[[1]](#footnote-1)

Speedy Tilt Shift Acceleration

Sarvesh Patil, Shubham Nandanwankar, Achal Shah  
Arizona State University (2018)

**Abstract—Fancy image transformations -- colloquially, “filters” -- have dramatically captured the attention of the mobile**

**market, to the point that operating systems and devices take special measures to promote efficient image. In this project we have built “Tilt-shift Blur” functionality simulating macro photography for full-size scenes. The tilt-shift effect simulates the focal blur effects created when photographing miniature scenes. This is achieved using Gaussian blur algorithm that uses kernel of different sigma value (blurriness) depending on the assumed physical depth of the subject. Here we have used y-coordinate to assume the depth of the image contents.** **We have designed and benchmarked our implementation using three popular platforms namely Java, C++ and ARM NEON intrinsics. The application allows user to choose any image on user’s device or drive storage and one of three platforms & displays the tilt-shift filtered image.**

*Index Terms*— ARM Neon Intrinsic, C++, Filters, Gaussian Blur kernel, Image processing, Java, NVIDIA Shield tablet

# INTRODUCTION

I

MAGE transformations or filters are one of the significant aspect of ongoing mobile computing technology. In this project we have attempted to implement one of the many popular filters known as “Tilt-shift blur filter” which simulates the focal blur effect when photographing miniature scenes. This filter effect is achieved through technique known as Gaussian blur algorithm which is built on kernel that uses varying sigma values (blurriness). For this implementation we have chosen three major platforms namely Java, C++ & ARM Neon intrinsic. The goal of implementing the algorithm with three different platforms is to take advantage language specific features of each implementation and based on that evaluate the runtime performance of our filter implementation for benchmarking. All the development & testing is done with NVIDIA Shield tablet.

# Design & Implementation

The application is designed to run on user selected image of any size, the regions of blurring, blurring intensity & the language platform with which user wishes to run the filter.

The User interface consists of three components to facilitate this user input process. The image picker icon provides user the option of either choosing an image from Gallery i.e. local storage of device or from linked drive by opening the image picker activity on selection. This selection menu also offers few sample images as well to user if user wishes to run app for these images. The selected pre-blur image is then made available on screen of device.

The scroll bar selections allows user to set the regions of blur along y-axis & intensity. Depending on selected formation of first four seekbars (a0, a1, a2, a3), the image regions are set for the location and gradient of transition between blurry and non-blurry regions. Two more seekbars are used to set the sigma\_near (σ1) and sigma\_far (σ2) values, representing the blur of near and far objects.

The spinner adapter is used to enable drop-down selection menu for user which allows one of Java, C++ or Neon I(ntrinsic) to be selected as running platform during that execution.

The “Blur it” button when pressed, on receiving all valid setup inputs, initiates the blurring operation on selected image using the blur region & implementation information. The pop up toast messages are used to update user about the status of execution during run. On successful execution, app replaces the original image with filtered image on screen in image view area. User is also intimated of timing it took for application to complete blurring operation for that particular instance of implementation on that image. A “Reset” button lets user to revert back to original image.

All these UI components communicate with underlying libraries that handles these inputs & implements Gaussian blur algorithm for given constraints. For this we calculate a Gaussian kernel window based on near & far sigma values. Then we iteratively apply the two level convolution operation on each pixel of the image with this kernel iteratively. In case of Java & C++ approaches we have used the naïve brute force method of nested loops to iterate over the image for convolution. Although in case of Neon Intrinsic we have taken advantage of vectorization method calls exposed by ARM Neon library to vectorize our approach. All three design & implementation approaches have been discussed in detail in next section. The corresponding speed ups we observed because of our approaches have been documented under Results section in later sections of this document.

## User Interface

All the UI components design is done in *activity\_main.xml* file. We have used RelativeLayout approach for designing our UI. ImageView component is used which loads the user selected image on screen.

The seekbars are used to get fine control when selecting sections image of to blur. Based on seekbar position it returns value between 0 to 255 (0 for bottom 255 when at top). These position values are then used by musicfx class to evaluate the length of section of image along y-axis that should undergo blurring effect of desired intensity.

## Main Activity

This is the main launcher activity that is presented to user on opening the tilt\_shift\_blur application. MainActivity.java file handles the all main activity initializations & operations.

