernel size to 7by7.	int kernelsize =7, sumOfKernelValues=32; int[] kernel = {1,3,7,10,7,3,1};
4)	*GaussianSmooth.java*: Change kernel to different Gaussian distribution by changing the kernel elements.	int kernelsize =7, sumOfKernelValues=64; int[] kernel = {1,6,15,20,15,6,1};

4. Learning module offered in class and survey results

A learning module based on the components of the proposed template image processing toolkit is offered at the University of Texas at San Antonio, Electrical and Computer Engineering Department, Wireless Communications class exploring application layer. Module duration: 4-weaks, 2:30h/week, 8 days, 1:15h/day. About thirty-one students have participated in this 8 days lab session and survey data was collected. Pre-screening data was collected before beginning of the session to clarify if the students have prior knowledge in programming, and how do they perceive mobile application development. Students have learned from setting up the Android development environment and they have executed successfully the given templates by using lab instruction materials within the lab duration. Consequently, they deployed the application in real phone and tested the results. Finally, a mini-project was assigned to students followed by a questionnaire to obtain students view on learning development process and challenges in coding.

The pre-screening survey data is represented in Chart 1 along with the set of asked questions. The responses of this survey are scaled[12] down from '5-Excellent', '4-good', '3-average', '2-below average' and '1-bad'. Also, the mean and standard deviation values for the pre-survey results are shown in Table 5. Most of the students do not have any programming experience. For example, in response to the first question "How do you grade your programming skills", only 6% of the students (mean=2.4) answered positively. 94% of students do not know Java programming; however they were able to execute these designed templates. 77% of students (mean =2.71) have perceived that mobile application development is difficult as shown in Chart 1 for Question 3. But, majority of the students (mean =4.12) have shown interest in learning mobile application development.



Q1) How do you grade your programming skills?

Q2) How do you grade your interest in Mobile Application Development?

Q3) At this point, how do you perceive the difficulty of mobile phone programming?

Q4) Do you think that the mobile application development is a rewarding career but you have doubts on possible difficulties when pursuing it?

Chart 1: General questions before Lab session

The feedback data collected after the lab session is plotted in Chart 2. The students' responses are scaled down from '5-excellent' to '1-bad'. Also, the mean and standard deviation results are shown in Table 6. Majority of the students (Mean =3.93) now changed their perception positively on the difficulty of the mobile application development as shown in Question 1 in Chart 2. Additionally, most of the students (Mean=4.41) found that the mobile application development workshop has clarified the general phone application development and integration

process. As a result, 83% of students opted above average (mean =3.89) that they would able to develop any Android applications. Many of the students (mean=4.34) are satisfied with this workshop and gave a positive feedback. 86% of students felt convenient to make code changes even without prior knowledge in Java language. Moreover, most of the students (mean=4.45) are satisfied with the tutorials and materials offered during Lab session are helpful.

The overall opinion of the students on this Lab session is gathered from second set of questions as shown in Table 7. This feedback was designed in terms of several aspects such as subject importance, level of difficulty, usefulness, workshop work load ,etc .Students graded the mobile application development as excellent (Mean=4.34), interesting (Mean=4.48), easy (Mean=3.72), useful (Mean=4.41), valuable (Mean=4.31), motivational (Mean=3.34), and effortless (Mean=3.27). Majority of students suggested that this session motivated them to learn mobile application development.

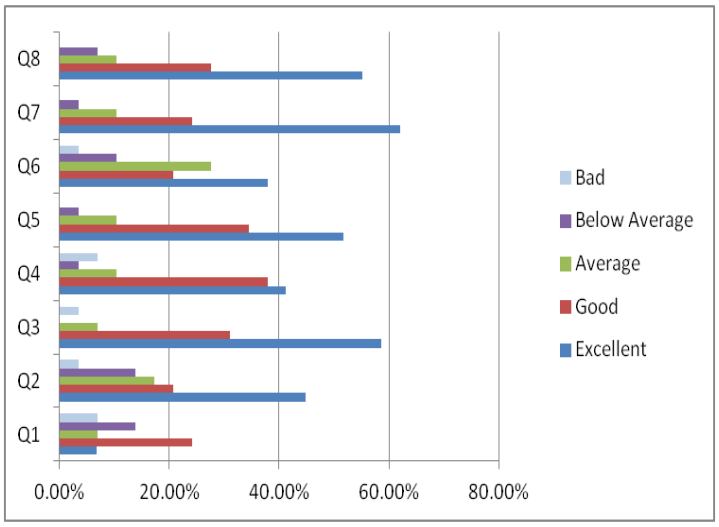


Chart 2. Application specific questionnaire after Lab

Q1) Has this workshop changed your perception on the difficulty of mobile phone programming?

Q2) Has this workshop changed your perception on the difficulty of mobile phone programming?

Q3) Do you think that you would be able to develop Android applications if needed?

Q4) Has this workshop clarified the phone application development and integration process?

Q5) Might this workshop influence your opinion on a possible use of Android Application in your senior design project?
Are your expectations from the workshop met?