The onCreate() callback sets the layout UI & calls setupUI() method defined in same file which handles all initial setup of UI components. This links component’s ids with corresponding objects that are then used throughout execution. setupUI() also defines onClick() callback behavior set on image select button to handle the user’s image selection option. If user has chosen to select an image from local gallery storage or drive, it calls a separate method showFileChooser() to handle the request. Otherwise it sets the selected pre-loaded image in app(in res/drawable directory) to bitmap object & makes it visible on screen. onActivityResult() callback handles the displaying operations for selected input stream.

The showFileChooser() method uses the intent object on storage media to open image picker activity on top of main activity. startActivityForResult() method call handles all the subsequent operations using this intent.

Six seekbar objects, four for a0,a1,a2,a3 & two for sigma values, are created and used on the callback method setOnSeekBarChangeListener() in order to track & evaluate the seekbar position from UI input stream. Also setOnClickListener() listener gets called on two button objects, one for “Blur it” & another for “Reset”. On clicking “Blur it”, application reads the set spinner adapter value to know which implementation library is to be referred to perform filtering operations. Depending on that, call is made to corresponding GaussianBlur.tiltBlur\_<java/cpp/neon>() method from respective library from a separate class file GaussianBlur.java which s explained in next section. Bitmap of original image, corresponding parameters for kernel are passed as argument in this call. This method, on completion of blurring operation, returns the output image which gets set on display in place of original image through setImageBitmap() method. We have included the following block of code to load the 'native-lib' library on application startup.

Static { System.loadLibrary("native-lib"); }

We calculate the time it takes starting from when functional call is made to time when final output image is set on UI as required execution time. We achieve this by encapsulating subsequent code block in System.nanoTime() call. Toast messages are always displayed on UI to keep user updated about same.

## Gaussian Blur Class File

This class file contains the complete implementation of Java platform code of tilt shift blur, & subsequent calls to native methods in cpp & neon libraries.

First method tiltBlur\_cpp() is the code section that takes care of C++ platform blurring. This call is made from MainActivity & method receives the original raw image & subsequent a0,a1,a2,a3 and near & far sigma values as parameters. We first perform the scaling of received sigma values & also evaluate width-height of input image to store respective pixels in an array datastructure. Then these updated information is passed to tiltshiftcppnative() method that resides in cpp library & implements the tilt shift blur operation for C++ platform. Similar approach is used for Neon intrinsic case where tiltshift\_neon\_intrinsic() is method that takes care of blurring with Neon as platform. Same steps are followed in this methos as well before a call is made to tiltshiftneonnative() method defined in Neon intrinsic library to perform blurring.

The complete tilt shift blur operation is performed in this class using tiltBlur\_java() method & other supporting utility methods residing in this class. firstTransform() & secondTransform() are the two main helpers methods in the implementation along with other utility methods such as colorGaussBlur(), getGaussian(), kernelMatrix().

Method kernelMatrix() is defined in file to calculate the entire Gaussian vector. Depending on size of radius r, it create the Gaussian vector of size (2\*r+1). For this we utilize Gaussian vector radius for near & far pixels based on sigma value. getGaussian() is the method that calculates this value by implementing the Gaussian kernel formula discussed in MATH section of this document.

firstTransform() method is called to calculate first transform on each pixel of image. We iterate width wise in each row of image. The outer loop runs first from 0th row to a0th, then from a0 to a1 row with new kernel vector based on updated sigma value for this section & so on so forth for a2 to a3 section & for a3 to bottom of image. Similar brute force approach is taken to calculate second transform on image by calling secondTransform() method. Since each pixel is comprised in ARGB\_8888 form (each of A,R,G & B is represented by 8-bits), we calculate row transform for the each pixel on the given index by the calling colorGaussBlur() method that performs the colour wise transform on respective R,G or B bytes. We finally merge these updated bytes in end of both these method to get back the complete value of transformed pixel.

colorGaussBlur() method is used to calculate transform of each color of an individual pixel. This is done by shifting the bits of pixel for respective R,G or B byte & ANDing with 0xff. We then multiply this with corresponding kernel matrix value to get updated value of R,G or B byte of pixel.

# Math

To implement Gaussian blur we perform two independent transforms across each x-y dimensions. To do this, we generate Gaussian kernel vector G, where k ∈ [-r, r] and given by below equation:

If σ is less than 0.6, then we clamp σ to 0.6

After applying the first transform, we get the intermediate matrix, say q(y, x) which is given by below equation :

q(y, x) = G(-r)\*p(y-r, x), +..+ G(0)\*p(y, x),+..+ G(r)\*p(y+r, x)

Similarly for second transform we take the similar approach but here use the intermediate matrix generated in previous step to generate the final output image. This transformation is formulated as below :

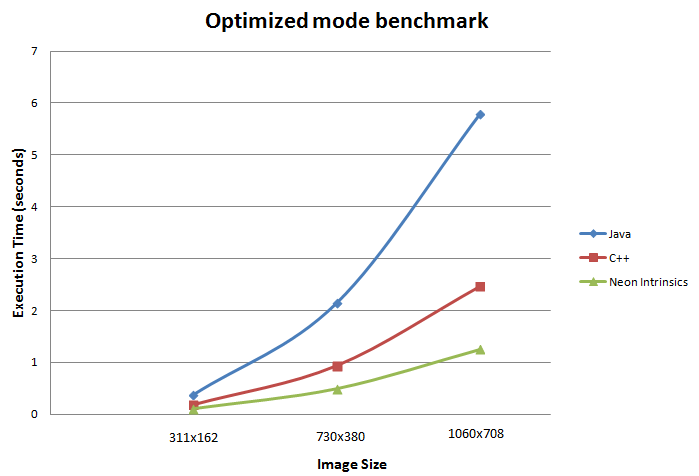
P(y, x) = G(-r)\*q(y, x-r), +..+ G(0)\*q(y, x),+ ..+ G(r)\*q(y,x+r)

# Evaluation & benchmarking

All evaluations & benchmarking tests have been performed on NVIDIA shield tablet as base hardware. We have recorded the observations for both “Performance mode” & “Optimize mode” of NVIDIA Shield tablet. All tests have been run for all three platforms for any selected image. The combinations of varying input testing conditions were used to evaluate the performance. All the subsequent observations have been listed in this section.

 We have tested application all these scenarios for three sample images provided with app as well as a high resolution image for perspectives.

## As seen in Fig. 1, we see a performance improvement in our implementation as we go from Java to C++ and then further improvement as we go to ARM Neon Intrinsics as our platform. The difference is distinctly evident in cases when image size is comparatively large.



Also we observe that the latency of execution is reduced in “Performance mode” of our underlying hardware (NVIDIA Shield tablet) as compared with “Optimized mode”.



The SI unit for magnetic field strength *H* is A/m. However, if you wish to use units of T, either refer to magnetic flux density *B* or magnetic field strength symbolized as µ0*H*. Use the center dot to separate compound units, e.g., “A·m2.”

# Some Common Challenges & Mistakes

Along the development of this project, we encountered few bugs/issues which we have listed in this section. These issues were at UI level as well as were related to implementation logic on occasions.

# Conclusion

## Observing the results, we conclude that due to the vectorization approach implementations, the Neon Intrinsics platform is fastest among the three platforms under testing as it reduces the latencies incurred due to multiple load-stores by vectorizing those operations.

At the same time we wish to note that even though the speed improvements are observed in case of Neon Intrinsics & C++, it comes at the cost of code complexity which is advantage of Java platform when using with Android.

Acknowledgement

We owe a great thanks to everyone who helped and supported us during the completion of this project.

Our deepest thanks go to Prof. Robert LiKamWa, Professor for the subject Mobile Systems Architecture who acted as our mentor and guided us throughout this project. We thank him for giving ideas and briefs for the work that had to be done.

A special thanks to Jinhan Hu, Teaching Assistant, Mobile Systems Architecture, for constant help and guidance in the labs as well as discussion group.

.

References and Footnotes

## References

All listed references must be cited in text at least once. Use number citations that are placed in square brackets and inside the punctuation. Multiple references are each numbered with separate brackets. When citing a section in a book, please give the relevant page numbers. In text, refer simply to the reference number. Do not use “Ref.” or “reference” except at the beginning of a sentence: “Reference [3] shows ... .” Please do not use automatic endnotes in *Word*, rather, type the reference list at the end of the paper using the “References” style.

Reference numbers are set flush left and form a column of their own, hanging out beyond the body of the reference. The reference numbers are on the line, enclosed in square brackets. In all references, the given name of the author or editor is abbreviated to the initial only and precedes the last name. Use them all; use *et al*. only if names are not given. Use commas around Jr., Sr., and III in names. Abbreviate conference titles. When citing IEEE transactions, provide the issue number, page range, volume number, year, and/or month if available. When referencing a patent, provide the day and the month of issue, or application. References may not include all information; please obtain and include relevant information. Do not combine references. There must be only one reference with each number. If there is a URL included with the print reference, it can be included at the end of the reference.