Q6) Modifying known applications might help to overcome programming skill limitations for beginners?

Q7) Were Android Application Development tutorials helpful?

Q8) Were PowerPoint presentations helpful?

Table 5. Students pre-workshop feedbacks

	Questions	**Mean**	**SD**
Q1	How do you grade your programming skills?	2.40	0.90
Q2	How do you grade your interest in Mobile Application Development?	4.12	1.09
Q3	At this point, how do you perceive the difficulty of mobile phone programming?	2.71	1.05
Q4	Do you think that the mobile application development is a rewarding career but you have doubts on possible difficulties when pursuing it?	3.90	1.06

Table6. Students' post-workshop feedbacks

	Questions	Mean	SD
Q1	Has this workshop changed your perception on the difficulty of mobile phone programming?	3.93	1.31
Q2	Do you think that you would be able to develop Android applications if needed?	3.89	1.21
Q3	Has this workshop clarified the phone application development and integration process?	4.41	0.89
Q4	Might this workshop influence your opinion on a possible use of Android Application in your senior design project?	4.03	1.13
Q5	Are your expectations from the workshop met?	4.34	0.79
Q6	Modifying known applications similar to the project (image processing toolkit) might help to overcome programming skill limitations for beginners?	3.79	1.15
Q7	Were Android Application Development tutorials helpful?	4.45	0.81
Q8	Were PowerPoint presentations helpful?	4.31	0.91

Table 7. Students' general feedback about the workshop

How do you grade Mobile Application Development workshop in general?					Mean	SD
Excellent	5	-	1	Bad	4.34	0.92
Interesting	5	-	1	Boring	4.48	0.89
Easy	5	-	1	Hard	3.72	0.91
Useful	5	-	1	Useless	4.41	0.85
Valuable	5	-	1	Worthless	4.31	0.91
Motivational	5	-	1	Dry	3.34	1.15
Effortless	5	-	1	Labor-intensive	3.27	0.83

5. Conclusion

Fast learning modules may help to motivate students to learn mobile application development and prepare workforce in response to current industry needs. We proposed a learning strategy of using template-solutions for students to navigate through a challenging learning area. An image

processing toolkit implemented in Java and integrating various templates is used as such a training testbed. The development environment is the popular Android platform. The toolkit can be used by diverse students from novices to advanced users. The beginners can simply play with images and check various processing scenarios given in menu selections whereas advanced users can modify the Java code and create their own algorithms. The modules have been offered to undergraduate electrical engineering students during regular class hours, as a part application layer coverage of a senior level Wireless Communication course. A survey has been conducted to evaluate the impact. The results clearly demonstrated that within 8 days, and total of 10h, we were able to change student attitudes and positively shape student interest in pursuing careers in mobile application development.

ACKNOWLEDGEMENTS

This work was partly supported by iTEC-UTSA Learning Center and NSF grants 0942852 and 0932339.

REFERENCES

[1] C. David Caverly, R. Anne Ward, J. Michael Caverly "Techtalk: Mobile Learning and Access", Journal of Developmental Education, vol. 33, Issue 1, pp. 38-39,2p, 2009.

[2] myHomework https://myhomeworkapp.com/ (Accessed on 2011)

[3] Patricia Thornton , Chris Houser, "Using mobile phones in English education in Japan", Journal of Computer Assisted Learning,vol.21 , Issue 3, pp. 217 - 228, 2005.

[4] "Mobile phones switch young people on to learning", Education and Training , London, vol. 45, Issue 4/5, pp 288-230, 2003.

[5] M. F. Williams, "Cell Phones as Teaching Tools", Journal of Educational Leadership, vol. 68, No 2, pp. 85-86, 2010.

[6] Android SDK , http://developer.android.com/index.html (Accessed on 2011).

[7] JAVA tutorial , http://docs.oracle.com/javase/tutorial (Accessed on 2011).

[8] D. Sage and M. Unser, "Teaching image-processing programming in Java," Signal Processing Magazine, IEEE, vol. 20, No. 6, pp. 43-52, Nov.2003.

[9] Intel. Implementation of Fast Fourier Transform for Image Processing in DirectX 10. http://software.intel.com/en-us/articles/implementation-of-fast-fourier-transform-for-image-processing-in-directx-10/ (Accessed on 2011).

[10] S. Manley. DCT – A Java implementation of the Discrete Cosine Transform implementation. http://www.nyx.net/~smanley/dct/DCT.html (Accessed on 2011)

[11] R. Fisher, S. Perkins, A. Walker and E. Wolfart. Hough transform, http://vase.essex.ac.uk/software/HoughTransform/index.html (Accessed on 2011)

[12] D. Lang, C. Mengelkamp, R.S. Jäger, D. Geoffroy, M. Billaud, and T. Zimmer, "Pedagogical Evaluation of Remote Laboratories in eMerge Project", European Journal of Engineering Education, Vol. 32, No. 1, pp. 57-72, 2007.