Other than books, capitalize only the first word in a paper title, except for proper nouns and element symbols. For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation See the end of this document for formats and examples of common references. For a complete discussion of references and their formats, see “The IEEE Style Manual,” available as a PDF link off the *Author Digital Toolbox* main page.

## Footnotes

Number footnotes separately in superscripts (Insert | Footnote).[[2]](#footnote-2) Place the actual footnote at the bottom of the column in which it is cited; do not put footnotes in the reference list (endnotes). Use letters for table footnotes (see Table I).

# Submitting Your Paper for Review

## Review Stage Using ScholarOne® Manuscripts

All manuscripts must be submitted through ScholarOne® to be considered for publication in IEEE TMI. Please prepare the manuscript for initial review in the two-column format as detailed in this document, including figures distributed throughout the text near the site of first reference. If you use LATEX to prepare your manuscript, please approximate the format seen visually in this template document.

First submissions must be 10 pages or less unless the editor has granted a rare exception (e.g. challenge papers). Revised manuscripts may exceed 10 pages as required to respond to reviewer comments. Large tables and figures not critical to the main message should be uploaded separately as Supplementary Materials in Supporting Documents.

Submit a pdf version of your manuscript to ScholarOne® using <https://mc.manuscriptcentral.com/tmi-ieee> as a single file. If you wish to include a larger version of some or all of the figures or a video for reviewers, please upload these as a Supplementary file. Do not upload zipped or other compressed files.

Even if this is your first submission, first check to see if you have an existing account on ScholarOne®. If there is none, create a new account as instructed. After logging in, go to your Author Center and click “Submit First Draft of a New Manuscript.” The corresponding author will be asked to submit information for the first 10 authors. Please check to see if they have existing accounts before adding a new account. If there are more than 10 authors and all are listed on the manuscript, then just add the first ten during submission.

Along with other information, you will be asked to select the subject from a pull-down list. You must complete all steps for a complete submission. At the end of each step you must click “Save and Continue”; just uploading the paper is not sufficient. After the last step, you should see a confirmation that the submission is complete. You should also receive an e-mail confirmation. For inquiries regarding the submission of your TMI paper on ScholarOne please contact the editorial office at the email account listed.

You will be asked to file an electronic copyright form immediately upon completing the submission process (authors are responsible for obtaining any security clearances). Failure to submit the electronic copyright could result in publishing delays later. You will also have the opportunity to designate your article as “open access” if you agree to pay the IEEE open access fee.

## Final Stage Using ScholarOne® Manuscripts

Upon acceptance, you will receive an email from the editor with specific instructions regarding the submission of final files. To avoid any delays in publication, be sure to follow these instructions carefully. Final files must be uploaded through ScholarOne as instructed. Final submissions should include source files of your accepted manuscript, high quality graphic files, and a formatted pdf file. If you have any questions regarding the final submission process, please contact the TMI editorial office.

Be sure that the individual designated as the “Corresponding Author” is available during the coming months. This is the only author to whom proofs of the paper will be sent.

## Copyright Form

A completed IEEE copyright form should be submitted no later than at the time of your final submission. You can get a .pdf, .html, or .doc version at <http://www.ieee.org/copyright>*.* Authors are responsible for obtaining any security clearances. Do NOT attach the copyright form to your manuscript at any time during the process.

# Editorial Policy

Do not submit a reworked version of a paper you have submitted or published elsewhere. Do not publish “preliminary” data or results. The submitting author is responsible for obtaining agreement of all coauthors and any consent required from sponsors before submitting a paper.

The IEEE Transactions and Journals Department strongly discourages courtesy authorship. An author is an individual who has made an identifiable intellectual contribution to a manuscript to the extent that the individual can defend its contents. Consequently, organization cannot be an author. It is the obligation of all authors to ensure that relevant prior work has been cited.

If the submission is based on a previously or simultaneously published conference paper, authors are asked to cite the conference paper in the submission, attached the conference paper to Supplemental Materials for reviewer access, and clearly state in a cover letter how the submission extends the work presented in the conference paper. This is important and required since TMI only publishes new materials.

Indecipherable English is a valid reason for immediate rejection. If in the opinions of the reviewers the ideas are not clearly presented, the paper will be rejected as a result of the review. TMI allows only one major review. There is a service available that can help improve your English writing for a fee, and the link to that service can be found at <http://www.ieee.org/web/publications/authors/transjnl/index.html>.

# Publication Principles

IEEE TMI publishes scholarly articles of archival value as well as tutorial expositions and critical reviews of classical subjects and topics of current interest that have been approved by the editor-in-chief prior to first submission.

Authors should consider the following points:

1. Technical papers submitted for publication must advance the state of knowledge and must cite relevant prior work.
2. The length of a submitted paper should be commensurate with the importance, or appropriate to the complexity, of the work. For example, an obvious extension of previously published work might not be appropriate for publication or might be adequately treated in just a few pages.
3. Authors must convince both peer reviewers and the editors of the scientific and technical merit of a paper; the standards of proof are higher when extraordinary or unexpected results are reported.
4. Because replication is required for scientific progress, papers submitted for publication must provide sufficient information to allow readers to perform similar experiments or calculations and use the reported results. Although not everything need be disclosed, a paper must contain new, useable, and fully described information. For example, a specimen’s chemical composition need not be reported if the main purpose of a paper is to introduce a new measurement technique. Authors should expect to be challenged by reviewers if the results are not supported by adequate data and critical details.
5. Papers that describe ongoing work or announce the latest technical achievement, which are suitable for presentation at a professional conference, may not be appropriate for publication.

References

1. https://developer.android.com/guide/
2. https://community.arm.com/android-community/b/android/posts/arm-neon-programming-quick-reference
3. W.-K. Chen, Linear *Networks and Systems.* Belmont, CA: Wadsworth, 1993, pp. 123–135.

*Basic format for periodicals:*

1. J. K. Author, “Name of paper,” *Abbrev. Title of Periodical*, vol. *x,* no. *x,* pp*. xxx-xxx,* Abbrev. Month, year.

*Examples:*

1. J. U. Duncombe, “Infrared navigation—Part I: An assessment   
   of feasibility,” *IEEE Trans. Electron Devices*, vol. ED-11, no. 1, pp. 34–39, Jan. 1959.
2. E. P. Wigner, “Theory of traveling-wave optical laser,” *Phys. Rev*.,   
   vol. 134, pp. A635–A646, Dec. 1965.
3. E. H. Miller, “A note on reflector arrays,” *IEEE Trans. Antennas Propagat*., to be published.

*Basic format for reports:*

1. J. K. Author, “Title of report,” Abbrev. Name of Co., City of Co., Abbrev. State, Rep. *xxx*, year.

*Examples:*

1. E. E. Reber, R. L. Michell, and C. J. Carter, “Oxygen absorption in the earth’s atmosphere,” Aerospace Corp., Los Angeles, CA, Tech. Rep. TR-0200 (4230-46)-3, Nov. 1988.
2. J. H. Davis and J. R. Cogdell, “Calibration program for the 16-foot antenna,” Elect. Eng. Res. Lab., Univ. Texas, Austin, Tech. Memo. NGL-006-69-3, Nov. 15, 1987.

*Basic format for handbooks:*

1. *Name of Manual/Handbook*, *x* ed., Abbrev. Name of Co., City of Co., Abbrev. State, year, pp. *xxx-xxx.*

*Examples:*

1. *Transmission Systems for Communications*, 3rd ed., Western Electric Co., Winston-Salem, NC, 1985, pp. 44–60.
2. *Motorola Semiconductor Data Manual*, Motorola Semiconductor Products Inc., Phoenix, AZ, 1989.

*Basic format for books (when available online):*

1. Author. (year, month day). *Title.* (edition) [Type of medium]. *volume (issue).* Available: site/path/file

*Example:*

1. J. Jones. (1991, May 10). *Networks.* (2nd ed.) [Online]. Available: [http://www.atm.com](http://www.atm.com/)

*Basic format for journals (when available online):*

1. Author. (year, month). Title. *Journal.* [Type of medium]. *volume (issue),* pages. Available: site/path/file

*Example:*

1. R. J. Vidmar. (1992, Aug.). On the use of atmospheric plasmas as electromagnetic reflectors. *IEEE Trans. Plasma Sci.* [Online]. *21(3),* pp. 876–880. Available:<http://www.halcyon.com/pub/journals/21ps03-vidmar>

*Basic format for papers presented at conferences (when available online):*

1. Author. (year, month). Title. Presented at Conference title. [Type of Medium]. Available: site/path/file

*Example:*

1. PROCESS Corp., MA. Intranets: Internet technologies deployed behind the firewall for corporate productivity. Presented at   
   INET96 Annual Meeting. [Online]. Available: <http://home.process.com/Intranets/wp2.htp>

*Basic format for reports and handbooks (when available online):*

1. Author. (year, month). Title. Comp an y . C ity, State or Country. [Type of Medium]. Available: site/path/file

*Example:*

1. S. L. Tall een. (1996 , Apr . ). The In t r an et Archi -tecture: M a nagi ng i n f o rm at i on i n t h e ne w paradigm. Amdahl Corp., CA. [Online]. Available:<http://www.amdahl.com/doc/products/bsg/intra/infra/html>

*Basic format for computer programs and electronic documents (when available online):* ISO recommends that capitalization follow the accepted practice for the language or script in which the information is given.

*Example:*

1. A. Harriman. (1993, June). Compendium of genealogical software. *Humanist.* [Online]. Available e-mail: [HUMANIST@NYVM.ORG](mailto:HUMANIST@NYVM.ORG) Message: get GENEALOGY REPORT

*Basic format for patents (when available online):*

1. Name of the invention, by inventor’s name. (year, month day). *Patent Number* [Type of medium]. Available: site/path/file

*Example:*

1. Musical toothbrush with adjustable neck and mirror, by L.M.R. Brooks. (1992, May 19). *Patent D 326 189*

[Online]. Available: NEXIS Library: LEXPAT File: DESIGN

*Basic format for conference proceedings (published):*

1. J. K. Author, “Title of paper,” in *Abbreviated Name of Conf.*, City of Conf., Abbrev. State (if given), year, pp. *xxxxxx.*

*Example:*

1. D. B. Payne and J. R. Stern, “Wavelength-switched pas- sively coupled single-mode optical network,” in *Proc. IOOC-ECOC,* 1985,   
   pp. 585–590.

*Example for papers presented at conferences (unpublished):*

1. D. Ebehard and E. Voges, “Digital single sideband detection for interferometric sensors,” presented at the 2nd Int. Conf. Optical Fiber Sensors, Stuttgart, Germany, Jan. 2-5, 1984.

*Basic format for patents:*

1. J. K. Author, “Title of patent,” U.S. Patent *x xxx xxx*, Abbrev. Month, day, year.

*Example:*

1. G. Brandli and M. Dick, “Alternating current fed power supply,”   
   U.S. Patent 4 084 217, Nov. 4, 1978.

*Basic format**for theses (M.S.) and dissertations (Ph.D.):*

1. J. K. Author, “Title of thesis,” M.S. thesis, Abbrev. Dept., Abbrev. Univ., City of Univ., Abbrev. State, year.
2. J. K. Author, “Title of dissertation,” Ph.D. dissertation, Abbrev. Dept., Abbrev. Univ., City of Univ., Abbrev. State, year.

*Examples:*

1. J. O. Williams, “Narrow-band analyzer,” Ph.D. dissertation, Dept. Elect. Eng., Harvard Univ., Cambridge, MA, 1993.
2. N. Kawasaki, “Parametric study of thermal and chemical nonequilibrium nozzle flow,” M.S. thesis, Dept. Electron. Eng., Osaka Univ., Osaka, Japan, 1993.

*Basic format for the most common types of unpublished references:*

1. J. K. Author, private communication, Abbrev. Month, year.
2. J. K. Author, “Title of paper,” unpublished.
3. J. K. Author, “Title of paper,” to be published.

*Examples:*

1. A. Harrison, private communication, May 1995.
2. B. Smith, “An approach to graphs of linear forms,” unpublished.
3. A. Brahms, “Representation error for real numbers in binary computer arithmetic,” IEEE Computer Group Repository, Paper R-67-85.

*Basic format for standards:*

1. *Title of Standard*, Standard number, date.

*Examples:*

1. IEEE Criteria for Class IE Electric Systems, IEEE Standard 308, 1969.
2. Letter Symbols for Quantities, ANSI Standard Y10.5-1968.

**Note that author photographs and biographies are not accepted in IEEE TMI papers. This policy can vary among the different transactions.**

**This template has been modified from the IEEE Template for**

**Transactions specifically for TMI by the TMI Editorial Office, February 2016.**

1. This project was submitted for evaluation on October 5th, 2018 at Arizona State University.

   All development done for this project is licensed under Arizona State University. Authors Sarvesh Patil, Shubham Nandanwankar & Achal Shah were with Arizona State University during the development of this project (2018). This project was developed under the guidance of Professor Robert LiKamWa of Arizona State University (2018). [↑](#footnote-ref-1)
2. It is recommended that footnotes be avoided (except for the unnumbered footnote with the receipt date on the first page). Instead, try to integrate the footnote information into the text. [↑](#footnote-ref-2)